Training Text-to-Text Transformers with Privacy Guarantees

Natalia Ponomareva **Jasmijn Bastings** **Sergei Vassilvitskii**
Google Research
{nponomareva,bastings,sergeiv}@google.com

Abstract

Recent advances in NLP often stem from large transformer-based pre-trained models, which rapidly grow in size and use more and more training data. Such models are often released to the public so that end users can fine-tune them on a task dataset. While it is common to treat pre-training data as public, it may still contain personally identifiable information (PII), such as names, phone numbers, and copyrighted material. Recent findings show that the capacity of these models allows them to memorize parts of the training data, and suggest differentially private (DP) training as a potential mitigation. While there is recent work on DP *fine-tuning* of NLP models, the effects of DP *pre-training* are less well understood: it is not clear how downstream performance is affected by DP pre-training, and whether DP pre-training mitigates some of the memorization concerns. We focus on T5 and show that by using recent advances in JAX and XLA we can train models with DP that do not suffer a large drop in pre-training utility, nor in training speed, and can still be fine-tuned to high accuracy on downstream tasks (e.g. GLUE). Moreover, we show that T5's span corruption is a good defense against data memorization.

1 Introduction

Recent advances in natural language processing tasks are largely due to introduction of large Transformer-based models trained on large amounts of data. Models such as GPT-2 (Radford et al., 2019) and T5 (Raffel et al., 2020) have billions of parameters and are trained on hundreds of gigabytes of mostly uncurated public crawl data. These models are often released as modifiable checkpoints, and the end users have the ability to fine-tune these models to their final tasks using an often more limited amount of data and compute.

While pre-training datasets are typically treated as public, their sheer size makes them difficult to curate or scrutinize (Bender et al., 2021; Rogers, 2021). Moreover, such public datasets (e.g., web crawls) likely contain private information (Dodge et al., 2021), e.g., data erroneously released to the web or copyrighted text. The capacity of recent models makes it possible for them to memorize parts of the training data (Carlini et al., 2020), even after subsequent fine-tuning, and poses risks to the owners of pre-trained language models. In this work, we focus on a potential mitigation: making the model fully private using differential privacy (DP). We focus on T5 (Raffel et al., 2020) and explore how well DP mitigates privacy risks and how it affects pre-training and downstream performance.

Our contributions are as follows:

1. We describe how to achieve fully private T5 models by (a) introducing private SentencePiece (DP-SP) and (b) combining it with private training (DP-Training).
2. To the best of our knowledge, we are the first to look into private pre-training (as opposed to private fine-tuning) of T5, while also showing how it affects downstream tasks. More concretely, we demonstrate that fully private models are able to achieve good pre-training and fine-tuning utility. Part of the drop in utility introduced by DP-Training is mitigated by DP-SP (unigram) tokenizer.
3. We show that all private pre-training components of T5 (DP-SP and DP-Training) help reduce memorization of T5 models. The biggest reduction comes from DP-Training, while DP-SP memorization protection is much smaller.
4. We demonstrate that the pre-training objective (i.e., span corruption, next token prediction) has a significant impact on the ability to memorize training instances. In particular, if memorization is the main concern, models trained with span corruption, even without any additional privacy changes, exhibit excellent resilience to training data extraction.

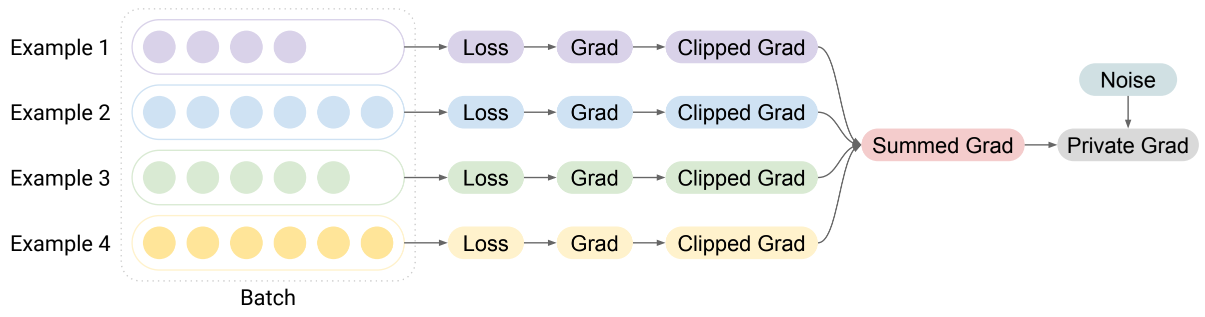

Figure 1: Differentially private training. Unlike conventional batched training, the gradient is computed and clipped for each example in the batch separately, then accumulated and noised before updating the parameters.

2 Related work

In this section, we discuss what privacy in ML means (§2.1), followed by overview of private training (§2.2) and privacy in language models (§2.3).

2.1 Privacy in ML

Privacy guarantees can come in many forms. On the one hand, for a trained ML model, one can provide **theoretical** Differential Privacy (DP) guarantees in the form of (ϵ, δ) (Dwork and Roth, 2014), that (roughly) say that with probability $1 - \delta$, no attacker can increase their prior on whether a specific example is part of the training data by more than a factor of $\exp(\epsilon)$. These can be further categorized into guarantees on all of the weights of the model (usually achieved via DP-training) or guarantees on the outputs of the model only (private prediction), which translates into training data label protection. The latter is usually achieved via adding noise to the output. Full model guarantees provide also the weaker guarantees, i.e., private training also ensures private prediction but not vice versa.

On the other hand, the term 'private' is sometimes applied to ML models to describe **empirical characteristics of the model**. For example, a model can be described as private if it is robust to membership attacks, training data extraction attacks, or to attacks that attempt to infer some private attribute (e.g., the race of a speaker) from the data. It is worth noting that DP methods like DP-training can mitigate some (membership attack, training data extraction attacks) but not all attacks in this category. And heuristic methods to make models robust to these attacks, such as adversarial heads or adversarial training data augmentation, don't provide any theoretical privacy guarantees.

In this paper, we will focus on theoretical privacy (full model protection) achieved via DP-Training. In §5.2 we verify how our models fare with respect to an empirical "privacy" definition, namely robustness to training data extraction attacks.

2.2 Differentially Private (DP) Training

DP training is a modification of the training process of ML models that guarantees that the resulting models (and all of the post processing on them) are also differentially private. DP training is usually achieved via gradient noise or perturbing the loss.

Gradient noise, which is by far the most common method, involves adding noise to the gradients like DP-SGD and its variants (Abadi et al., 2016; Pichapati et al., 2019). This is shown in Figure 1 with a batch of examples of various lengths. An alternative is to perturb the loss function and then optimize as usual (Chaudhuri et al., 2011; Phan et al., 2016). Here DP guarantees hold only when the algorithm is fully converged, e.g. a global optimum is reached, which is not guaranteed for non-convex problems, and large LMs require many steps to get there. Iyengar et al. (2019) suggested an alternative perturbation that does provide guarantees even if the model reaches only the vicinity of a global optimum, but convexity remains a requirement.

All of these methods inject noise into the training process and are known to result in a drop of utility (Appendix D discusses ways to mitigate the utility drop in DP-Train). Since Transformer-based NLP models are non convex, and are usually not trained to full convergence, we employ one of the most popular methods that work by noising the gradients (Abadi et al., 2016).

2.3 NLP models and Privacy

In NLP it is common to pre-train on large amounts of unlabeled data and then fine-tune on the final task. A lot of related work assumes that pre-training data is essentially public, and makes models private with respect to the limited fine-tuning data. For example, Kerrigan et al. (2020) pre-trained GPT-2 on public data and DP fine-tuned it on private data. They demonstrated that such pre-training on public data helps reduce perplexity. Li et al. (2021) performed a similar analysis and showed that private fine tuning can maintain accuracy given a good pretrained model. Hoory et al. (2021a) looked into DP fine-tuning of a (publicly) pre-trained BERT model (Devlin et al., 2019) in the medical domain and explored how DP fine tuning affects the performance and privacy of the models. They point out that multiple components for language models may need to be adjusted to incorporate privacy. For BERT-like models, the tokenization algorithm (WordPiece) can be trained on the private data (to improve the utility), and thus needs to be adjusted to preserve the privacy. Secret sharer (Carlini et al., 2018) was used for evaluation, and the authors demonstrate that with adjustments (e.g., larger batch size) for private models, utility is hurt only marginally while being more robust to leaking "secrets", even those with high frequency.

At the same time, several works show that publicly (e.g. not using DP-Training methods) pre-trained NLP models are vulnerable to privacy attacks (even after subsequent fine tuning). (Thomas et al., 2020) looked into whether pre-trained BERT, Glove and ELMO embeddings contained private data. The authors inserted secret information into the embeddings' training data, and then explored LSTM models subsequently fine-tuned on these embeddings. They showed that higher dimensional embeddings leak more information than lower dimensional ones, and DP training reduced this leakage. Additionally, for all but Glove embeddings, the presence of multiple secret values with the same pattern (e.g. multiple sentences of "John is sick with flu" and "Mary is sick with cold") reduced the leakage. Leakage is also correlated with the number of epochs used to pre-train the embeddings. DP training (the authors used (ϵ, δ) of $(10, 0.00002)$ with a noise level of 0.44) did reduce the "exposure", sometimes up to 7 fold. However, it is worth noticing that the exposure metric is calculated by looking at what log perplexity the model assigns to the secret word that was present during training in comparison to the scores that the model assigns to other secret words (from a limited secret word vocabulary). This also means that a sequence-to-sequence model is not guaranteed to never output a secret word, even if it was trained privately. Instead, it means that the probability of outputting such words is greatly reduced (and the scores with which they are output are also lower). Additionally, for sequence generation, it is common to use Beam search, which takes not just the top prediction but top k predictions into consideration, so it is still possible to leak secret pre-training data.

Taking this further, Carlini et al. (2020) demonstrated that it is possible to extract some training data instances by prompting the pre-trained GPT-2 (Radford et al., 2019) with enough context: first the model was used to generate text sequences by sampling from the model repeatedly word by word, and then perplexity scores for generated sequences were used to decide whether the generated data was actually present in the training data. Finally, the authors hypothesized (but didn't verify empirically) that DP-training might help mitigate this training data attack, but highlight that it usually does hurt the utility. They also mention that curating the training data could be helpful but is hard to do, especially for large pre-training datasets. Additionally, fine-tuning on the downstream task could potentially remove some of memorized information.

Finally, Lee et al. (2021) demonstrated that due to non-uniqueness of training data, language models may output training data instances verbatim, which obviously is a privacy concern. They proposed to mitigate this by deduplicating the pre-training data and showed that it resulted in substantial decrease in verbatim training data generation.

2.4 Summary

To summarize, prior work showed that publicly pre-training LLMs results in privacy vulnerabilities (e.g., memorization of the training data) that is exacerbated for larger models, and it was hypothesized that DP training can mitigate these risks. However, most of the works treat pre-training data as public and do DP fine-tuning only. Further, it is not known whether DP pre-trained models can perform well on downstream tasks after (public) fine tuning. In subsequent sections, we look to privately (DP) pretrain LLMs and investigate how their pre-training and subsequent fine-tuning performance is affected, as well as verify whether DP pretraining can mitigate some privacy risks outlined above.

3 Implementing a Fully Private T5

We focus on T5 (Raffel et al., 2020), a popular encoder-decoder. It uses a slightly modified Transformer architecture (Vaswani et al., 2017) and both the input and output is a sequence of tokens, as tokenized by SentencePiece (Kudo and Richardson, 2018). T5 is a good model to focus on, since it can be used for many input-output tasks, is trained on a large public crawl data set, is publicly available, and has been shown to have excellent performance on subsequent fine-tuning tasks.

What we are protecting. We use the DP definition, so we provide protection at the level of a training instance. For encoder-decoders like T5, that means a pair of input and output sequences. Importantly, if the same training example is repeated multiple times in the training data, the level of protection for such an example will be smaller.

Modifications. There are **two parts** to training T5 that need to be modified to achieve a fully private model. The first part is the tokenizer, which is trained on the training data. This part is often overlooked by papers claiming to train private NLP models. It is also unique to NLP models (e.g., in comparison to image models). In §5.1 we show that making the tokenizer private is very important and allows us to reduce the utility drop introduced by DP-training. The second part is the modification of the optimization algorithm (DP-Training).

3.1 Private tokenizer (DP-SentencePiece)

SentencePiece (Kudo and Richardson, 2018) is a tokenizer commonly used for pre-processing text data. It comes with a number of algorithms that can be used (e.g., unigram, char, BPE). One of the first papers that looked into making tokenizers private is Hoory et al. (2021a), who devised an algorithm that adds Laplacian noise to the histogram of the word counts and applied it to the WordPiece algorithm used by BERT. Hoory et al. (2021a) improved on these bounds by using Gaussian noise.

Algorithm 1 is a slight modification of their algorithm. For each sentence in the data, compute the histogram of words and counts. Then, compute the histogram of the overall dataset by adding the word counts across all histograms. Contrary to Hoory et al. (2021a), we do not limit the count of a word in a sentence to 1, to give per-example (as opposed to per word) DP guarantees. The words in the original histogram are not modified or normalized; it may contain words such as “Chrysler&apos;s”. The rest of SentencePiece algorithm is unmodified.

Algorithm 1: DP-SentencePiece histogram

```
Input  : A histogram $h = \{w_i : c_i\}$ with
         $w_i$ a word type and $c_i$ the total
         count of $w_i$ in the data.
         $\sigma, C$ - noise and clipping threshold
Output : Private histogram
1 for $i \leftarrow 0$ to $size(h)$ do
2     $count'_i = h[w_i] + \mathcal{N}(0, \sigma^2)$
3     if $count'_i >= C$ then
4         $h'[w_i] = count'_i$ ;
5 end for
6 return $h'$
```

To calculate the bounds, we use Theorem 1 from Hoory et al. (2021b): Given N the number of words in a sentence, k the maximum L2 norm of a sentence-level histogram, m the maximum infinity norm of sentence-level histogram, and σ the noise level added to the counts, we would obtain (ϵ, δ) DP guarantees with $\epsilon = \frac{k}{\sigma}\sqrt{2\log(2.5/\delta)}$ when the clipping threshold of $C = m + \sigma\,\mathrm{erf}^{-1}(1 - \delta/2N)$ is used. Note that in reality there are two reasons that our ϵ guarantees will be even better. Please refer to the discussion in Appendix B.

Finally, it is worth mentioning that there are alternatives to using DP SentencePiece algorithm. Firstly one can use SentencePiece trained on a related public dataset. We explore the performance of such models in §5.1. Alternatively, one can consider using models that don't require a pre-trained tokenizer, such as ByT5 (Xue et al., 2021). The character-level SentencePiece algorithm is sometimes seen as more “private” than the unigram one, however that is not a precise definition of privacy.

3.2 DP-Training

For DP-Training, we protect individual example privacy and implement the algorithm outlined in (Abadi et al., 2016). Specifically, we use the AdaFactor optimizer that was used for training T5 with the following adjustments (See Figure 1):

1. We take the individual examples' gradients and clip each to some fixed norm (determined by privacy parameters).
2. When taking the parameter update step, noise (determined by privacy parameters) is added to the accumulated gradients.

3.2.1 Fast per-example gradients with JAX

We use a reimplementation[1] of T5 in JAX (Bradbury et al., 2018) and Flax (Heek et al., 2020). By doing so, we can follow Subramani et al. (2020) in leveraging JAX's vectorization supported by the XLA compiler. JAX lets us vectorize ('vmap') the computation of the gradient of the loss on a *single* example, so that we obtain a *batch* of per-example gradients efficiently. This way, we still get most of the speedup of batched neural network training, while having a correct implementation of DP. For each example, we average the loss incurred over all target tokens in the target sequence, compute the gradient, and then clip the gradient norm. An update for a batch of examples computes the gradient for each example in parallel (using vectorization), accumulates the gradients and adds noise, before updating the parameters of the model.

4 Experiments

Hyperparameters. Our experiments in the main text use T5 small, which has 6 encoder layers, 6 decoder layers, 8 64-dimensional heads, embedding dimension of 512, MLP dimension of 2048. We chose T5 small since it is relatively fast to train and produces results comparable with that of larger models (see Appendix A for a discussion of the effect of the model size on pre-train and fine-tuning performance). We use AdaFactor (Shazeer and Stern, 2018) with learning rate 0.5, decay rate 0.8, warm-up 1000, and `rsqrt` learning rate decay.

Datasets. We use The Colossal Clean Crawled Corpus (C4; Raffel et al., 2020) as a pre-training task and look into the original "prefix" unsupervised training objective, that predicts next tokens given the context and the span corruption training objective (Raffel et al., 2020, §3.3.4), where randomly removed spans of the input are predicted. We use 512 tokens as input/context and attempt to predict 114 target tokens. To evaluate fine-tuning performance, we utilize GLUE datasets (Wang et al., 2018) that allows to evaluate model performance accross a range of NLU tasks.

Ablations. We look separately into the effect of DP-SentencePiece and DP-Train on Memorization and pretraining and subsequent fine-tuning performance. For DP-SP we use the unigram SentencePiece algorithm (Kudo and Richardson, 2018).

Pre-training and fine-tuning performance. It is known that DP-training hurts the utility (e.g., accuracy) of models. However, the common scenario in NLP is that models are pre-trained on some data and then subsequently fine-tuned for the end task. We look into private *pre-training* (contrary to the majority of the papers which look into private fine-tuning of a publicly pre-trained model), and we hope that (public) fine-tuning such privately pre-trained models on public data provides the same utility as publicly pre-trained models. For these experiments, we train T5 models with the span corruption objective with a batch size of 8192 for 100K steps, and fine-tune on GLUE for 150K additional steps with a (standard) 128 batch size. The batch size of 8192 was chosen for pre-training since it provides good performance for DP-Train. T5 without DP-training trains with approximately the same performance using a batch size of 128, however for DP-Training it is known that the batch size should be increased significantly in order to get reasonable performance (see Appendix D). For a fair comparison we use the same batch size for the baseline and DP-T5 variants.[2] Another alternative is to tune hyper-parameters (for both baseline and DP variants) and compare the best possible models, however since tuning parameters for DP will change the ϵ guarantees, we don't go this route. We use an initial learning rate of 0.5 (both baseline and DP-T5 variants pre-training) and weight decay, and train with 64 cores. For DP-training, please refer to Appendix 5 for details on noise, clipping norm and ϵ. For the Full DP T5 model, we use a DP-SP unigram model trained on C4 with $\epsilon = 0.17$ (see Table 1) and combine it with DP-Training with various noise levels.

Additionally, Appendix D discusses additional modifications that can be further explored to minimize the utility drop due to DP-Train.

Testing for memorization. It is expected that DP (both DP-training and DP-SP) should reduce data memorization of the models. To verify that, we conduct an evaluation similar to Carlini et al. (2020) and Lee et al. (2021). In particular, we train models on C4 and attempt to "extract" the training data by providing an input prefix and allowing model to generate the rest of the sequence.

[1]Our code is available at `https://github.com/google-research/google-research/tree/master/private_text_transformers`.

[2]The DP version is only 25% slower than the baseline.

We use 512 tokens prefix length, similar to the input length that was used for training. When the model generates the output, we calculate the *exact match* on a per-instance basis and report the average across all instances in the dataset. Exact match is different from per-token accuracy: it is either 0 or 1 for each prefix, depending on whether the model generated the exact target sequence or not, whereas *token accuracy* would report how many predicted tokens match the target tokens for each instance. Exact match allows us to gauge what fraction of instances was output **verbatim** by the model, serving as a useful metric for memorization. Token-level accuracy serves as an additional metric; while getting some tokens right does not guarantee that memorization occurred, high values of token-level accuracy would indicate that some tokens generated by the model words might have matched those from the target exactly. Finally, we also report *median edit distance* between predicted and target tokens, averaged across all instances in the data. This metric also serves as indirect way of measuring memorization.

The difference between our setup and that of Carlini et al. (2020) and Lee et al. (2021) is that our model is an encoder-decoder, as opposed to GPT-2 which is a decoder only. Additionally, on top of using just next word prediction as a training objective (referred to as prefix training), we get to experiment with the span corruption objective.

We test for memorization on the the same four C4-based datasets as Lee et al. (2021):

- **Train dup and Train unique**: contains examples from the training set that had near-duplicates and which had no near-duplicates, respectively, in the training set.
- **Valid in train and Valid unique**: examples from the validation set (not used for training directly) which are very similar to the examples from the training data and data that contains examples from the validation set which had no near-duplicates, respectively.

We would expect the most memorization to be exhibited on Train dup, and the least memorization to be present on Valid unique data.

5 Results & Discussion

5.1 Pre-training & Fine-tuning performance

Table 1 presents an ablation study of DP-SentencePiece's effect on pre-training and fine-tuning performance. Additionally, we explore how tokens pre-selected via SentencePiece trained on other public (for example Wikipedia) datasets affect the performance of both pre-training and fine tuning tasks. First, we see that the best pre-training accuracy does not necessarily translate into the best fine tuning accuracy. Second, we see that DP-SP (unigram) serves as a regularizer on the pre-training task, significantly improving pre-training performance (approx. 13% improvement for the best ϵ). This might be due to the fact that the C4 pretraining data is not clean; without DP-SP, misspelling and non-words like "rein-forced;" were passed to the SentencePiece algorithm and resulted in sub-optimal tokens being selected. For example, for $\epsilon = 0.17$, 88.6% of C4 words were dropped at the histogram creation stage, resulting in only the most common 11.4% of the words being used for token selection. Next, we can see that SP trained on Wikipedia, a different dataset that we may consider public, performs just as well (if not better on the pre-training task). The choice of (public) data is important here, and if the data on which the tokenizer is trained is not similar to the training data, the performance might be compromised. Additionally, character SentencePiece, while without any privacy guarantees, provides excellent pre-train and fine-tuning performance. Finally, other SentencePiece models (like BPE) might be more robust to the noisy data than the unigram and char models we trained, so it is possible that the regularization effect of DP will not be as pronounced for those. We chose unigram because it is the SP algorithm used for the majority of T5 models.

Table 2 demonstrates pretraining and fine-tuning performance of DP-Train (only) models and Fully Private (Full-DP) models that combine DP-SP and DP-Train. We observe that again better pre-training utility does not directly translate into better downstream fine-tuning performance. Even for the most stringent guarantees of DP-Train (ϵ of 6.06) which result in approx 20% of pretrain accuracy drop, on average GLUE fine-tuned performance is not significantly different from the baseline. Full-DP is able to recover or improve pre-train accuracy. On average, we also see that full-DP is not significantly better on subsequent fine-tuning tasks (e.g. mnli_m, mnli_msm, qnli etc), however for some tasks (e.g. cola) fine-tuning performance is significantly better than that of a (non-private) baseline. On average, even DP-train models have approximately the same

	Model name	ϵ	Pretrain	GLUE fine-tuning									
				cola	*mnli_m*	*mnli_msm*	*mrpc*	*qnli*	*qqp*	*rte*	*sst2*	*stsb*	*Avg*
SP	T5, unigram C4-SP	∞	56.4	84.6	87.7	88.5	91.7	95.8	96.4	90.4	92.3	66.2	88.2
	90% signif (+/-)		*0.1*	*0.0*	*2.5*	*2.0*	*1.4*	*0.8*	*0.8*	*2.1*	*0.9*	*0.8*	*1.1*
	T5, unigram wiki-SP	∞	71.9	91.4	82.6	82.9	91.1	95.3	95.4	90.4	85.3	50.0	84.9
	T5, char C4-SP	∞	76.5	97.3	94.6	94.4	92.6	96.6	97.3	94.8	95.1	65.9	92.0
DP-SP	T5, unigram C4-DP-SP	0.17	69.1	91.5	90.8	91.5	91.1	96.6	96.4	92.9	94.1	56.6	89.1
		3.37	63.7	90.5	90.2	90.8	88.8	96.4	95.3	90.6	91.4	62.6	88.5
		33.70	65.5	84.6	87.1	87.6	94.9	96.9	97.7	92.6	91.7	58.3	87.9
		336.00	66.0	84.6	86.2	87.3	95.1	96.9	97.5	92.5	91.8	58.8	87.9

Table 1: Accuracy on C4 span corruption pre-training and GLUE fine-tuning. **SP** is standard SentencePiece, **DP-SP** is private SentencePiece, and *Avg* is the average across GLUE tasks.

Model	ϵ	Pretrain	GLUE fine-tuning									
			cola	*mnli_m*	*mnli_msm*	*mrpc*	*qnli*	*qqp*	*rte*	*sst2*	*stsb*	*Avg*
Baseline	∞	59.8	85.3	90.7	91.4	95.0	96.4	98.1	91.4	94.8	61.3	89.4
90% signif (+/-)		*0.4*	*0.0*	*3.1*	*3.2*	*2.1*	*0.6*	*0.5*	*0.5*	*12.5*	*0.8*	*2.4*
DP-train	6.06	39.5	82.7	87.4	87.2	92.6	94.8	97.6	90.72	91.4	60.4	87.2
	8.69	41.3	82.6	87.2	87.0	93.6	94.8	97.6	90.75	91.8	62.9	87.6
	13.46	42.8	81.9	87.0	87.2	93.4	94.6	97.6	90.8	91.6	62.2	87.4
	319.19	48.1	82.5	88.1	88.1	93.1	94.5	97.7	91.39	92.6	62.1	87.8
Full DP	6.23	51.2	90.6	91.6	91.5	92.1	96.3	97.8	93.5	93.3	57.1	89.3
	8.86	52.4	90.1	92.0	91.8	92.5	96.5	97.9	93.3	94.0	57.5	89.5
	13.63	55.4	90.0	91.9	91.8	93.2	96.4	97.9	93.8	93.9	57.7	89.6
	319.36	62.8	90.6	92.2	92.2	93.1	96.6	98.0	94.6	94.2	67.7	91.0

Table 2: Accuracy on C4 span corruption pretrain and GLUE fine tuning tasks. **DP-Train** are T5 models trained with public SentencePiece but DP-Adafactor training, and **Full-DP** combines DP-SP and DP-Train.

GLUE performance (difference insignificant).

5.2 Memorization discussion

Table 3 presents the result of memorization experiments for various fully-private (Full-DP) T5 models, along with ablation studies that look into effect of DP-SentencePiece and DP-Training only.

Firstly, we highlight that span corruption training is *extremely robust* to memorization. Even baseline non-private models do not output any training data verbatim when prompted with input from the `Train Dup` dataset (exact match of 0%). While some tokens generated by the model do match target tokens (the TA column for Train dup), it is only 0.29% of all (114) generated tokens on average, which indicates that almost no words were output verbatim from the training data. At the same time, the pretrain accuracy of a baseline model indicates that its performance is reasonably good (59.8% teacher-forced accuracy). The take-away message here is that if memorization is of a concern, one way to address it is to use span corruption training objective. Zero memorization (EM of 0%) is preserved after publicly fine-tuning these models on GLUE and retesting for pre-training data memorization.

One important caveat here is that the span corruption training objective was splitting a piece of text into input/target randomly, so it is possible a different definition of memorization would be more suitable. For example, instead of using prompts from `Train dup` and targets that immediately follow these prompts, it would be more suitable to test span corruption models to see if they can output a randomly selected set of words given other words in a sentence. This would mimmic the training objective of span corruption better. At the same time, since Lee et al. (2021) showed that it is duplicate sentences that are major source of memorization, and for such duplicate sentences, span corruption inputs and targets (randomly selected) during training will be different for each duplicate, it is still our belief that even with such alternative memorization definition span corruption models will be extremely robust. We leave this for future work.

Prefix training however does exhibit a lot of memorization, confirming the results from Carlini et al. (2020) and Lee et al. (2021). The baseline model outputs approx. 2% of the training data verbatim, when prompted with 512 tokens from the `Train dup` dataset. This number falls to 0.03% of the data for instances that were not repeated many times in the training data (`Train unique`). Full DP-T5 models are able to not only improve the pre-train performance, but also mitigate the effect of memorization: for an ϵ of 6.23, Full DP-T5

	Model	Eps	Pretrain	Train dup			Train unique			Valid in train			Valid unique		
				EM	*TA*	*MED*	*EM*	*TA*	*MED*	*EM*	*TA*	*MED*	*EM*	*TA*	*MED*
Span Corruption	Baseline	∞	59.8	0.00	0.29	133	0.00	0.14	220	0.00	0.29	126	0.00	0.14	228
	DP-Train	6.06	39.5	0.00	0.02	137	0.00	0.01	229	0.00	0.02	119	0.00	0.01	228
		8.69	41.3	0.00	0.06	140	0.00	0.04	226	0.00	0.07	125	0.00	0.04	224
		13.46	42.8	0.00	0.05	136	0.00	0.03	228	0.00	0.05	118	0.00	0.03	226
		319.19	48.1	0.00	0.25	143	0.00	0.25	143	0.00	0.26	133	0.00	0.14	215
	DP-SP	0.17	72.9	0.00	1.12	141	0.00	0.58	211	0.00	1.18	134	0.00	0.58	209
		3.37	70.7	0.00	1.25	144	0.00	0.71	203	0.00	1.34	138	0.00	0.71	201
		33.68	68.6	0.00	0.30	129	0.00	0.16	219	0.00	0.31	112	0.00	0.16	217
		336.00	68.7	0.00	0.13	133	0.00	0.07	226	0.00	0.14	116	0.00	0.07	224
	Full DP	6.23	51.2	0.00	0.51	158	0.00	0.28	207	0.00	0.51	153	0.00	0.29	206
		8.86	52.4	0.00	1.18	132	0.00	0.78	216	0.00	1.33	116	0.00	0.79	214
		13.63	55.4	0.00	0.34	137	0.00	0.21	213	0.00	0.36	127	0.00	0.21	212
		319.36	62.7	0.00	1.44	130	0.00	0.81	204	0.00	1.52	125	0.00	0.82	202
Prefix Training	Baseline	∞	39.6	2.20	3.33	112	0.03	0.48	208	1.17	2.49	104	0.02	0.48	206
	DP-Train	6.06	23.0	0.00	0.21	136	0.00	0.15	226	0.00	0.25	118	0.00	0.15	225
		8.69	23.5	0.05	0.25	135	0.00	0.16	227	0.01	0.27	117	0.00	0.16	225
		13.46	24.2	0.06	0.25	135	0.00	0.16	226	0.05	0.28	118	0.00	0.16	225
		319.19	31.4	0.15	0.64	127	0.00	0.27	218	0.07	0.65	111	0.00	0.27	217
	DP-SP	0.17	55.7	1.44	3.41	120	0.01	1.22	216	0.73	3.11	216	0.01	1.23	105
		3.37	53.1	1.48	3.35	118	0.02	1.16	215	0.75	2.99	215	0.01	1.17	103
		33.68	49.9	1.90	3.04	117	0.02	0.76	214	0.99	2.46	214	0.01	0.76	103
		336.00	49.8	1.95	3.10	117	0.01	0.75	215	0.99	2.47	215	0.01	0.75	103
	Full DP	6.23	42.8	0.01	2.02	135	0.00	1.17	225	0.00	2.16	117	0.00	1.18	224
		8.86	43.2	0.01	2.12	134	0.00	1.27	223	0.00	2.30	117	0.00	1.28	222
		13.63	43.7	0.01	1.67	136	0.00	0.97	225	0.00	1.83	118	0.00	0.98	223
		319.36	49.2	0.15	1.57	131	0.00	0.85	222	0.08	1.66	113	0.00	0.86	221

Table 3: Effect of DP on Memorization. **EM** is Exact match, **TA** is Token-level accuracy, **MED** is Median Edit Distance. **DP-SP** are T5 models trained only with Differentially Private SentencePiece. **DP-Train** are T5 models trained with public SentencePiece but DP-Adafactor training, and **Full-DP** combines DP-SP and DP-Train.

models output verbatim only 0.006% of training instances that were repeated multiple times in the training data (366x less memorization) and even very large values of ϵ like 320 provide 15x improvement in memorization as measured by exact match. For instances that occurred in training only a few times (`Train unique`), pretty much any level of DP-protection provides almost full elimination of memorization (0.002% EM even for an ϵ of 320.)

With respect to ablation studies, the DP-Training has the most (positive) effect on memorization, accounting for the majority of improvement of Full DP models. DP-SentencePiece does affect memorization of T5 models, albeit much less than DP-Train. For example, for prompts that look like training data duplicates, DP-SP (ϵ of 0.17) is able to reduce the exact match from approx. 2% to 1.4%. For a large ϵ this protective effect is almost non-existent.

Finally, it is important to mention that while DP T5 does significantly reduce memorization (on the prefix objective), it does not completely eliminate it, especially for sentences that were repeated multiple times (`Train dup`). As mentioned previously, it might be because such sentences will have a lower level of protection guarantees and thus can still be output verbatim. Combining DP (DP-SP and DP-Training) with deduplication techniques from Lee et al. (2021) should thus be beneficial.

6 Conclusion

While the majority of recent work looks into private fine-tuning of pre-trained NLP models, we investigated how private *pre-training* of a model on a large corpus of data affects its pre-training and subsequent fine-tuning performance, as well as how much memorization such privately pre-trained models exhibit. We worked with T5, a transformer-based encoder-decoder, and demonstrated how to achieve a fully private T5 version by introducing DP-SentencePiece to train a differentially private subword tokenizer, and implementing DP-Training for the actual pre-training. We leveraged recent advances in JAX to do so without incurring a large performance hit in terms of training speed. Our results show that both DP-SentencePiece and DP-Training contribute to reducing memorization, but that the latter has the largest effect. Moreover, we demonstrated that the span corruption task from Raffel et al. (2020) also effectively mitigates memorization, which isn't the case for the next token prediction objective. We also show that fully private T5 models exhibit reasonable pre-training performance and don't hurt subsequent fine-tuning, and that private SentencePiece serves as a regularizer on noisy datasets and is able to improve pre-training and fine-tuning performance of models such as T5.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*.

Raef Bassily, Adam D. Smith, and Abhradeep Thakurta. 2014. Private empirical risk minimization, revisited. *CoRR*, abs/1405.7085.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. JAX: composable transformations of Python+NumPy programs.

Nicholas Carlini, Chang Liu, Jernej Kos, Úlfar Erlingsson, and Dawn Song. 2018. The secret sharer: Measuring unintended neural network memorization & extracting secrets. *CoRR*, abs/1802.08232.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2020. Extracting training data from large language models. *CoRR*, abs/2012.07805.

Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. 2011. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(29):1069–1109.

Ali Davody, David Ifeoluwa Adelani, Thomas Kleinbauer, and Dietrich Klakow. 2020. Robust differentially private training of deep neural networks. *CoRR*, abs/2006.10919.

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.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus.

Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. 2020. Flax: A neural network library and ecosystem for JAX.

Shlomo Hoory, Amir Feder, Avichai Tendler, Alon Cohen, Sofia Erell, Itay Laish, Hootan Nakhost, Uri Stemmer, Ayelet Benjamini, Avinatan Hassidim, and Yossi Matias. 2021a. Learning and evaluating a differentially private pre-trained language model. In *Proceedings of the Third Workshop on Privacy in Natural Language Processing*, pages 21–29, Online. Association for Computational Linguistics.

Shlomo Hoory, Amir Feder, Avichai Tendler, Sofia Erell, Alon Peled-Cohen, Itay Laish, Hootan Nakhost, Uri Stemmer, Ayelet Benjamini, Avinatan Hassidim, and Yossi Matias. 2021b. 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.

Roger Iyengar, Joseph P. Near, Dawn Song, Om Thakkar, Abhradeep Thakurta, and Lun Wang. 2019. Towards practical differentially private convex optimization. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 299–316.

Gavin Kerrigan, Dylan Slack, and Jens Tuyls. 2020. Differentially private language models benefit from public pre-training.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2021. Deduplicating training data makes language models better. *CoRR*, abs/2107.06499.

Xuechen Li, Florian Tramèr, Percy Liang, and Tatsunori Hashimoto. 2021. Large language models can be strong differentially private learners.

Ilya Mironov. 2017. Renyi differential privacy. *CoRR*, abs/1702.07476.

Ilya Mironov, Kunal Talwar, and Li Zhang. 2019. Rényi differential privacy of the sampled gaussian mechanism. *CoRR*, abs/1908.10530.

Nicolas Papernot, Abhradeep Thakurta, Shuang Song, Steve Chien, and Úlfar Erlingsson. 2020. Tempered sigmoid activations for deep learning with differential privacy.

NhatHai Phan, Yue Wang, Xintao Wu, and Dejing Dou. 2016. Differential privacy preservation for deep auto-encoders: An application of human behavior prediction. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 1309–1316. AAAI Press.

Venkatadheeraj Pichapati, Ananda Theertha Suresh, Felix X. Yu, Sashank J. Reddi, and Sanjiv Kumar. 2019. Adaclip: Adaptive clipping for private SGD. *CoRR*, abs/1908.07643.

Francesco Pittaluga, Sanjeev J. Koppal, and Ayan Chakrabarti. 2018. Learning privacy preserving encodings through adversarial training. *CoRR*, abs/1802.05214.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Anna Rogers. 2021. Changing the world by changing the data. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2182–2194, Online. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost.

Pranav Subramani, Nicholas Vadivelu, and Gautam Kamath. 2020. Enabling fast differentially private sgd via just-in-time compilation and vectorization. *ArXiv*, abs/2010.09063.

Aleena Thomas, David Ifeoluwa Adelani, Ali Davody, Aditya Mogadala, and Dietrich Klakow. 2020. Investigating the impact of pre-trained word embeddings on memorization in neural networks. In *Text, Speech, and Dialogue: 23rd International Conference, TSD 2020, Brno, Czech Republic, September 8–11, 2020, Proceedings*, page 273–281, Berlin, Heidelberg. Springer-Verlag.

Florian Tramer and Dan Boneh. 2021. Differentially private learning needs better features (or much more data). In *International Conference on Learning Representations*.

Laurens van der Maaten and Awni Y. Hannun. 2020. The trade-offs of private prediction. *CoRR*, abs/2007.05089.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2021. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *CoRR*, abs/2105.13626.

A Small vs Base vs Large T5 model performance

Table 4 outlines the performance difference between the "small", "base" and "large" T5 architectures on Pretraining span corruption task on the C4 dataset and subsequent fine tuning performance on GLUE datasets. Since the performance differences are not very large, we chose to run all of our experiments with the T5 small architecture.

B DP-SentencePiece ϵ guarantees discussion

Theorem 1 (Hoory et al., 2021b) uses the notion of k and m (maximum L2 and infinity norms) of sentence-level (1-D vector) histogram. If N is the length of the sentence, for a histogram where we exactly count the number of times each word appears in the sentence, we have $k = m = N$. However the definition of a sentence is loose. Note that ideally a “sentence” would mimic how the subsequent training of T5 model will happen, since we aim to obtain example-level DP protection. T5 however is not trained on words—it is trained on tokens—and tokens are chosen by the SentencePiece algorithm. The length of tokens chosen varies: they can be as short as 1 character or long tokens of 3-4 characters or more. Our T5 experiments use 512 input tokens as features and 114 tokens as target, so the whole “example” or “sentence” is 626 tokens. Just as in (Hoory et al., 2021b), we assume that sentence length is 256 words, which is a very pessimistic estimate—this translates into approximately 2.45 tokens per word. Note that if in reality 626 tokens represent fewer words, the SentencePiece $\epsilon \sim N$ will be better.

Additionally, (Hoory et al., 2021b) authors provide a Corollary that allows to obtain slightly better ϵ bounds while using approximately the same clipping norm and the same level of noise.

C DP-Training ϵ discussion

In order to come up with ϵ guarantees for DP-Training, we consider that C4 dataset has approximately 133,897,2430,182 words. Assuming, just as in DP-SentencePiece discussion, that each word consists of approx. 2.45 tokens, and each training example is 512+114 tokens, our total number of examples is approximately 5,240,387,307 (and the δ used is 1/5,240,387,307).

Table 5 presents the noise and clip norm that we used for our experiments, along with ϵ guarantees.

We use differential privacy accountant (Abadi et al., 2016) and Renyi Differential Privacy outlined in (Mironov, 2017), (Mironov et al., 2019) which has been implemented in https://github.com/tensorflow/privacy.

When combining DP-SP and DP-Training, we use simple composition and sum the respective ϵ and δ.

D Related work: Mitigating Utility Drop in DP-Training

To preface the below discussion, we would like to highlight that the goal of our paper was not to obtain the smallest pre-training utility drop possible. We thus did limited hyperparameter tuning and didn’t explore methods outlined below.

Some works attempts to mitigate the performance drop by considering architectural or hyperparameter changes. For example, (Tramer and Boneh, 2021) argues that the drop can be mitigated by the large amount of training data, whereas (Bassily et al., 2014) shows that DP risk minimization bounds, compared to non DP bounds, have a polynomial dependency on the number of features and ϵ. (Papernot et al., 2020) demonstrated the need to adjust the parameters for DP training and argued for use of different activation functions when using DP. Additionally, various other architecture adjustments like increasing the batch size, or using batch/layer normalization were proposed, for example in (Davody et al., 2020). It is also important to point out that hyper-parameter tuning (which includes changing batch size, learning rate, architecture etc) can’t be used “for free” with DP-training as it has to be accounted for in the privacy budget. Thus for DP training experiments, it is common not to tune the hyperparameters and choose some predefined values before the experiments begin.

Another direction explored in literature is the modification to DP-SGD algorithm itself. For example, (Davody et al., 2020) introduce scale-invariant DP-SGD and use normalization techniques to dampen the effect of additional noise during training. In this modification, the final network weights are sampled from the normal distribution whose mean and variance were updated to account for the privacy budget.

Finally, instead of going for differentially private training, the utility drop can be mitigated by relaxation of privacy guarantees. For example, private-inference, which works by adding noise to the final

Model	# params	Pretrain	GLUE									
			cola	*mnli_m*	*mnli_msm*	*mrpc*	*qnli*	*qqp*	*rte*	*sst2*	*stsb*	Avg
T5 Small	60M	60.7	88.2	94.3	94.4	96.1	98.2	97.9	95.3	95.4	71.7	92.4
T5 Base	220M	64.6	92.0	95.5	95.7	96.2	98.9	98.2	96.3	97.0	71.5	93.5
T5 Large	770M	66.7	92.1	96.4	96.5	97.3	99.0	98.2	98.0	98.1	71.9	94.2

Table 4: Performance of various T5 architectures on pretrain C4 task and their fine tuning performance on GLUE.

Clip	Noise	ϵ
0.001	0.40	6.0573157
0.001	0.35	8.6898032
0.001	0.30	13.4586238
0.001	0.20	47.2630501
0.001	0.10	319.1941523

Table 5: DP-Train clipping norm and noise hyper-parameters and ϵ achieved for a batch size of 8192, 100K steps.

prediction of the models trained conventionally, is known to protect just the labels of the data and does not provide full DP guarantees with respect to all the model weights and data features (van der Maaten and Hannun, 2020). Additionally, heuristic methods like in (Pittaluga et al., 2018) that prevent discovery of some predefined "private attributes" from the data (.e.g like inferring the race of the speaker) can be used without any DP guarantees.