Privacy-Preserving Models for Legal Natural Language Processing

Ying Yin and **Ivan Habernal**
Trustworthy Human Language Technologies
Department of Computer Science, Technical University of Darmstadt
ivan.habernal@tu-darmstadt.de
www.trusthlt.org

Abstract

Pre-training large transformer models with in-domain data improves domain adaptation and helps gain performance on the domain-specific downstream tasks. However, sharing models pre-trained on potentially sensitive data is prone to adversarial privacy attacks. In this paper, we asked to which extent we can guarantee privacy of pre-training data and, at the same time, achieve better downstream performance on legal tasks without the need of additional labeled data. We extensively experiment with scalable self-supervised learning of transformer models under the formal paradigm of differential privacy and show that under specific training configurations we can improve downstream performance without sacrifying privacy protection for the in-domain data. Our main contribution is utilizing differential privacy for large-scale pre-training of transformer language models in the legal NLP domain, which, to the best of our knowledge, has not been addressed before.[1]

1 Introduction

Transformer-based models (Vaswani et al., 2017; Devlin et al., 2019) trained in a self-supervised fashion on a huge collection of freely accessible Web texts belong to the currently most successful techniques for almost any downstream NLP task across languages or domains. Their ability to 'learn' certain language properties (Rogers et al., 2020) and the need of having only a small amount of labeled data in the target domain for fine-tuning makes them superior to other approaches (Brown et al., 2020). Moreover, additional pre-training with unlabeled target-domain data typically boosts their performance further (Chalkidis et al., 2020).

However, when it comes to preserving private information contained in the original large unlabeled text data, transformer models tend 'remember' way too much. Carlini et al. (2020) show that it is possible to extract verbatim sensitive information from transformer models, such as names and addresses, even when such a piece of information had been 'seen' by the model during pre-training *only once*. Current transformer models thus represent a threat to privacy protection, which can have harmful consequences if such models trained on very sensitive data are published, as is the current trend in sharing pre-trained models.

In the legal domain, sensitive information, including names, addresses, dates of birth, are important part of many documents, such as court decisions. Especially in countries with the case-law system, court decisions make the largest fraction of legal texts. However, transformer models pre-trained on such corpora do not protect personal information by design, and ad-hoc solutions, e.g. whitening names in the original texts, are prone to errors and potential reconstruction attacks (Lison et al., 2021; Pilán et al., 2022).

Existing approaches to privacy-preserving deep learning have adapted differential privacy (DP) (Dwork and Roth, 2013), a rigorous mathematical treatment of privacy protection and loss. In particular, stochastic gradient descent with DP (DP-SGD) has been successfully applied to various NLP problems (Senge et al., 2022; Igamberdiev and Habernal, 2022), including transformer pre-training (Hoory et al., 2021; Anil et al., 2021). However, how well DP-regimes perform in the legal domain, pre-trained and fine-tuned across various downstream legal-NLP tasks, remains an open question.

This paper addresses the following three research questions. First, what are the best strategies for pre-training transformer models to be applied in the legal domain? Second, does DP-SGD training scale up to tens of gigabytes of pre-training data without ending up with an extremely big privacy budget? Finally, can large-scale privacy-

[1] https://github.com/trusthlt/privacy-legal-nlp-lm

Proceedings of the Natural Legal Language Processing Workshop 2022, pages 172 - 183

preserving transformers compete to their small-scale non-private alternatives?

2 Related work

Transformer models in legal NLP Large contextual LMs based on transformer architecture (Vaswani et al., 2017) are the state of the art in numerous NLP tasks. Domain adaptation aims to improve the model performance on downstream tasks in a specialized domain. A common approach is to pre-train BERT (Devlin et al., 2019) with a large collection of unlabeled in-domain texts. In the legal domain, Chalkidis et al. (2020) provide a systematic investigation of possible strategies for BERT adaptation and published their model as LEGAL-BERT. Their work shows that both training BERT form scratch or further pre-training the existing general BERT-BASE[2] model with legal corpora achieve comparable performance gains. Besides, broader hyper-parameter search has large impact on the downstream performance. Zheng et al. (2021) point out that despite the uniqueness of legal language, domain pre-training in the legal field rarely show significant performance gains probably due to the lack of appropriate benchmarks that are difficult enough to benefit from pre-training on law corpora. To address this issue, they release a new benchmark called CaseHOLD that gains up to 6.7% improvement on macro F1 by additional domain pre-training. In the legal field, the vast majority of benchmarks exhibit small performance gains after further pre-training BERT on law datasets (Elwany et al., 2019; Chalkidis et al., 2020). However, existing research on legal language models has not considered privacy of the textual datasets.

Privacy-preserving NLP with differential privacy Large machine learning models including transformer-based LMs can be prone to privacy attacks such as membership inference attack (Shokri et al., 2017; Hayes et al., 2019; Carlini et al., 2020), which means it is possible to predict whether or not a data record exists in the model's training dataset given only black-box query access to the model. It hinders the application of such models on numerous real-word tasks involving private user information. To mitigate this limitation, many recent studies devote to privacy-preserving algorithm for large NLP models.

Differential Privacy (DP) (Dwork et al., 2006; Dwork and Roth, 2013) has been taken as the gold-standard approach to ensure privacy for sensitive dataset. The main goal of privacy-preserving data analysis is to enable meaningful statistical analysis about the database while preventing leakage of individual information. The intuition behind DP is that an individual's data can't be revealed by a statistical release of the database regardless of whether or not the individual is present in the database, thus any individual shouldn't have significant influence on the statistical release. We formally introduce DP in Section 3.

Unlike works focusing of privatization of individual texts (Habernal, 2021, 2022; Igamberdiev et al., 2022), applying DP to training neural networks is typically done through differentially-private stochastic gradient descent (DP-SGD) (Abadi et al., 2016); see also (Yu et al., 2019) for a great explanation. Although DP pre-training of BERT has been shown to gain performance on a Medical Entity Extraction task (Hoory et al., 2021), how well it performs in the legal area still remains an open question.

2.1 Off-the-shelf strategies for training with differential privacy

DP-SGD training often suffers from big running time overhead that comes from the per-sample gradient clipping. Mainstream DL frameworks such as PyTorch and TensorFlow are designed to produce the reduced gradients over a batch that is sufficient for SGD but are unable to compute the per-sample gradients efficiently. A naive way to achieve this is to compute and clip the gradient of each sample in the batch one by one through a for-loop, which is implemented in PyVacy.[3] This approach completely loses parallelism and hence dramatically slows down the training speed. A more advanced method is to derive the per-sample gradient formula and compute it in a vectorized form. Opacus[4] implements this by replacing the matrix multiplication between the back-propagated gradients and the activations from the previous layer in the original PyTorch back-propagation with outer products via einsum function (Yousefpour et al., 2021). The activations and back-propagated gradients are captured through forward and backward hooks. A disadvantage of this method is that it cannot cur-

[2] BERT-BASE stands for 'bert-base-uncased' from https://huggingface.co/bert-base-uncased.

[3] https://github.com/ChrisWaites/pyvacy

[4] https://github.com/pytorch/opacus

rently support all kinds of neural network modules. In addition, it is restricted by quadratic memory consumption (Subramani et al., 2021).

3 Learning with differential privacy

This section formally introduces differential privacy and can be skipped by readers familiar with that topic.

3.1 Pure Differential Privacy (ε-DP)

Definition of ε-DP Given a real number $\varepsilon > 0$, a randomized mechanism (or algorithm) $\mathcal{M} : D \mapsto R$ satisfies ε-DP if for any two neighboring input datasets $d, d' \in D$ that differs in a single element and for any subset of outputs $S \subseteq R$ it holds that

$$\frac{\Pr[\mathcal{M}(d) \in S]}{\Pr[\mathcal{M}(d') \in S]} \leq \exp(\varepsilon), \tag{1}$$

where Pr stands for the probability distribution taken from the randomness of the mechanism, and ε refers to the privacy budget.

The value of ε upper-bounds the amount of influence any individual data has on the mechanism's outputs. Smaller ε value means stronger privacy guarantee. However, there is no conclusive answer to how small we should set ε to prevent information leakage in practice. The general consensus is that $\varepsilon \leq 1$ would indicate strong privacy protection, while $\varepsilon \geq 10$ possibly doesn't guarantee much privacy, although the value is application-specific.[5]

The above definition implies that the outputs of the mechanism should not differ much, with or without any specific data record. In this case, an adversary can't infer whether or not a record exists in the input dataset from the outputs of the mechanism, which prevents the extraction of individual training data from a pre-trained model.

The *sensitivity* of a mechanism $\mathcal{M}$ is the upper bound of the amount of output difference when it's input changes by one entry. Formally, the Global Sensitivity (GS) of $\mathcal{M}$ is given by

$$GS(\mathcal{M}) = \max_{d,d':|d|=|d'|\pm 1} |\mathcal{M}(d) - \mathcal{M}(d')|, \tag{2}$$

where d and d' are neighboring datasets. The "global" means this holds for any pair of neighboring datasets, as opposed to the "local" sensitivity with one of the datasets fixed. For example, sensitivity of the counting query that computes how many entries in a database is 1.

[5] https://programming-dp.com/

There are two important properties of DP: Sequential composition and post-processing.

- **Sequential Composition** For mechanisms $\mathcal{M}_1(d)$ satisfies ε_1-DP and $\mathcal{M}_2(d)$ satisfies ε_2-DP, the mechanism $\mathcal{M}(d) = (\mathcal{M}_1(d), \mathcal{M}_2(d))$ that releases both results satisfies $(\varepsilon_1 + \varepsilon_2)$-DP.
- **Post Processing** If a mechanism $\mathcal{M}(D)$ satisfies ε-DP, then after performing arbitrary function f on $\mathcal{M}(D)$, the mechanism $f(\mathcal{M}(D))$ still satisfies ε-DP.

These properties facilitate the design and analysis of a DP algorithm. The composability enables the track of privacy loss for algorithms that traverse the dataset multiple times, and the post processing property ensures that a DP algorithm is robust to privacy attack with auxiliary information. Moreover, advanced composition exists for approximate DP that provides tighter upper bound of privacy.

3.2 Appropriate Differential Privacy ((ε, δ)-DP)

Definition of (ε, δ)-DP Approximate DP relaxes the pure ε-DP requirement by introducing a "failure probability" δ. Similar to the definition of ε-DP, given real numbers $\varepsilon > 0$ and $\delta > 0$, we say a mechanism $\mathcal{M} : D \to R$ satisfies (ε, δ)-DP if for all adjacent inputs $d, d' \in D$ and all $S \subseteq R$, we have

$$\Pr[\mathcal{M}(d) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(d') \in S] + \delta \tag{3}$$

The pure ε-DP is equivalent to $(\varepsilon, 0)$-DP. A non-zero item δ allows the mechanism fails to be ε-DP with probability δ. This sounds a bit scary, since under certain probability we get no guarantee of privacy at all and there is a risk of compromising the whole dataset. Therefore, the value of δ must be small enough, preferably less than one over the size of dataset (i.e. $\frac{1}{|D|}$) in order to deliver meaningful results. One of the biggest advantage of (ε, δ)-DP is that even with negligible δ, it can significantly reduce the sample complexity compared to the pure DP (Beimel et al., 2013; Steinke and Ullman, 2015; Bun et al., 2014). Roughly speaking, given the same size of dataset, (ε, δ)-DP can achieve higher statistical accuracy than ε-DP while preserving the privacy. Additionally, the (ε, δ)-DP mechanisms in practice usually don't fail catastrophically and release the whole dataset. Instead, they fail gracefully and still satisfy $c\varepsilon$-DP for some value c in

the case of failure probability. For these reasons, approximate DP becomes popular in real applications.

The Gaussian Mechanism A Gaussian mechanism that satisfies (ε, δ)-DP can be obtained by injecting Gaussian noise as follows

$$\mathcal{M}_G(x, f, \varepsilon, \delta) = f(x) + \mathcal{N}(0, \frac{2S^2 \ln(\frac{1.25}{\delta})}{\varepsilon^2}). \quad (4)$$

3.3 Deep Learning with DP

In general, the goal of deep learning is to optimize the model parameters so that the output of the loss function is minimized. This optimization is usually achieved by Gradient Descent and its variants. Basically, the model are learned from the gradient of the loss outputs w.r.t. the model parameters. Take the mini-batch Stochastic Gradient Descent (SGD) as example, at each step t, a certain number of randomly selected training samples $\{\boldsymbol{x}_i \mid i \in \boldsymbol{B}_t, \boldsymbol{B}_t \subseteq \{1, ..., N\}\}$[6] are fed into the loss function $\mathcal{L}$ and the average of their output gradients are calculated as an estimate of the loss gradient w.r.t the model weights $\boldsymbol{\theta}$, which is then multiplied by the learning rate η for Gradient Descent. This can be formulated as follows: A DP algorithm has certain guarantee that it doesn't leak individual training examples. In the Gradient Descent algorithm, the only access to the training examples is occurred in the computation of the gradient. Therefore, one way to achieve DP is through introducing noise into the gradient before the update of model weights. If the access to the gradient calculated via training data remains DP, then the resulting model is DP according to the post-processing property. Based on this, Abadi et al. (2016) propose a sophisticated method that turns the mini-batch SGD algorithm into DP, named DP-SGD, which has become a dominant approach to privacy-preserving deep learning.

DP-SGD primarily modifies two places of the original SGD algorithm to ensure DP. One is to clip the per-example gradients so that the Euclidean-norm (L2-norm) of each individual gradient does not exceed a pre-defined upper bound C, which corresponds to a constraint for the sensitivity of gradient. The other one is to add scale-specific Gaussian noise $\mathcal{N}$ into the aggregated clipping gradient:

$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \frac{\eta}{|\boldsymbol{B}_t|} \Bigg(\sum_{\forall i \in \boldsymbol{B}_t} \text{clip}(\nabla_{\boldsymbol{\theta}_{t-1}} \mathcal{L}(\boldsymbol{\theta}_{t-1}, \boldsymbol{x}_i), C) + \mathcal{N}(0, \sigma^2 C^2 \boldsymbol{I}) \Bigg), \quad (5)$$

where σ refers to the a constant called "noise multiplier", higher σ produce stronger privacy guarantee. According to the definition of 4, the modified SGD is a Gaussian mechanism that satisfies (ε, δ)-DP. The choice of Gaussian noise is due to the high-dimensionality of the gradient. L2-norm can be applied to measure the sensitivity of a high-dimensional vector-valued function for Gaussian mechanism, which yields much lower sensitivity than Laplace mechanism that only allows the use of L1-Norm, thus much less noise needs to be added to the gradient. Moreover, Abadi et al. (2016) introduce the Moments Accountant for tighter estimation of the privacy cost. Despite its simplicity, DP-SGD brings successes in many deep learning fields.

4 Experimental setup and data

Our experiments aim to find a strategy where BERT can benefit from additional domain-specific DP pre-training. Moreover, we explore the trade-off between the privacy budget and model utility under the best setup we obtain.

Privacy-protecting scenario In our scenario, we assume that we publish a pre-trained or fine-tuned model, to which an adversary has a full access (Yu et al., 2019). The model can be pre-trained on (a) a public general dataset and (b) in-domain, potentially sensitive legal documents, and fine-tuned on (c) a public down-stream task. Our aim is to protect (b) from the adversary.

4.1 Pre-training BERT from scratch

BERT pre-training is a very expensive task, especially with DP. While further pre-training the existing BERT-BASE can take advantage of the already learned language features and greatly reduce the convergence time, the original generic vocabulary remains unchanged. A generic vocabulary might not match the specialized legal terminology and could lead to drastic splitting into sub-word units and reducing semantic expressiveness (Zheng et al., 2021; Habernal et al., 2022). To address this problem, pre-train BERT from scratch with a custom

[6] N is the total number of training examples.

legal tokenizer built on the legal corpus using the WordPiece algorithm (Wu et al., 2016).

In order to investigate the effect of domain vocabulary on model performance and also follow the setup in Hoory et al. (2021) that successfully introduce DP to the pre-training of medical BERT our pre-training from scratch can be roughly divided into three steps:

1. Generating a domain-specific tokenizer and vocabulary set based on the legal corpus.[7]
2. Pre-training BERT from scratch on the generic BookCorpus and Wikipedia dataset using the domain-specific tokenizer.
3. Further pre-training BERT with DP on the legal corpus.

In spite of that the first step also involves access to the legal corpus and may cause information leakage, there is no good solution to convert the WordPiece algorithm into DP with tight privacy bound. We leave this problem to future work. Currently, we only ensure privacy during the pre-training on the legal corpus. The second step only uses the general corpora and thereby has no privacy issue. We don't use the legal corpora at the beginning of the pre-training because the overhead to train DP BERT from scratch is too expensive. We call the model trained with the first two steps BERT-SC.

4.2 Further pre-training BERT-BASE

Continuing the pre-training of BERT-BASE with legal-domain corpora is an economical and effective way for domain transfer. We start with a small-scale pre-experimental corpus to quickly investigate the effectiveness of additional domain pre-training with different hyper-parameter settings. Afterward, we scale up the training on the full legal corpus and focus on the batch size and learning rate tuning. In order to avoid overfitting, 5% of the pre-training data is kept as a validation set, on which the sum loss of the MLM (masked language modeling) and NSP (next sentence prediction) objectives and their accuracy is evaluated at each checkpoint.

5 Downstream tasks and datasets

We experiment with two downstream benchmark datasets, Overrruling and CaseHOLD, on which we fine-tune our pre-trained models.[8] Note that for the downstream tasks, we do not use differential private training.

The Overruling dataset (Zheng et al., 2021) corresponds to a binary classification task that predicts if a sentence has the meaning of voiding a legal decision made in a previous case, which is important to ensure the correctness and validity of legal agreements. The sentences in the dataset are sampled from the Casetext law corpus, where positive overruling examples are manually annotated by lawyers, and negative examples are automatically generated by randomly sampling the Casetext sentences because over 99% of them are non-overruling. The complete dataset contains 2400 items and the two classes are balanced. It is a relatively simple task that has already achieved state-of-the-art performance on BERT-BASE model, since the positive examples explicitly contain 'overrule' or words with similar meaning such as disapprove, decline, reject, etc., which makes them highly distinguishable from the negative ones.

The CaseHOLD (Case Holdings on Legal Decisions) is a multiple-choice QA task to select a correct holding statement among 5 potential answers that matches the given citing context from a judicial decision. Zheng et al. (2021) construct the dataset by extracting the legal citations and the accompanying holding statements from the corpus of U.S. CaseLaw and using them as questions and answers respectively. Here the cases contained in the CaseHOlD are removed from our legal pre-training corpus according to the case IDs they provide. Moreover, they search for propositions that are semantically similar to the corresponding answer from other extracted holding statements as the wrong answers according to the TF-IDF similarity between them, which makes the CaseHOLD a multiple-choice QA task. The labels of the correct answers are uniformly distributed within the 5 indices 0-4. Excluding some samples containing invalid labels, the full dataset we use has a total of 52,978 items. It is a challenging task and yields only a macro F1 of around 0.613 using the general BERT-BASE (Zheng et al., 2021). We use it to investigate whether a sufficiently difficult legal task benefits from additional domain pre-training in the private preserving scenario.

[7] Here we use `BertWordPieceTokenizer` from https://github.com/huggingface/tokenizers, we set the vocabulary size to 30,522 which is the same as with BERT-BASE.

[8] Hyperparameters for the downstream tasks are discussed in Appendix A.

6 Our approach to pre-training legal transformer models with DP

6.1 Datasets for pre-training

For the in-domain pre-training with differential privacy, we prepare a legal corpus consisting of 14GB legal texts that are collected from three different resources (see Table 1). Although these are public datasets, we treat them as if they were private, containing sensitive data whose leakage from the pre-trained models should be prevented. For compiling and caching the large-scale pre-training corpora, we leverage the HuggingFace Datasets library[9] based on Arrow, which allows fast lookup for big datasets by building a memory-mapped cache on disk.

Source	Documents	Size (GB)
Sigma Law[10]	39,155	1.2
LEDGAR[11]	≈ 300,000	0.2
Case Law[12]	≈ 28,300,000	12.6

Table 1: Details of the legal corpora for pre-training.

6.2 Scalable pre-training with DP

In section 2.1 we discussed the shortcomings of off-the-shelf DP-SGD implementations in mainstream frameworks. We carried out preliminary experiments and found that these shortcomings make DP-SGD pre-training infeasible due to 12 to 28-times longer runtime per epoch.

The training speed of DP-SGD can be significantly improved by vectorization, just-in-time (JIT) compilation and static graph optimization using JAX framework,[13] which is defined by JIT compilation and automatic differentiation built up on the XLA compiler (Subramani et al., 2021). The core transformation methods of main interest in JAX includes `grad`, `vmap`, `jit`, and it allows us to arbitrarily compose these operations. In the DP-SGD scenario, `grad` can automatically compute the gradients of the loss objective w.r.t. the model parameters, and combing `vmap` enables efficient computation of per-example gradients by vectorizing the gradient calculation along the batch dimension.

Furthermore, the DP-SGD step can be decorated by `jit` to leverage XLA compiler that has proven acceleration in BERT MLPerf submission. Although JAX shows great advantages over other mainstream DP frameworks and libraries on a wide variety of networks such as Convolutional Neural Network (CNN) and Long-Short Term Memory network (LSTM) in Subramani et al. (2021), how much speedup it can produce on large transformer-based LMs remains unknown.

To investigate this, we implementat a JAX version of DP BERT based on FlaxBert models,[14] which provides transformers with JAX/FLAX backend including BERT. We adapt its training step into DP by adding the per-sample gradients clipping before aggregation and introducing randomly sampled Gaussian noise to the reduced gradients. Moreover, we use the strategy of gradient accumulation to enable DP pre-training with arbitrarily large batch sizes. Specifically, a training step is split into many iterations such that each iteration handles a shard of examples that the GPU[15] memory can maximally hold, and the clipped per-example gradients are accumulated over iterations within a batch.

6.2.1 Finding optimal hyper-parameters

Our starting point to further pre-training with differential privacy is the uncased BERT-BASE model that contains 110M parameters. For the optimization, we use Adam with weight decay (AdamW, (Loshchilov and Hutter, 2019)) and a linear learning rate schedule, which consists of a warm-up phrase followed by a linear decay. The warm-up steps are set to roughly 5% of the total training steps with a lower bound of 25. In addition, we use the TensorFlow privacy library[16] based on Rényi DP (RDP) (Mironov, 2017; Mironov et al., 2019) for the track of privacy, which can be converted to a standard (ε, δ)-DP but provides a tighter composition for Gaussian mechanism than directly using (ε, δ)-DP. The method takes the noise multiplier σ as input and calculates the privacy budget ε for each step. Conversely, we obtain the desired ε by the binary search for an optimal noise multiplier that leads to a privacy budget close enough to the target ε in a proper range.

[9] https://huggingface.co/docs/datasets/
[10] https://osf.io/qvg8s/
[11] Tuggener et al. (2020)
[12] https://case.law
[13] https://github.com/google/jax
[14] https://huggingface.co/docs/transformers/model_doc/bert
[15] All the experiments are carried out on an NVIDIA A100 40GB.
[16] https://github.com/tensorflow/privacy

Model	Overruling	CaseHOLD
BERT-BASE	0.971	0.617
BERT-SC	0.975	0.618

Table 2: Baseline Macro-F_1 scores without any domain pre-training.

σ	ε	Overruling	CaseHOLD
BERT-BASE			
–	–	0.975	0.652
1e-5	∞	0.967	0.648
0.1	4e+5	0.971	0.616
0.5	3.726	0.969	0.613
BERT-SC			
–	–	0.969	0.647
1e-5	∞	0.967	0.645
0.1	4e+5	0.967	0.618
0.5	3.726	0.964	0.616

Table 3: Macro-F_1 scores for further small-scale pre-training of BERT-BASE and BERT-SC. σ="–" corresponds to the training without DP.

In our experiments, the gradient clipping norm and the weight decay are less significant factors, and we fix them to 1.0 and 0.5 accordingly. To study the influence of the batch size, we keep the privacy ε to 5, which is considered as a sweet point between a very strong privacy guarantee 1 and a weak guarantee 10. In order to avoid overfitting, 5% of the pre-training data is kept as a validation set, on which the sum loss of the MLM and NSP objectives and their accuracy is evaluated at each checkpoint.

7 Results and analysis

Baselines Our baseline results (Table 2) are reported from BERT-BASE and BERT-SC with tuned hyper-parameters with no privacy gurantees. BERT trained from scratch with a custom legal vocabulary (BERT-SC) slightly outperforms vanilla BERT-BASE.

Small-scale pre-training with DP We experimented with further pre-training of two baseline models on a small-scale 2.3GB legal sub-corpus. The goal was to efficiently explore the effect of several key hyper-parameters on DP training with a small amount of data. We trained for 29k steps at batch size 256.

Table 3 shows that while both baseline models after further pre-training without privacy achieve ~ 3% substantial performance gains on CaseHOLD, the results of DP pre-training is disappointing. The benefits of domain training for CaseHOLD task seem to disappear after adding even a small amount of noise ($\sigma = 0.1$). The results from $\sigma = 0.1$ and $\sigma = 0.5$ don't outperform the baseline or are even marginally worse than it. In addition, the legal tokenizer doesn't indicate an advantage over the general one. We conclude that small-scale DP pre-training barely brings any improvement and even hurts the performance. We decide to scale up the training and explore larger batch sizes.

7.1 Large-scale domain pre-training with DP

As the batch size is one of the most important parameter in DP training, we fix the target privacy budget ε as 5 and further pre-train BERT-BASE on the large-scale full legal corpus starting with the default parameters (see Table 4 in the Appendix). Then we explore the parameter space by gradually increasing the batch size up to ~ 1M and roughly tune the learning rate at the same time. Although we have significantly accelerated the DP training by JAX framework, large-scale DP pre-training is still quite expensive. Due to resource and time constraints, we do not perform a complete grid search but only experiment with the likely best learning rates at each batch size in our experience.

Gradient-SNR Following the work in Anil et al. (2021), we keep track of the gradient signal-to-noise ratio at each step during the DP pre-training of BERT. Figure 1 shows the impact of batch size and learning rate on the Gradient-SNR. In general, the SNR decreases with training and eventually converges to a small value. This is probably due to the fact that the magnitude of the gradient decreases constantly during the learning, whereas the magnitude of the noise remains basically the same, so the ratio of the two keeps shrinking until the gradients become stable. From the left subplot 1(a) we can see that a larger batch size leads to higher Gradient-SNRs. Moreover, the right subfigure (b) shows that an appropriate learning rate can also improve the Gradient-SNR for a certain batch size. However, a too large learning rate leads to dramatic oscillation of Gradient SNR, and the model may move away from the local optima and thus increase the training loss.

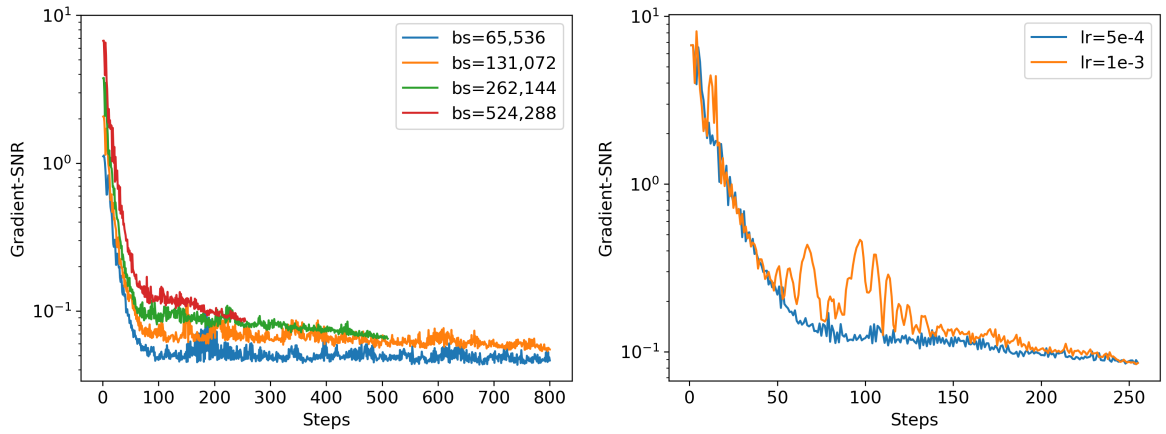

Figure 1: Gradient-SNR over steps for DP pre-training with same privacy budget and fixed epochs while varying the batch size (bs) or learning rate (lr). The left plot shows the trends of SNR at four different batch sizes. For smaller batch sizes, the SNRs after 800 steps are not presented, but they've basically converged to a small value as seen in the figure. Their initial learning rates are uniformly set as 5e-4. The right-hand figure draws the changes on SNR at batch size 524,288 using two different learning rates.

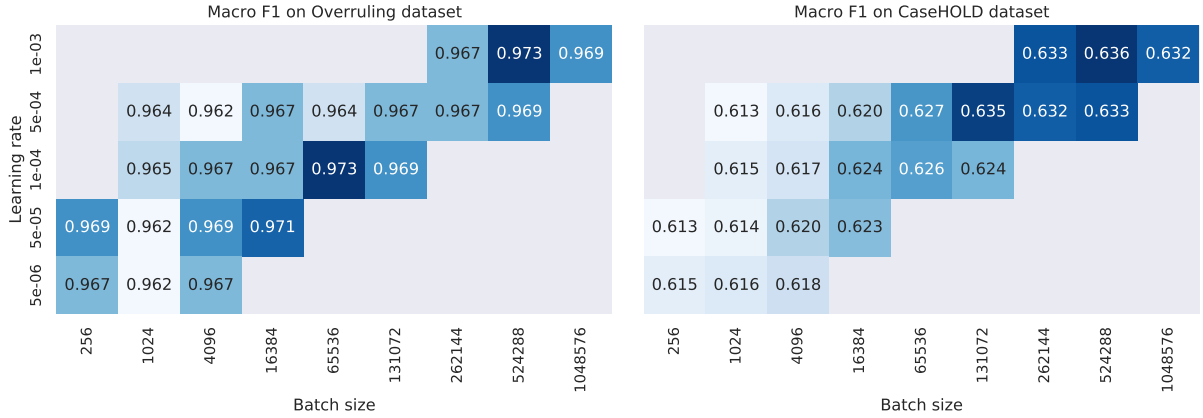

Figure 2: Downstream results obtained by tuning the batch size and learning rate of large-scale domain pre-training when fixing both the target privacy ε and training epochs to 5.

Results on downstream tasks In our experiments with fixed training epochs, the batch size and learning rate jointly influence the performance on CaseHOLD. As can be seen from the bottom subplot of Figure 2, training with unusually large batch sizes and high learning rates (upper right area) produces significantly better Macro F1 scores than using small batch sizes and low learning rates (bottom left area). By scaling up the batch size and tuning the learning rate accordingly, we achieve the best Macro F1 of 0.636 at batch size 524,288 and learning rate 1e-3. This **outperforms the baseline by almost 2%**.

As a summary, for a fixed training epoch setting, enlarging the batch size is not always beneficial and tuning the learning rate is crucial as well. However, according to our experiments, DP pre-training of BERT with a regular small batch size performs overall very poorly, and it starts to make performance gains on CaseHOLD when the batch size is stepped up to 4,096. We obtain a significant boost when increasing the batch size to 130K+. We conclude that scaling up the batch size and in-domain corpus is necessary to obtain good performance for DP pre-training of BERT in the legal field.

8 Discussion

Here we clarify some questions and comments raised by the reviewers.

Is the 2% improvement worth the effort? We believe so. Let's put our result into a broader context by having a closer look at results achieved by LEGAL-BERT (Chalkidis et al., 2020). On three downstream tasks, they gained similar improve-

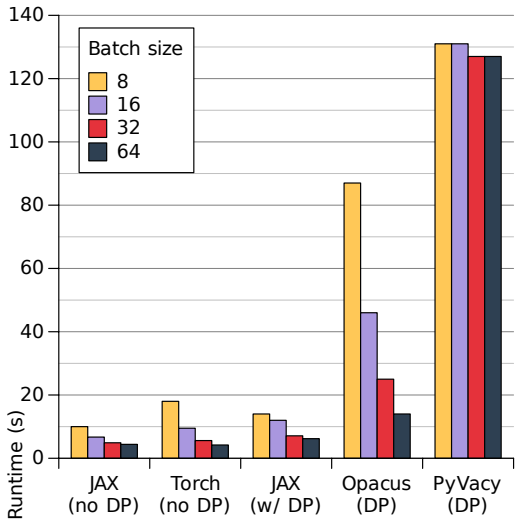

Figure 3: Runtimes (in seconds) per epoch for fine-tuning BERT on the Overruling binary classification task with different batch sizes and frameworks.

ments. First, *"in ECHR-CASES, we [...] observe small differences [...] in the performance on the binary classification task (0.8% improvement)."* Second, on NER they observed an *"increase in F1 on the contract header (1.8%) and dispute resolution (1.6%) subsets. In the lease details subset, we also observe an improvement (1.1%)."* Finally, on EURLEX57k, they observed *"a more substantial improvement in the more difficult multi-label task (2.5%) indicating that the LEGAL-BERT variations benefit from in-domain knowledge."* Moreover, our approach achieves similar gains under differential privacy guarantees.

How expensive is DP training? We experimentally evaluate the running performance of different frameworks on a binary classification task (Overruling) in both private and non-private cases. Figure 3 show the runtimes per epoch taken from the median over 20 epochs of training. In our experiments, Opacus is unable to support BERT's Embedding layer, although we use its official tutorial for training. This also prevents us to use it for the DP pre-training. We freeze its Embedding layer for the fine-tuning, which reduces nearly 22% training parameters compared to other methods. By doubling the batch size each time, 64 is the maximum batch size that the current GPU can support for JAX framework. Opacus uses a `BatchMemoryManager` to eliminate the limit of batch size similar to gradient accumulation, but the physical batch size it can achieve is actually much smaller than 64. This indicates that JAX has higher memory-efficiency than Opacus. The runtime of all the methods decreases significantly as the batch size grows except for PyVacy. In summary, due to the performance of JAX in the DP training, the 'extra costs' are negligible and allows us to upscale DP pre-training.

The title is misleading, the authors do not propose a privacy-preserving legal NLP model. If we take the definition of privacy through the lenses of differential privacy, then our pre-trained model is privacy-preserving; see, e.g., Yu et al. (2019) for a terminology clarification, or parallel works with the T5 language model (Ponomareva et al., 2022).

Why even do this? The scenario in which we want to protect privacy is the following. Say a company has huge amounts of in-house sensitive legal texts (e.g., contracts) which are valuable for pre-training a LM. This model is likely to be better performing on similar domains, so the company wants to offer an API or provide the model to other parties for further fine-tuning. Without DP, privacy of the pre-training data can be compromised (Pan et al., 2020; Carlini et al., 2020; Yu et al., 2019).

9 Conclusion

This paper shows that we can combine large-scale in-domain pretraining for a better downstream performance while protecting privacy of the entire pre-training corpus using formal guarantees of differential privacy. In particular, we implemented highly-scalable training of the BERT model with differentially-private stochastic gradient descent and pre-trained the model on $\approx$ 13 GB legal texts, using a decent $\varepsilon = 5$ privacy budget. The downstream results on the CaseHOLD benchmark show up to 2% improvements over baseline models with tuned hyper-parameters and models trained from scratch with a custom legal vocabulary. Our main contribution is utilizing differentially-private large-scale pre-training in the legal NLP domain. We believe that adapting formal privacy guarantees for training models might help overcome the difficulties of using large but potentially sensitive datasets in the legal domain.

Acknowledgements

The independent research group TrustHLT is supported by the Hessian Ministry of Higher Education, Research, Science and the Arts. We thank the anonymous reviewers for their useful feedback.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318, Vienna, Austria. ACM.

Rohan Anil, Badih Ghazi, Vineet Gupta, Ravi Kumar, and Pasin Manurangsi. 2021. Large-Scale Differentially Private BERT. *arXiv preprint*, pages 1–12.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. In *NIPS 2016 Deep Learning Symposium*.

Amos Beimel, Kobbi Nissim, and Uri Stemmer. 2013. Private Learning and Sanitization: Pure vs. Approximate Differential Privacy. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 363–378, Berlin, Heidelberg. Springer Berlin Heidelberg.

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. *arXiv preprint*.

Mark Bun, Jonathan Ullman, and Salil Vadhan. 2014. Fingerprinting codes and the price of approximate differential privacy. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, page 1–10, New York, New York. ACM.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. 2020. Extracting Training Data from Large Language Models. *arXiv preprint*.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The Muppets straight out of Law School. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer.

Cynthia Dwork and Aaron Roth. 2013. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3-4):211–407.

Emad Elwany, Dave Moore, and Gaurav Oberoi. 2019. Bert goes to law school: Quantifying the competitive advantage of access to large legal corpora in contract understanding. In *Document Intelligence 2019 Workshop at NeurIPS*.

Ivan Habernal. 2021. When differential privacy meets NLP: The devil is in the detail. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1522–1528, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ivan Habernal. 2022. How reparametrization trick broke differentially-private text representation learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 771–777, Dublin, Ireland. Association for Computational Linguistics.

Ivan Habernal, Daniel Faber, Nicola Recchia, Sebastian Bretthauer, Iryna Gurevych, Indra Spiecker genannt Döhmann, and Christoph Burchard. 2022. Mining Legal Arguments in Court Decisions. *arXiv preprint*.

Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. 2019. Logan: Membership inference attacks against generative models. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*, volume 2019, pages 133–152. De Gruyter.

Shlomo Hoory, Amir Feder, Avichai Tendler, Sofia Erell, Alon Peled-Cohen, Itay Laish, Hootan Nakhost, Uri Stemmer, Ayelet Benjamini, Avinatan Hassidim, and Yossi Matias. 2021. Learning and Evaluating a Differentially Private Pre-trained Language Model. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1178–1189, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Timour Igamberdiev, Thomas Arnold, and Ivan Habernal. 2022. DP-Rewrite: Towards Reproducibility and Transparency in Differentially Private Text Rewriting. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2927–2933, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Timour Igamberdiev and Ivan Habernal. 2022. Privacy-Preserving Graph Convolutional Networks for Text

Classification. In *Proceedings of the Language Resources and Evaluation Conference*, pages 338–350, Marseille, France. European Language Resources Association.

Pierre Lison, Ildikó Pilán, David Sánchez, Montserrat Batet, and Lilja Øvrelid. 2021. Anonymisation Models for Text Data: State of the Art, Challenges and Future Directions. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4188–4203, Dublin, Ireland. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Ilya Mironov. 2017. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE.

Ilya Mironov, Kunal Talwar, and Li Zhang. 2019. Rényi Differential Privacy of the Sampled Gaussian Mechanism. *arXiv preprint*.

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, San Francisco, USA. IEEE.

Ildikó Pilán, Pierre Lison, Lilja Øvrelid, Anthi Papadopoulou, David Sánchez, and Montserrat Batet. 2022. The text anonymization benchmark (TAB): A dedicated corpus and evaluation framework for text anonymization. *arXiv pre-print*.

Natalia Ponomareva, Jasmijn Bastings, and Sergei Vassilvitskii. 2022. Training Text-to-Text Transformers with Privacy Guarantees. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2182–2193, Dublin, Ireland. Association for Computational Linguistics.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Manuel Senge, Timour Igamberdier, and Ivan Habernal. 2022. One size does not fit all: Investigating strategies for differentially-private learning across NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Abu Dhabi, UAE. Association for Computational Linguistics.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE.

Thomas Steinke and Jonathan Ullman. 2015. Between pure and approximate differential privacy. *arXiv preprint arXiv:1501.06095*.

Pranav Subramani, Nicholas Vadivelu, and Gautam Kamath. 2021. Enabling fast differentially private sgd via just-in-time compilation and vectorization. *Advances in Neural Information Processing Systems*, 34.

Don Tuggener, Pius vo von Däniken, Thomas Peetz, and Mark Cieliebak. 2020. LEDGAR: A Large-Scale Multi-label Corpus for Text Classification of Legal Provisions in Contracts. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1235–1241, Online. European Language Resources Association.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems 30*, pages 5998—-6008, Long Beach, CA, USA. Curran Associates, Inc.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, et al. 2021. Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint arXiv:2109.12298*.

Lei Yu, Ling Liu, Calton Pu, Mehmet Emre Gursoy, and Stacey Truex. 2019. Differentially Private Model Publishing for Deep Learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 332–349, San Francisco, USA. IEEE.

Lucia Zheng, Neel Guha, Brandon R. Anderson, Peter Henderson, and Daniel E. Ho. 2021. When Does Pretraining Help? Assessing Self-Supervised Learning for Law and the CaseHOLD Dataset of 53,000+ Legal Holdings. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*, pages 159–168, Sao Paulo, Brazil. ACM.

A Hyperparemeters for downstream tasks

As shown by Chalkidis et al. (2020) and Zheng et al. (2021), downstream hyper-parameters have a significant impact on the evaluation results and an enriched search range is necessary for the legal benchmarks. Therefore, instead of blindly following the recommended search range given by Devlin

	Learning Rate	Batch Size	Epochs
Devlin et al. (2019)	2e-5, 3e-5, 4e-5, 5e-5	16, 32	3, 4
First round	5e-6, 1e-5, 5e-5, 1e-4	8, 16, 32, 64, 128	max 10, early stop
Second round	7e-6, 2e-5, 3e-5, 7e-5	16 Overruling; 128 CaseHOLD	max 10, early stop
Final setup	1e-5, 3e-5, 5e-5, 7e-5	16 Overruling; 128 CaseHOLD	max 5, early stop

Table 4: Summary of the hyper-parameter search

ω	FP eval loss	MLM acc	NSP acc	F1 on Overruling	F1 on CaseHOLD
0.1	1.706	0.682	0.947	0.973	0.636
0.5	1.701	0.681	0.947	0.973	0.636
1.0	1.695	0.681	0.948	0.969	0.636

Table 5: Evaluation results for tuning the weight decay ω on the best setup (bs=524,288, lr=1e-3).

et al. (2019), we perform a broader search through two rounds of coarse- to fine-grained grid search. The details of the searched hyper-parameters are shown in Table 4. In the final setup, we fix the batch size as 16 for Overruling and 128 for CaseHOLD, and train for a maximum of 5 epochs. Furthermore, the downstream performances are relatively sensitive to the learning rate, we do a search over {1e-5, 3e-5, 5e-5, 7e-5} and the best macro-f1 scores are reported for each pre-trained model.

B Additional experiments with limited impact

B.1 Weight Decay ω

BERT uses layer normalization (Ba et al., 2016) that makes the output of a layer independent of the scale of its weights. As explained in Anil et al. (2021), the Frobenius norm of the layer weights tends to grow due to the noise introduced in the DP training, which reduces the norm of the gradients and thereby slows down the learning process under the layer normalization. To address this problem, they suggest using a much larger weight decay for Adam optimizer compared to the non-private training. Therefore, we experiment with several different weight decays on the best setup of batch size and learning rate. The results are outlined in Table 5. Different from the results in Anil et al. (2021), changing the weight decay causes almost no impact on the downstream performance and accuracy of MLM and NSP. One can only observe a negligible decline in loss as the weight decay increases. This is probably because our training starts from a well pre-trained base model, the weight update is more stable than training from scratch.

B.2 L2 Clipping Norm C

Recall that the two critical steps in DP-SGD are to clip the L2 norms of per-example gradients to C and to introduce randomly sampled Gaussian noise with standard deviation σC. Both steps involve the clipping norm C, thus it is likely to be an important hyper-parameter for DP training. We experiment with different values of C in {0.01, 0.1, 1.0, 10} at batch size 1024 and σ 0.5. However, the MLM and NSP accuracy and downstream performance are almost unchanged when we drastically vary C. Hence, we consider that the L2 clipping norm may not be a key factor to DP pre-training and fix it to 1.0 in future experiments based on the common best results of two end tasks.