A Split-and-Privatize Framework for Large Language Model Fine-Tuning

Anonymous ACL submission

Abstract

Fine-tuning is a prominent technique to adapt a pre-trained language model to downstream scenarios. In parameter-efficient fine-tuning, only a small subset of modules are trained over the downstream datasets, while leaving the rest of the pre-trained model frozen to save computation resources. In recent years, a popular productization form arises as Model-as-a-Service (MaaS), in which vendors provide abundant pre-trained language models, server resources and core functions, and customers can fine-tune, deploy and invoke their customized model by accessing the one-stop MaaS with their own private dataset. In this paper, we identify the model and data privacy leakage risks in MaaS fine-tuning, and propose a Split-and-Privatize (SAP) framework, which manage to mitigate the privacy issues by adapting the existing split learning architecture. Furthermore, we propose a contributing-token-identification (CTI) method to alleviate the utility degradation caused by privatization. The proposed framework is sufficiently investigated by experiments, and the results indicate that it can enhance the empirical privacy by 65% at the cost of 1% model performance degradation on the Stanford Sentiment Treebank dataset.

1 Introduction

In recent years, pre-trained language models (PLMs) represented by BERT (Kenton and Toutanova, 2019) and GPT (Brown et al., 2020) have demonstrated powerful text learning capabilities and have been widely used in various fields such as law (Jiang and Yang, 2023), finance (Arslan et al., 2021), and healthcare (Arora and Arora, 2023). To improve the adaptability of a PLM on downstream applications, it is necessary to fine-tune it on datasets related to the downstream tasks. Considering that PLMs contain hundreds of millions of parameters, researchers have proposed several parameter-efficient fine-tuning (PEFT) algorithms to reduce the cost of secondary training (Ding et al., 2023), such as LoRA (Hu et al., 2021) and prompt tuning (Lester et al., 2021). In practice, most users are unable to independently acquire the PLM and perform fine-tuning due to resource or technical constraints, which has given rise to a new business direction known as model-as-a-service (MaaS). In MaaS, enterprises with ample resources and technical capabilities (called vendors) release PLMs in the form of cloud services and provide customers with a fine-tuning API so that they can customize their own LLM based on private data.

However, while this solution provides customers with efficient and customizable LLM services, it also carries the risk of privacy leakage. On the one hand, since the pre-training process requires a large amount of computational overhead, the weights of PLMs are typically considered as proprietary assets of vendors and cannot be made public. On the other hand, customers' text data usually contains sensitive information such as identity and age, so directly transmitting the original data or representations to the vendor may result in serious privacy leaks (Pan et al., 2020; Qu et al., 2021; Song and Raghunathan, 2020), which hinders privacy-conscious customers from using the customization service. Therefore, there is an urgent need for a privacy-preserving fine-tuning framework to alleviate privacy concerns and promote the development of customized services for LLM.

Some prior works have ventured into this domain, albeit encountering certain challenges along the way. For example, the work (Qu et al., 2021) proposed a text privatization mechanism based on $d\chi$-privacy, where the consumer perturbs each individual data entry locally before releasing it to the vendor. But it must be acknowledged that implementing text privatization will invariably lead to performance degradation on downstream tasks (Qu et al., 2021; Li et al., 2023), involving the delicate trade-off between utility and privacy. In order to protect both parties' privacy and achieve efficient

fine-tuning, the work (Xiao et al., 2023) proposed the offsite-tuning framework, in which the vendor sends a lightweight adapter and a lossy compressed emulator to the customer, and the customer performs fine-tuning and then returns the final adapter. However, this framework does not consider privacy concerns during the inference phase.

To address the challenges mentioned above, we propose a Split-and-Privatize (SAP) federated fine-tuning framework based on the existing split learning architecture (Vepakomma et al., 2018; Ceballos et al., 2020). Specifically, the vendor first splits the entire PLM into a bottom model and a top model, and then sends the bottom model to the customer while preserving the confidentiality of the majority of the PLM. During fine-tuning, the customer feeds local sensitive data into the bottom model and privatizes the outputs by applying privacy-preserving mechanisms before sending them to the vendor. Furthermore, to improve the utility-privacy trade-off caused by privatization, we propose a contributing-token-identification (CTI) method. By reducing the perturbation to a small number of token representations that are strongly related to the utility task, we significantly improve the utility performance while maintaining a similar level of empirical privacy.

In order to comprehensively evaluate the performance and security of the proposed framework, we conduct a series of experiments covering multiple tasks, including sentiment analysis, topic classification and semantic equivalence judgment. Additionally, we conduct simulated privacy attacks to validate the effectiveness of SAP on protecting data privacy. Experimental results indicate that the proposed framework can achieve a good balance between protecting model privacy and data privacy while maintaining competitive performance.

2 Related Work and Preliminary

2.1 Related Work

2.1.1 Split Learning

Split learning (SL) is a distributed learning technique that divides the entire model into multiple segments held by different parties, allowing multiple parties to collaboratively train the model without revealing their original data (Vepakomma et al., 2018). In SL, split neural network (SplitNN) is the most commonly used paradigm (Romanini et al., 2021). Specifically, the entire neural network is split into a top network on the server and multiple bottom networks held by different clients. During training, each client transforms its local input data into intermediate features and sends them to the server. The server first concatenates all the intermediate features and continues to perform forward propagation in the top model to compute the loss. Then it performs backpropagation to compute the gradient and sends the corresponding intermediate-layer gradient to each client so that they can complete the update of their local network segment.

Although SL has the advantage of avoiding the disclosure of raw data, some studies have shown that there is still a potential risk of privacy leakage when clients directly transmit intermediate representations (Dosovitskiy and Brox, 2016; He et al., 2020). For instance, the work (He et al., 2020) proposed attack methods for both the white-box and black-box scenarios, which can partially recover the original inputs from the transmitted representations.

2.1.2 Privacy-preserving LLM Services

Existing research on privacy-preserving LLM mainly focuses on centralized learning and addresses concerns about the potential privacy leakage of training data when deploying LLMs publicly, which is referred to as the memory privacy of LLMs (Carlini et al., 2021; Peris et al., 2023). For the MaaS scenario, a few of studies effort to propose methods that protect both model privacy and data privacy. Typically, the vendor keeps the backbone of the PLM confidential at the cloud server and only releases the embedding layer. The customer is then required to send perturbed texts or text representations to the vendor to complete subsequent fine-tuning and inference. The work (Lyu et al., 2020) proposed a method based on differential privacy (DP) and word dropout, which protects data privacy during the inference phase by randomly dropping some words and adding Gaussian noise to the text representation. Chen et al. (Qu et al., 2021) investigated the impact of applying the $d\chi$-privacy mechanism (a variant of local DP) to BERT fine-tuning on both privacy and utility, and proposed a privacy-adaptive LLM pre-training method, which applies the same perturbations to the pre-training corpus to improve the practicality of the fine-tuned model. Compared with our work, (Qu et al., 2021) requires retraining the PLM based on the designed masked LM objective on the publicly available corpora, which incurs significant computation costs. Besides, considering the

high cost of fine-tuning the entire model on private data, Li et al. (Li et al., 2023) proposed a privacy-preserving prompt tuning framework called RAPT, where the customer applies text-to-text privatization based on the $d\chi$-privacy locally and the vendor performs prompt tuning on the privatized data, which also introduced a token reconstruction task to learn better task-related representations. In comparison, we propose a more general federated fine-tuning framework SAP. By splitting some encoder blocks to the bottom model instead of just the embedding layer, the SAP framework achieves a better trade-off between model performance and data privacy. Furthermore, compared with the token reconstruction method, the proposed CTI method saves extra resources required for plain token transmission and reconstructing model training.

2.2 Preliminary

In this subsection, we introduce some preliminaries about $d\chi$-privacy mechanism used in this paper, which is a variant of LDP (Chatzikokolakis et al., 2013). The specific definition of $d\chi$-privacy is given below.

Definition 1. *A randomized mechanism $\mathcal{M}$ satisfies $\eta d\chi$-privacy if for any two inputs $x, x' \in \mathcal{X}$,*

$$\frac{\Pr[M(x) = y]}{\Pr\left[M\left(x'\right) = y\right]} \leq e^{\eta d(x,x')}, \forall y \in \mathcal{Y}, \quad (1)$$

where $\eta > 0$ is a privacy parameter and $d(x, x')$ is a distance function.

Compared with the definition of LDP, $d\chi$-privacy replaces the exponent term on the right side of the inequality (1) from ϵ to $\eta d\left(x, x'\right)$, so it is a relaxation of LDP. LDP is a strong privacy standard, which requires that any two inputs have similar and indistinguishable output distributions regardless of how different they are. Therefore, the output may not retain enough information of the original input, resulting in severe performance degradation. In contrast, $d\chi$-privacy allows the indistinguishability of the output distributions to be scaled by the distance between inputs, which enables the randomized mechanism to retain more information about input.

The work (Feyisetan et al., 2020) proposed a text-to-text privatization mechanism that guarantees the $\eta d\chi$-privacy. For any word w, the mechanism first computes the embedding vector $\phi(w)$, and then adds appropriate random noise $\boldsymbol{n}$ to obtain the perturbed vector $\hat{\phi}(w) = \phi(w) + \boldsymbol{n}$, where the probability density of the noise satisfies $p(\boldsymbol{n}) \propto \exp(-\eta\|\boldsymbol{n}\|)$. Finally the word w is replaced by the word w' closest to $\hat{\phi}(w)$.

3 Problem Setting

3.1 Problem Definition

In this paper, we focus on the customization of LLM involving two parties, where the vendor holds the complete PLM $\boldsymbol{w}$ and abundant server resources, and the customer holds the private labeled dataset $\mathcal{D} := \{(\boldsymbol{x}_i, y_i)|i = 1, 2, \ldots, |\mathcal{D}|\}$ and limited computation resources. In order to achieve optimal adaptation on the downstream task, the vendor and customer need to collaboratively fine-tune the PLM, which can be formulated as

$$\arg\min_{\boldsymbol{\delta}} \mathcal{L}(\boldsymbol{w} + \boldsymbol{\delta}, \mathcal{D}). \quad (2)$$

However, due to privacy constraints, the above fine-tuning process cannot be performed in a centralized manner on one party. Specifically, the privacy constraints include that the vendor cannot share the PLM $\boldsymbol{w}$ with the customer, and the customer cannot share the private dataset $\mathcal{D}$ with the vendor.

3.2 Threat Model and Design Goals

For the threat model, we assume that both participants are honest-but-curious, that is, they always follow the designed framework but are curious about other's private information (i.e. the private data of customer and the model parameters of vendor). The customer might peek at the model architecture and parameters transferred from the vendor, while the vendor may attempt to infer some privacy information from text representations transmitted from the customer, such as the embedding inversion attack (Qu et al., 2021) and the attribute inference attack (Song and Raghunathan, 2020).

Under the above threat model, the proposed framework aims to achieve the following goals. Firstly, most parameters of the PLM cannot be disclosed to the customer. Secondly, the framework should ensure that it is difficult for the vendor to recover the original input text from the transmitted representations. Lastly, the utility performance should not degrade significantly compared to centralized fine-tuning.

4 Proposed Method

In this section, we first give an overview of the proposed framework and then introduce two important modules (model split and text privatization) in

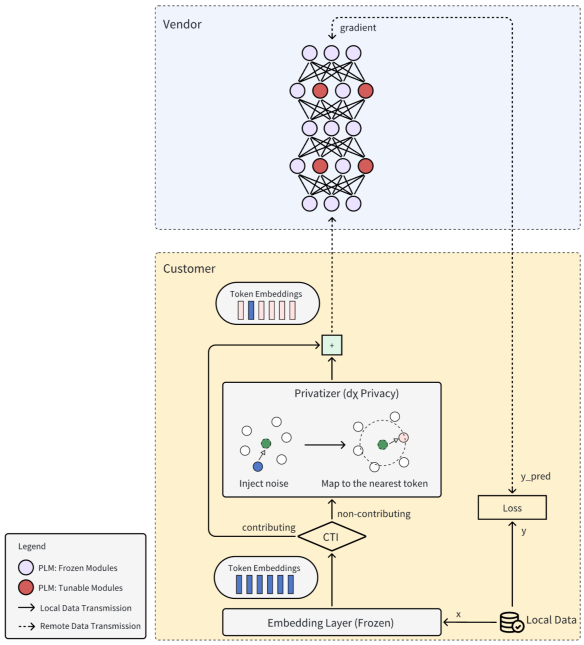


Figure 1: An overview of the SAP framework with CTI, where the PLM is split into a bottom model (embedding layer) and a top model, and the customer privatizes the embedding vectors before releasing them.

detail. Following that, the CTI method is proposed to improve the utility-privacy trade-off.

4.1 Split-and-Privatize Framework

To protect both the vendor's model privacy and the customer's data privacy while achieving LLM customization, we propose the SAP framework based on the split learning architecture, as shown in Figure 1. In general, the entire PLM is split into a top model on the vendor and a bottom model on the customer. During fine-tuning, the customer first computes the outputs of bottom model on local private data, adaptively privatize the text representations and sends the results to vendor. After receiving the perturbed representations, the vendor proceeds to perform forward propagation in the top model to compute the output of the PLM. Since the sample labels are also held by the customer, to update trainable parameters in the PLM such as the LoRA module, the vendor needs to send the output to customer and receive the gradients of the output layer in return. The following subsections provide detailed descriptions of the SAP framework.

4.2 Model Split

Similar to the idea of SplitNN (Romanini et al., 2021), the vendor splits the PLM into a bottom model and a top model, and sends the bottom model to the customer before fine-tuning. The choice of which layer to split the model is an important option in SAP. If the bottom model only has an embedding layer, the customer's computational burden is relatively small, but the vendor can easily recover the input text from the transmitted representations by nearest neighbor search (Qu et al., 2021). If the bottom model contains more encoder blocks, it becomes more difficult to recover the original text. The work (Song and Raghunathan, 2020) demonstrated that inverting input text from higher layers of a deep model is more challenging, as the representations at higher layers are more abstract and generic. However, since the weights of PLM are also important assets, the vendor may request to release as few weights as possible. Therefore, determining the split position of PLM requires a comprehensive consideration of multiple factors.

Besides, the SAP framework can be divided into two cases depending on whether the bottom model is trainable or not. For the case where the bottom model is frozen, the customer computes the representations of all samples after receiving the bottom model, adds perturbations and sends them to the vendor all at once. Therefore, the customer does not need to repeatedly perform forward computations in the fine-tuning phase, which significantly reduces computational and communication overhead. Another implementation is to make the bottom model also trainable. It is hoped that by altering the parameters of bottom model during fine-tuning, it will be more challenging for the vendor to infer the original input text. In this paper, we focus on the case where the bottom model is frozen due to page limitations. A detailed discussion on the other case is presented in Appendix A.

4.3 Text Privatization

Given the bottom model, the customer can obtain the text representation for each sample. However, if plain representations are directly released, the vendor might be able to accurately recover the original input text (Song and Raghunathan, 2020). Therefore, to achieve stronger privacy protection, it is necessary for the customer to employ privatization mechanisms to perturb text representations. We take the case where the bottom model is a frozen embedding layer as an example to demonstrate how to combine the SAP framework with the privatization mechanism proposed in (Feyisetan et al., 2020) to guarantee $\eta d\chi$-privacy.

Let $[x_i^1, x_i^2, \ldots, x_i^n]$ represent a sequence of to-

kens for the input text $\boldsymbol{x}_i$. The customer first obtains the embedding vector $\phi(x_i^j)$ for each token x_i^j in the sample $\boldsymbol{x}_i$ based on the embedding layer. Then independent random noise $\boldsymbol{n}$ is added to each embedding vector,

$$\hat{\phi}(x_i^j) = \phi(x_i^j) + \boldsymbol{n}, \quad p(\boldsymbol{n}) \propto \exp(-\eta \|\boldsymbol{n}\|). \quad (3)$$

The specific generation method of noise $\boldsymbol{n}$ can be found in Section 2.6 of (Feyisetan et al., 2020). Then the perturbed vector is replaced by its nearest neighbor in the embedding space,

$$\bar{\phi}(x_i^j) = \arg\min_{\boldsymbol{w}_l} \left\| \hat{\phi}(x_i^j) - \boldsymbol{w}_l \right\|, \quad (4)$$

where $\boldsymbol{w}_l$ represents the vector in the embedding space. Finally, the customer sends $|\mathcal{D}|$ perturbed sequences $\bar{\phi}(\boldsymbol{x_i}) = [\bar{\phi}(x_i^1), \bar{\phi}(x_i^2), \ldots, \bar{\phi}(x_i^n)]$ to the vendor.

4.4 Contributing Token Identification

Although text privatization strengthens the protection of data privacy, fine-tuning PLM on the perturbed representations will inevitably lead to performance degradation on the downstream task, so there is a trade-off between utility and privacy. To improve the utility-privacy trade-off of the SAP framework, we propose a contributing-token-identification (CTI) method. The key rationale of this method is to use statistical analysis to identify the tokens that contribute the most to the utility target in each class of samples, and then reduce the perturbations applied to these specific tokens, aiming to improve utility performance while maintaining a similar level of privacy protection. The following is a detailed description of this method.

In natural language processing, term frequency-inverse document frequency (TF-IDF) (Salton and Buckley, 1988) is a metric used to measure the importance of a word to a document in a collection or corpus, which is widely used in information retrieval and text mining. The TF-IDF value is proportional to the frequency of a word appearing in a document, and inversely proportional to the proportion of documents containing this word in the corpus, thereby reducing the impact of common words. Inspired by TF-IDF, we propose a metric that measures the importance of each token in relation to the utility target for text classification tasks. Let $p(t = t_m | y = c)$ represent the frequency of token t_m appearing in the c-th class of samples, then the utility importance (UI) of token t_m to class c is defined as

$$\mathrm{UI}_{mc} = \frac{1}{N-1} \sum_{c', c' \neq c} \ln \frac{p(t = t_m | y = c)}{p(t = t_m | y = c')}, \quad (5)$$

where $\ln \frac{p(t=t_m|y=c)}{p(t=t_m|y=c')}$ can be regarded as the difference between the probability distribution of tokens in the c-th class of samples and that in the c'-th class of samples specifically at token t_m, and N is the number of categories. Intuitively, tokens that appear frequently in the c-th class of samples while having low frequency in other class of samples will be considered to contribute significantly to distinguishing the c-th class from other classes, and thus will be assigned a larger UI value.

For each class, the customer first computes the utility importance of each token in the vocabulary. Then, when applying text privatization locally, the customer adaptively assigns different privacy parameters to each token according to

$$\eta_{mc} = \frac{2\eta_0}{1 + \exp(-\mathrm{UI}_{mc} + c_0)}, \quad (6)$$

where η_0 is the basic privacy budget and c_0 is a constant. Compared to a fixed privacy budget η_0 for all tokens, the customer reduces the perturbation to the embedding vectors of tokens with larger UI values and increases the perturbation to those with smaller UI values, thereby achieving a better utility-privacy trade-off.

4.5 Implementation

In specific implementation, the SAP framework can utilize a variety of existing PEFT algorithms to reduce computation cost during fine-tuning. In the inference phase, the customer can perform the same forward computation and text privatization locally to protect the inference data.

5 Empirical Experiments

5.1 Experiment Settings

To begin with, let us introduce the experiment settings, including the PLM, datasets, attack methods, and implementation details.

Model and Datasets. In the experiments, we use the Roberta-Large (Liu et al., 2019) published by Huggingface[1] as the PLM, which consists of a total of 355 million parameters. We evaluate the SAP framework on the Financial Phrasebank (FP) (Malo et al., 2014), the Blog dataset used in (Lyu

[1]https://huggingface.co/roberta-large

Dataset	Task	#Train	#Dev	#Test	Centralized Accuracy (%)
FP	sentiment analysis	1808	226	226	98.75
Blog	topic classification	7098	887	887	96.71
SST	sentiment analysis	66675	674	872	95.89
MRPC	equivalence judgment	3301	367	408	89.42

Table 1: Description of the used datasets and performance of centralized fine-tuning on these datasets.

et al., 2020), as well as Stanford Sentiment Treebank (SST) and Microsoft Research Paraphrase Corpus (MRPC) datasets from the GLUE benchmark[2] (Wang et al., 2019). Detailed descriptions of these datasets are provided in Table 1.

Attack Methods. Following the work (Song and Raghunathan, 2020), simulated attacks are employed to investigate the capability of SAP framework to protect customer's data privacy. To maximize the attacker's abilities, we consider the white-box setting, assuming that the attacker has the same perspective as the vendor and can access the perturbed text representations transmitted by customer as well as the parameters of bottom model. The attack methods used are listed as follows:

- Embedding inversion attack (EIA) is a token-level attack whose goal is to recover the original input text from the perturbed text representations. Specifically, for the case where the bottom model only has the embedding layer, the nearest neighbor of each perturbed embedding is searched in the embedding space as a prediction of the original token (Qu et al., 2021). For the case where the bottom model contains more layers, a complex optimization-based attack method proposed in (Song and Raghunathan, 2020) is employed. For each input sample, the method iteratively optimizes the word selection vectors by minimizing the distance between the predicted text's representations and the observed representations.
- Attribute inference attack (AIA) aims to infer sensitive attributes of users from the text representations. In this paper, it is assumed that the attacker can obtain privacy attribute labels of some samples, such as the author's gender in the Blog dataset. The privacy inference problem is then treated as a downstream task, and a classifier is trained using the text representations of these samples and the corresponding privacy labels. After training, the attacker can predict the privacy attributes of other samples by feeding their representations into the classifier.

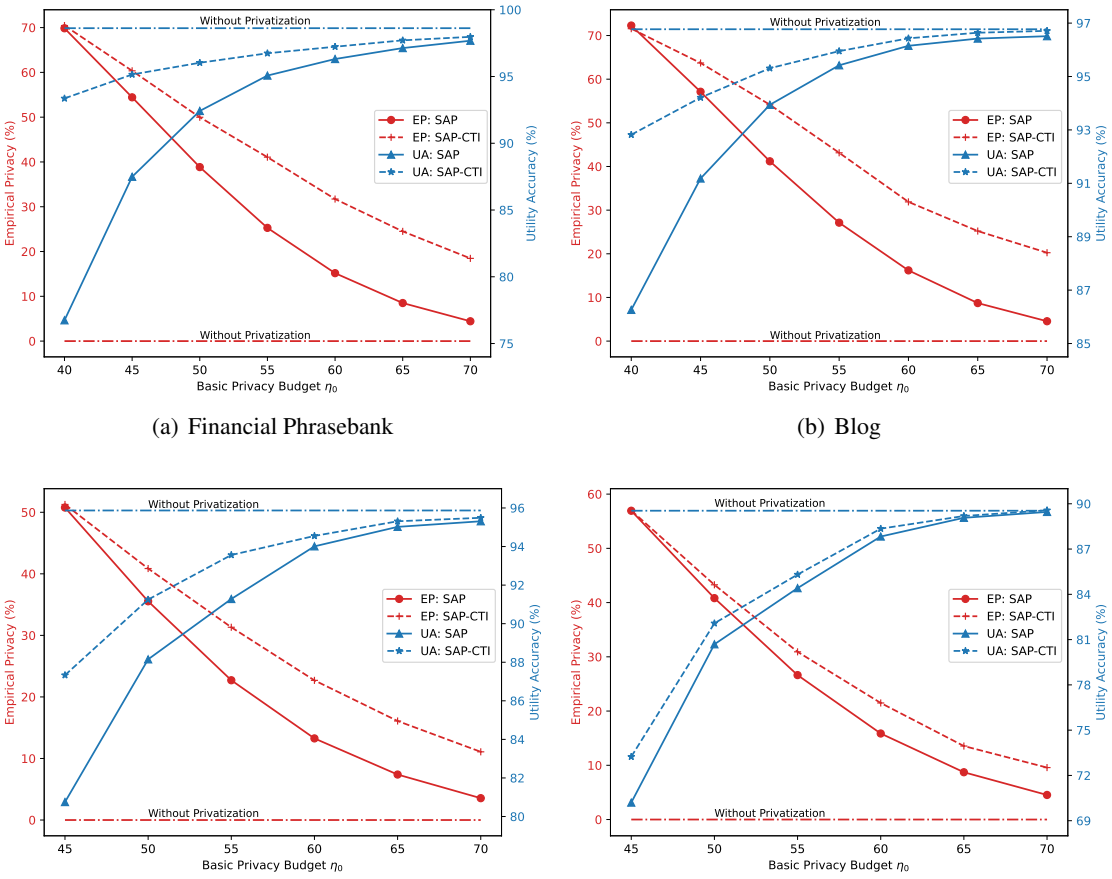
[2]https://gluebenchmark.com

Implementation Details. Our experiments are implemented based on the Transformers library and PEFT library of Huggingface. Specifically, the LoRA (Hu et al., 2021) method is adopted to fine-tune the Roberta-Large model, and the AdamW optimizer with a linear learning rate scheduler is used during fine-tuning, where the initial learning rate is set to 3e-4. Empirically, the constant c_0 in Equation 6 is set to $(\max(\text{UI}_{mc}) + \min(\text{UI}_{mc}))/2$. Following the work (Li et al., 2023), we use utility classification accuracy (UA) and empirical privacy (EP) as metrics to evaluate utility performance and privacy protection capability, where empirical privacy is defined as $1-X$ and X represents the attack success rate.

5.2 Effectiveness of SAP with CTI

First, we evaluate the performance and security of the proposed framework under different privacy parameter η_0 when the bottom model is a frozen embedding layer. From the results in Figure 2, it can be observed that model split without text privatization does not result in performance loss compared with the centralized fine-tuning accuracy given in Table 1. However, if embedding vectors are released without privatization, the attacker can easily recover the input text through EIA, with an attack success rate of up to 100%. By adding perturbations to text representations to guarantee ηd_χ-privacy, the privacy protection capability of the SAP framework is strengthened. As the basic privacy parameter η_0 decreases, we obtain better empirical privacy against EIA, but at the same time, the utility accuracy keeps decreasing. In other words, the SAP framework involves a trade-off between utility and privacy. For example, on the FP dataset, SAP improves the empirical privacy to 38.85% at the cost of 6.17% performance degradation when η_0 is set to 50.

Furthermore, the experimental results on the FP, Blog, and SST datasets demonstrate that the



(a) Financial Phrasebank (b) Blog

(c) Stanford Sentiment Treebank (d) Microsoft Research Paraphrase Corpus

Figure 2: Impact of the privacy parameter η_0 on the empirical privacy (EP) against EIA and utility accuracy (UA).

CTI method significantly improves the trade-off between utility and privacy of the SAP framework. By adaptively adjusting the privacy budget according to the utility importance of each token for the target task, the CTI method can significantly improve both UA and EP. Specifically, on the FP dataset with η_0 set to 50, the SAP-CTI algorithm can achieve the empirical privacy of 49.98% with only 2.73% performance loss. However, the CTI method does not yield very significant improvement on the MRPC dataset. This is because the CTI method only performs correlation analysis at the token level, while for tasks like semantic equivalence judgment, the class label of each sample is less relevant to individual tokens and more relevant to the overall semantics of the sentence. This is an inherent limitation of the CTI method.

Figure 3 presents the results of the SAP framework defending against AIA on the Blog dataset with different numbers of labeled data N_l. Specifically, the attacker fine-tunes the Roberta model using some auxiliary gender labels along with the corresponding text representations sent by the customer to infer the gender labels of other samples. The results indicate that the attack success rate of attacker is positively related to the amount of labeled data it possesses. By reducing the privacy parameter, the SAP framework becomes more capable of defending against AIA, which is consistent with the results of EIA.

5.3 Impact of Split Position

In the SAP framework, the split position of the PLM is an important option. The Roberta model comprises a total of 24 encoder blocks. In the experiment, we split the model after the 1st to 8th encoder block and compare them with the case where the bottom model only has the embedding layer. Figure 4 gives the impact of different split positions on the empirical privacy against EIA of the SAP framework without text privatization. The results show that as the number of encoder blocks in the bottom model increases, it becomes increasingly difficult for an attacker to infer the input text

Bottom Model	Metric	Privacy Parameter η_0						
		45	50	55	60	65	70	None
Embedding and 2 encoder blocks	EP	55.28	46.03	37.46	31.18	25.79	21.42	19.68
	UA	87.21	91.08	93.24	94.38	95.30	95.41	95.84
Embedding and 4 encoder blocks	EP	72.15	67.98	59.13	51.42	45.34	40.97	37.80
	UA	87.06	90.38	93.21	94.25	94.89	95.13	95.72
Embedding and 6 encoder blocks	EP	80.45	75.22	71.58	68.06	64.83	61.41	59.19
	UA	86.79	90.51	93.16	93.61	94.77	95.06	95.53

Table 2: Empirical privacy against EIA (%) and utility accuracy (%) of SAP-CTI with different split positions and different privacy parameter settings on SST dataset, where "None" represents the case without privatization.

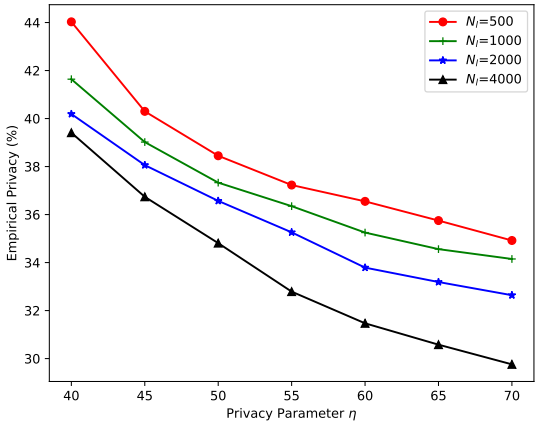


Figure 3: Results of the SAP framework defending against AIA on the Blog dataset.

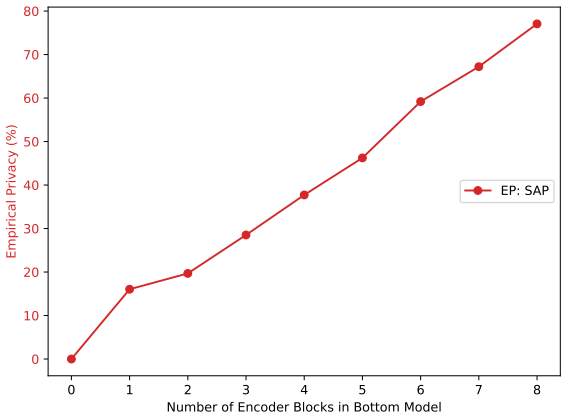


Figure 4: Empirical privacy against EIA of SAP framework (without privatization) with different split positions on the SST dataset.

from the representations transmitted by the customer. Even without text privatization, the empirical privacy reaches about 80% when there are 8 encoder blocks in the bottom model.

Furthermore, we delve into the privacy protection capability and utility performance of the SAP framework with different split positions and different privacy parameter settings. Compared with centralized fine-tuning, the results in the last column of Table 2 indicate that as the number of layers included in the bottom model increases, the UA of the SAP framework without privatization decreases slightly while the EP increases significantly. In addition, we can observe that by applying text privatization and reducing the privacy parameter, EP is further strengthened, but at the same time, the UA also decreases, which is consistent with the results in Figure 2.

6 Conclusion

We consider the model privacy and data privacy issues in LLM customization and propose a privacy-preserving fine-tuning framework SAP, along with a utility enhancement method called CTI. By splitting the PLM into a top model on the vendor and a bottom model on the customer, and applying text privatization to adaptively perturb the transmitted representations, the SAP framework with CTI can effectively protect both the backbone model privacy and data privacy while maintaining competitive performance. Moreover, SAP is a flexible framework capable of adapting to various scenarios in LLM customization services. For customers with limited computation resources, it is recommended to adopt the solution with a frozen embedding layer in the bottom model. Experimental results indicate that it enhances the empirical privacy by 40% at the cost of 4.6% performance degradation on the SST dataset. For customers with relatively abundant resources, a solution with more encoder blocks in the bottom model can be adopted. Results indicate that it can enhance the empirical privacy by 65% at the cost of 1% performance degradation on the SST dataset when the bottom model contains 6 encoder blocks.

Limitations

One limitation of this work is that it only considers text classification tasks. The effectiveness of the SAP framework on more complex text generation tasks, particularly the impact of text privatization on the usability of generated content, remains to be further researched.

In addition, the proposed CTI method analyzes the utility importance of each token for the target task based on the label information, and therefore cannot be directly applied to text generation tasks. How to alleviate the negative impact of text privatization on the usability of generated content is also one of our future research directions.

References

Anmol Arora and Ananya Arora. 2023. The promise of large language models in health care. *The Lancet*, 401(10377):641.

Yusuf Arslan, Kevin Allix, Lisa Veiber, Cedric Lothritz, Tegawendé F Bissyandé, Jacques Klein, and Anne Goujon. 2021. A comparison of pre-trained language models for multi-class text classification in the financial domain. In *Companion Proceedings of the Web Conference 2021*, pages 260–268.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.

Iker Ceballos, Vivek Sharma, Eduardo Mugica, Abhishek Singh, Alberto Roman, Praneeth Vepakomma, and Ramesh Raskar. 2020. Splitnn-driven vertical partitioning. *arXiv preprint arXiv:2008.04137*.

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

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235.

Alexey Dosovitskiy and Thomas Brox. 2016. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4829–4837.

Oluwaseyi Feyisetan, Borja Balle, Thomas Drake, and Tom Diethe. 2020. Privacy-and utility-preserving textual analysis via calibrated multivariate perturbations. In *Proceedings of the 13th international conference on web search and data mining*, pages 178–186.

Zecheng He, Tianwei Zhang, and Ruby B Lee. 2020. Attacking and protecting data privacy in edge–cloud collaborative inference systems. *IEEE Internet of Things Journal*, 8(12):9706–9716.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Cong Jiang and Xiaolei Yang. 2023. Legal syllogism prompting: Teaching large language models for legal judgment prediction. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law*, pages 417–421.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Yansong Li, Zhixing Tan, and Yang Liu. 2023. Privacy-preserving prompt tuning for large language model services. *arXiv preprint arXiv:2305.06212*.

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. *arXiv preprint arXiv:1907.11692*.

Lingjuan Lyu, Xuanli He, and Yitong Li. 2020. 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.

Pekka Malo, Ankur Sinha, Pekka Korhonen, Jyrki Wallenius, and Pyry Takala. 2014. Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, 65(4):782–796.

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.

Charith Peris, Christophe Dupuy, Jimit Majmudar, Rahil Parikh, Sami Smaili, Richard Zemel, and Rahul Gupta. 2023. Privacy in the time of language models. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 1291–1292.

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.

Daniele Romanini, Adam James Hall, Pavlos Papadopoulos, Tom Titcombe, Abbas Ismail, Tudor Cebere, Robert Sandmann, Robin Roehm, and Michael A Hoeh. 2021. Pyvertical: A vertical federated learning framework for multi-headed splitnn. *arXiv preprint arXiv:2104.00489*.

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

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.

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

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 the Proceedings of ICLR.

Guangxuan Xiao, Ji Lin, and Song Han. 2023. Offsite-tuning: Transfer learning without full model. *arXiv preprint arXiv:2302.04870*.

Bottom Model	Metric	Privacy Parameter η_0					
		45	50	55	60	65	70
Frozen Embedding Layer	EP	51.27	40.86	31.32	22.69	16.09	11.08
	UA	87.33	91.25	93.56	94.55	95.31	95.49
Unfrozen Embedding Layer	EP	20.52	14.31	10.86	7.64	5.19	3.05
	UA	87.83	91.46	94.30	94.71	95.59	95.78

Table 3: Empirical privacy against EIA (%) and utility accuracy (%) comparison on the SST dataset when the bottom model is a frozen or unfrozen embedding layer.

A Discussion on Trainable Bottom Model

In this appendix, we discuss another implementation of the SAP framework where the bottom model is also trainable. The premise of this setup is to make it more challenging for the vendor to infer the original input text based on the released text representations by altering the parameters of the bottom model during fine-tuning.

The specific implementation steps are as follows. During forward propagation, the customer no longer sends the perturbed representations of all samples to the vendor at once. Instead, in each iteration, the customer randomly selects a batch of samples, computes the output of the bottom model for that batch, independently privatize the representations, and then sends the privatized results to the vendor. The specific privatization process is consistent with the case where the bottom model is frozen. During backpropagation, the vendor needs to return the gradient of the input layer of top model to the customer so that it can update the bottom model.

We also evaluate the privacy protection capability and utility performance of the SAP framework when training the bottom model and top model together. When the bottom model is trainable, the customer needs to send a batch of sample representations in each iteration. If the training is conducted for E epochs, the attacker will have access to E representations for each sample, and it can independently use each representation to launch the EIA. The input tokens successfully inferred by the attacker are the union of the results of performing E attacks. The results in Table 3 indicate that when the bottom model consists only of an embedding layer, joint training of the bottom model has a minor impact on utility accuracy but results in a significant decrease in empirical privacy. The reason is that the parameters of the embedding layer change slightly after fine-tuning, and the attacker can still use the initial parameters to perform nearest neighbor search. Therefore, multiple observations of the same sample will significantly increase the attack success rate.