Just Fine-tune Twice: Selective Differential Privacy for Large Language Models

Weiyan Shi[†], **Ryan Shea**[†], **Si Chen**[‡], **Chiyuan Zhang**[◇], **Ruoxi Jia**[‡], **Zhou Yu**[†]
Columbia University[†], Virginia Tech[‡], Google Research[◇]
{ws2634,rs4235}@columbia.edu, chensi@vt.edu, chiyuan@google.com,
ruoxijia@vt.edu, zy2461@columbia.edu

Abstract

Protecting large language models from privacy leakage is becoming increasingly crucial with their wide adoption in real-world products. Yet applying *differential privacy* (DP), a canonical notion with provable privacy guarantees for machine learning models, to those models remains challenging due to the trade-off between model utility and privacy loss. Utilizing the fact that sensitive information in language data tends to be sparse, Shi et al. (2021) formalized a DP notion extension called *Selective Differential Privacy* (SDP) to protect only the sensitive tokens defined by a policy function. However, their algorithm only works for RNN-based models. In this paper, we develop a novel framework, *Just Fine-tune Twice* (JFT), that achieves SDP for state-of-the-art large transformer-based models. Our method is easy to implement: it first fine-tunes the model with *redacted* in-domain data, and then fine-tunes it again with the *original* in-domain data using a private training mechanism. Furthermore, we study the scenario of imperfect implementation of policy functions that misses sensitive tokens and develop systematic methods to handle it. Experiments show that our method achieves strong utility compared to previous baselines. We also analyze the SDP privacy guarantee empirically with the canary insertion attack[1].

1 Introduction

With the rapid advancement in natural language processing (NLP), it has become increasingly important to protect NLP models from leaking privacy information. Previous work has attempted to tackle this challenge by applying *differential privacy* (DP, Dwork et al., 2014) on these models (McMahan et al., 2018; Li et al., 2021). However, existing DP learning algorithms suffer from *limited user control* and *low utility*, as they protect the entirety of each training example (e.g., one complete sentence) regardless of users' privacy preference, and therefore tend to be overly pessimistic when only partial information in a training example is sensitive. This problem is particularly pertinent in NLP, as NLP training data are often mixed with sparse domain-dependent private information, and not all tokens need to be protected. For example, for the sentence *"My SSN is 123-45-6789"*, only the last few tokens of the actual SSN need to be protected.

In fact, the definition of DP does *not* prevent us *at all* from protecting only the sensitive part of data. Specifically, DP ensures that the output of a data analysis algorithm stays roughly the same for neighboring datasets, while providing the flexibility to adjust the definition of neighboring relation to specific application contexts. Shi et al. (2021) recently proposed an instantiation of DP, called Selective-DP (SDP), which defines neighboring datasets to differ only in the sensitive part of a training example and as a result, SDP *selectively* hides the difference in the sensitive part only. SDP is particularly suitable for NLP and many other unstructured, high-dimensional data, wherein sensitive information only accounts for a small part. But their privacy mechanism to achieve SDP suffers from three problems: 1) it requires substantial knowledge about the model to separate the private and public variables, and it is unclear how their algorithm tailored to recurrent neural networks could be extended to modern Transformer-based NLP models; 2) it has only been evaluated with explicit private entities but not with contextual sensitive information; 3) it doesn't provide protection for undetected sensitive tokens; These constraints limit the applicability of SDP in real-world scenarios.

Large language models (LLMs) (Vaswani et al., 2017) have achieved tremendous success in NLP. They are pretrained on a massive amount of public textual data, and thus excel at capturing general language structures. A common practice in

[1]Our code and data are available at https://github.com/wyshi/sdp_transformers

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 6327–6340
December 7-11, 2022

NLP is to fine-tune these LLMs on downstream tasks. Such a fine-tuning process also works well in the private training context. Previously, Yu et al. (2021a) showed that privately fine-tuning an additional small set of parameters on top of off-the-shelf LLMs with private data achieves comparable performance to non-private baselines. Inspired by their findings, in this paper, we propose a two-phase fine-tuning privacy mechanism, *Just fine-tune twice* (JFT), to achieve SDP for LLMs. Instead of directly using off-the-shelf models to fine-tune once, we have two fine-tuning steps: 1) we first redact the in-domain data of the downstream tasks, and fine-tune the model with these in-domain redacted data (*redacted*-fine-tune), and 2) then privately fine-tune the model on the original private data (*private*-fine-tune). This additional *redacted*-fine-tune step allows the model to directly learn information from the in-domain data and thus leads to a better model initialization for the second *private*-fine-tune step. Moreover, in the *redacted*-fine-tune step, we show that even with limited public data (where manual screening is possible), JFT achieves better utility than fine-tune-once baselines. Additionally, we can apply lightly-noised optimizers and privacy amplification to protect undetected sensitive tokens.

Our contributions are as follows. First, we propose an effective and generalizable privacy mechanism to achieve SDP for large language models for various NLP tasks. Second, we design secret detectors of different privacy levels (explicit and contextual sensitive data) and study their implications on the models. Third, our method can utilize even a small amount of public data to achieve better utility, and mitigate the missed sensitive token problem with lightly-noised optimizer and privacy amplification. Finally, we show that, opposite to the common belief that privacy is at odds with utility, private learning doesn't have to conflict with the utility because private information in the data could be irrelevant to the learning task.

2 Preliminary

A differentially private algorithm hides the difference between two neighboring datasets.

Definition 1 (Differential Privacy)**.** *Given a domain $\mathcal{D}$, any two neighboring datasets $D, D' \subseteq \mathcal{D}$ a randomized algorithm $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ is (ϵ_w, δ_w)-differential private if for all neighboring datasets D and D' and all $T \subseteq \mathcal{R}$,*

$$\Pr[\mathcal{M}(\mathcal{D}) \in T] \leq e^{\epsilon_w} \Pr[\mathcal{M}(\mathcal{D}') \in T] + \delta_w.$$

The neighboring relation captures what is protected. Traditional DP literature has considered neighboring datasets as those differing in one training example; thus, the corresponding DP protects each training example as a whole. We denote by ϵ_w and δ_w the privacy parameters achieved under this traditional neighboring relation definition. Given the sparsity of sensitive information in language data, this instantiation of neighboring relations is apparently over-pessimistic. Shi et al. (2021) proposed Selective-DP (SDP), which instantiates the neighboring datasets to be those that differ in the sensitive attributes of a training sample; as a result, SDP *selectively* hides the difference in the sensitive part only. In the context of NLP, a training example could be a sentence or a paragraph depending on the task and the attributes are individual tokens.

In this paper, we will focus on designing learning algorithms to achieve SDP. Formally, SDP relies on a policy function F that specifies the sensitive information in a training example to be protected in an *application-dependent* fashion.

Definition 2 (Policy Function)**.** *A policy function $F : \tau \to \{0, 1\}^{|r|}$ decides which attributes of an example $r \in \tau$ are public ($F(r)_i = 1$) or private ($F(r)_i = 0$). $|r|$ is the number of attributes in r.*

Detecting private information manually in a large corpus based on the policy function is often costly. In that case, one may resort to building automatic *secret detectors* to identify the sensitive attributes. A simple example of a secret detector is a regular expression to capture phone numbers. However, secret detectors could miss some private attributes and produce false negatives, which intuitively would weaken the privacy guarantees. Existing work (Doudalis et al., 2017; Shi et al., 2021; Zhao et al., 2022) that selectively protects data either assumes a perfect detector or uses an overly conservative detector with a low false negative but at the cost of a high false positive. In this paper, we provide alternative ways to address this issue with a better privacy-utility tradeoff (Section 3).

With F, SDP defines F-Neighbors.

Definition 3. *(F-Neighbors). Consider a policy function F and two datasets D and D'. D' is a F-neighbor of D (denoted by $D' \in N_F(D)$) if and only if $\exists r \in D$ s.t., $F(r)$ has at least one private attribute, $\exists r' \in D'$ s.t., $F(r)$ and $F(r')$ differ by at least one private attribute, and $D' = D \backslash \{r\} \cup \{r'\}$.*

Given the definition, the dataset with *"My ID is 123"* and the dataset with *"My ID is 456"* are

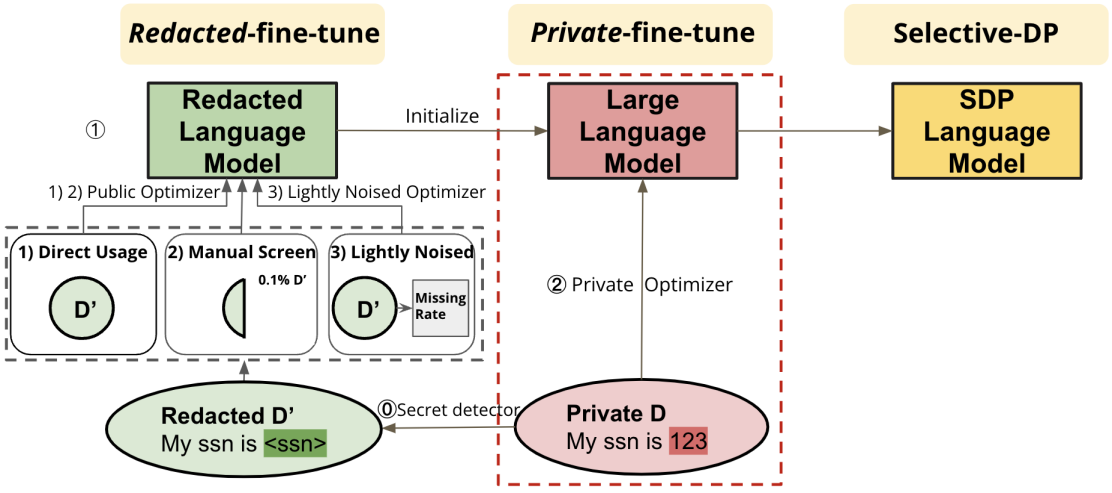

Figure 1: The two-phase JFT mechanism. As pre-processing, we apply the secret detector to redact the private data D and obtain the redacted data D'. Next, depending on the detector's performance, we use different ways to fine-tune the language model on the redacted D' and obtain a redacted model. Then we fine-tune the model again on the private data D with a private optimizer (e.g., DPSGD) to achieve an SDP-protected model.

F-Neighbors (because except for the actual ID number, the other tokens are the same), while the datasets with *"Hi there"* and the dataset with *"Hello there"* are not F-neighbors because the only token they differ in is not sensitive. An SDP algorithm guarantees that F-neighbors cannot be distinguished by attackers if they observed the output.

Definition 4. *(Selective Differential Privacy). Under a policy function F, a randomized algorithm $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ satisfies $(F, \epsilon_s, \delta_s)$-selective differential privacy if for $\forall D, D' \in N_F(D)$, and $\forall T \subseteq \mathcal{R}$, $\Pr[\mathcal{M}(D) \in T] \leq e^{\epsilon_s} \Pr[\mathcal{M}(D') \in T] + \delta_s$.*

In this paper, we differentiate between SDP and DP for two reasons. Firstly, we want to highlight that the privacy parameters associated with SDP (ϵ_s and δ_s) and DP (ϵ_w and δ_w) are incomparable. For instance, one cannot claim which of (1, 0.001)-SDP and (2, 0.001)-DP provides stronger privacy guarantees because they are under different privacy notions. To meaningfully present the value of these privacy parameters, we need to specify under which definition these parameters are calculated; to meaningfully compare the value of these parameters, we need to make sure that the parameters are calculated under the same privacy definition. Secondly, we would like to remain the same terminology as our main reference, Shi et al. (2021), which also uses the terms SDP and DP to refer to the privacy guarantees under the two different neighboring relations. In the rest of the paper, we use different notations to distinguish the privacy parameters associated with SDP (ϵ_s and δ_s) and DP (ϵ_w and δ_w).

3 JFT: Just Fine-tune Twice

Now we describe JFT, a two-phase privacy mechanism to achieve SDP for large language models (Figure 1). In the first **redacted-fine-tune** phase, we redact the private data D with a secret detector to obtain the redacted version D', and learn a redacted model from D' in a privacy-preserving way. In the second **private-fine-tune** phase, we further fine-tune the redacted model (from phase one) on the private data D with a private optimizer to achieve SDP guarantees.

3.1 Phase 1: *Redacted*-fine-tune

JFT is built upon the observation that the public portion of the in-domain data does not require protection and can be utilized in various ways to help the model learn in-domain information. In this phase, we apply the secret detector to redact the private data D and obtain redacted in-domain data D'. Dependent on the detector performance, we propose the three following methods to use D' to fine-tune off-the-shelf language models.

Direct Usage. If the secret detector masks all the sensitive information in D (which is possible when D is small enough to support thorough inspection or when a detector is very conservative and removes most of the essential information, see examples in Table 1), we can use the redacted D' directly to fine-tune the model with a public and unnoised optimizer like SGD.

Selective Manual Screening. If the secret detector is imperfect, we can select an affordable subset

from D' and then manually sanitize all the missed secrets. Then we fine-tune the model with this small sanitized subset with a public optimizer. Experiments show that even with a small amount of sanitized in-domain data, the resulting model still outperforms the traditional DP learning algorithms that pessimistically protect every single token.

Lightly-Noised Fine-tuning. When the detector is imperfect, besides manually screening out the missed secrets, we could also employ a private optimizer to train on D' that contains missed sensitive tokens. Because missed tokens only account for a small portion of D', intuitively, a much smaller noise is needed to ensure the privacy of the missed tokens than the noise magnitude required to ensure the privacy of the entire D'. We propose to leverage *privacy amplification by subsampling* (PAS) (Balle et al., 2018) to calculate the privacy parameters associated with the private optimizer. The intuition of PAS is that if we perform a DP mechanism on random subsamples of the data, and one data point is not included in the subsamples, nothing about it could be leaked. In this way, we could amplify the privacy guarantee. In our scenario, we need to protect the missed sensitive tokens. If we know the secret detector's missing rate m (i.e., m=number of missed sensitive tokens/total tokens, the probability of sampling a missed secret), we can calculate the privacy budget ϵ_s by privacy amplification using the subsampling ratio m.

Note that the application of PAS requires the number of missed tokens that appear in any batch to be the same, which does not necessarily hold. Hence, the privacy parameters calculated from privacy amplification are an empirical estimate of the actual privacy loss. In practice, the secret detector's missing rate is unknown and we need to estimate it (denoted as $\widetilde{m}$). Then we change the original sampling rate p_0 in moment accounting-based privacy parameter calculation (Abadi et al., 2016) to $p = p_0 * \widetilde{m}$ and calculate the noise injected into each private optimizer iteration according to a predefined privacy budget ϵ under p.

In our experiments, we sample 0.01% training data for 10 times, and estimate the 95% confidence interval of the missing rate, $[\widetilde{m}_{\text{low}}, \widetilde{m}_{\text{high}}]$. For both $\widetilde{m}_{\text{low}}$ and $\widetilde{m}_{\text{high}}$, we can calculate an associated ϵ_{low} and ϵ_{high} according to Theorem 9 in Balle et al. (2018), and report both ϵ.

3.2 Phase 2: *Private*-fine-tune.

In the second phase, we initialize the model with the redacted model from phase one, and fine-tune it with the original private D and a private optimizer (e.g., DPSGD (Abadi et al., 2016) or any other more advanced private optimizer that achieves DP).

Unlike the privacy mechanism in Shi et al. (2021), our algorithm does not require knowledge about the models or the tasks, and therefore can be easily applied to different models such as GPT2 (Radford et al., 2019) and Roberta (Liu et al., 2019), and different tasks such as language generation and natural language understanding. See Section A.2 for more implementation details.

One-phase vs two-phase. Compared to conventional differentially private training, our algorithm introduces an additional stage that involves redaction and regular unnoised training on redacted data. In fact, the computational cost of the additional stage is much lower than the cost originally incurred by DP learning, because the additional first phase does not need costly per-sample gradient clipping and noising operations. And redaction is a common first-step people are already doing and familiar with. Also, there exist abundant off-the-shelf tools that allow redaction at scale.

3.3 Privacy Analysis

We provide Theorem 1 for privacy analysis. It ensures that, if the user has a secret detector with 100% recall, JFT-trained models achieve (0,0)-SDP after phase one and (ϵ, δ)-SDP after phase two. A secret detector with 100% recall is possible if the user can afford manual inspection or have enough domain knowledge. When a detector with 100% recall is not possible, we use *lightly-noised fine-tuning* to empirically protect the missed secrets as mentioned in Section 3.1.

Theorem 1. *Given that 1) in the first phase, the data used for fine-tuning do not contain sensitive tokens and a public optimizer is used, and 2) in the second phase, the private optimizer achieves* (ϵ, δ)*-DP,* JFT *achieves* (ϵ, δ)*-SDP.*

The proofs are deferred to Section A.1. The theorem shows that under direct usage or selective screening of D', JFT achieves SDP with the same privacy parameter values as the ones pertaining to the private optimizer used in the second phase.

4 Secret Detectors of Different Levels

Typical private information includes personal-identifiable information (PII) such as name and birthday. But as pointed out in Brown et al. (2022), one key challenge in NLP is that private information is often contextual. For example, they presented a dialogue between Alice and Bob about Alice's divorce (Table 1): none of the tokens in *"What are you going to do about the custody of the kids?"*, are PII by themselves, but combined together, the semantics reveals private information.

To build generalizable secret detectors, we utilize off-the-shelf NER, dependency parser, and POS tagger in spaCy (Honnibal and Montani, 2017) to label each token, and redact different sets of tokens to achieve the different privacy levels below (entity level and contextual level). To qualitatively show their protection levels, we apply them to redact two sentences from the divorce dialogue in Brown et al. (2022). The results are in Table 1.

Secret Detector	**What are you going to do about the custody of the kids?**	**Did you hear Alice is getting divorced?**
Low entity	What are you going to do about the custody of the kids?	Did you hear <PERSON> is getting divorced??
High entity	What are you going to do about the custody of the kids?	Did you hear <PERSON> is getting divorced??
Low contextual	<PRON> are <PRON> going to do about <OBJ> of the <OBJ>?	Did <PRON> hear <PROPN> is getting divorced??
High contextual	<PRON> are <PRON> <VERB> to <VERB> about <OBJ> of the <OBJ>?	Did <PRON> <VERB> <PROPN> is getting <VERB>??

Table 1: Results of different secret detectors on the two example sentences from Brown et al. (2022).

Low entity redacts four types of named entities (person, organization, date, and location), which are considered PII defined by the US Department of Labor[2]. We use NER in spaCy to detect them. If we apply this detector, *"Did you hear Alice is getting divorced?"* becomes *"Did you hear <PERSON> is getting divorced?"* An attacker who attacks a model trained on the latter sentence can at best learn about the divorce but cannot know who.

High entity redacts all the 18 entities in spaCy including the four above and more, like time entity[3].

The two secret detectors above rely on named entities, so they are more explicit than the two detectors below, which consider the overall sentence structure and thus are more contextual.

Low contextual protects all the 18 entities plus proper nouns, pronouns, and sentence subjects and objects. This detector drastically increases the privacy level: we cannot get any useful information from the left example in Table 1.

High contextual further redacts all the verbs, in addition to the tokens redacted by the low contextual detector. It increases the privacy level even further and we cannot learn anything from both examples. This is to *stress-test* JFT and see the model utility when the majority of the tokens are redacted [4].

Human language is diverse, and private information can take various forms. So instead of designing sophisticated algorithms, we intentionally rely on common NLP tools to build easy-to-use domain-agnostic secret detectors with *high recalls* to protect privacy as much as possible. As shown in Table 1, these detectors tend to over-sanitize the sentences. But we will show later, even with over-redaction, JFT still achieve good performance. Private textual information can be treated more sophisticatedly, but how to better detect private information is not the focus of this paper. Our goal is to show that simply redacting tokens achieves promising performance and JFT is compatible with better private information detection algorithms to further improve the results.

5 Experiments

We conduct experiments on two NLP tasks: *1)* natural language understanding (NLU, on GLUE) and *2)* language generation (on Wikitext-2 and ABCD).

Datasets. *1)* **GLUE** (Wang et al., 2018) is a widely-used multi-task benchmark dataset for NLU. It contains sensitive information such as name and date. *2)* **Wikitext-2** (Merity et al., 2017) contains Wikipedia articles with private information such as name and date. *3)* **ABCD** (Chen et al., 2021) is a human-human customer service dialogue dataset under real-world scenarios with user private information such as name and order IDs.

Models. We use Roberta (Liu et al., 2019) for the NLU classification task and GPT2 (Radford et al., 2019) for the language generation task. Due to computational constraints, we use Roberta-base and GPT2-small for the experiments. We use an effi-

[2] https://www.dol.gov/general/ppii

[3] For the full list, see https://spacy.io/usage/linguistic-features#named-entities.

[4] Note that "divorced" is actually an adjective in the example. As mentioned earlier, we can address the detector's mistakes with lightly-noised fine-tuning.

cient implementation of DPSGD in Li et al. (2021). Based on previous studies (Li et al., 2021; Yu et al., 2021a), larger DP models usually achieve better results and thus we expect that larger SDP models will achieve even better performances.

Baselines. *1)* **No-DP**: the model is fine-tuned using regular Adam optimizer (Kingma and Ba, 2014) without extra noise and hence it does not have any privacy guarantees (i.e., $\epsilon_w = \epsilon_s = \infty$). *2)* **DPSGD**[5]: the model is fine-tuned with traditional DPSGD Abadi et al. (2016) where the gradient is clipped and noised in every gradient descent iteration (we employ the DP-Adam variant where the optimizer is Adam but its gradient privatization is the same as DPSGD, and we keep the term DPSGD as it is more accessible to the community). While DPSGD was originally proposed to achieve the DP guarantees that protect a training example as a whole, it can also achieve SDP guarantees with the same privacy parameters (i.e., $\epsilon_s = \epsilon_w$ and $\delta_s = \delta_w$). *3)* **CRT**: the model is trained with the recently proposed Confidentially Redacted Training (Zhao et al., 2022) that achieves (ϵ_c, δ_c)-Confidentiality. Confidentiality is a new definition related to SDP but different from SDP, it ensures the indistinguishability between a secret and a <MASK> token, so its privacy parameters are not directly comparable to SDP. Thus, we add the same amount of noise to CRT and SDP, empirically compare SDP and CRT with the canary insertion attack in Figure 2 and 3, and report the utility in Table 6 in the Appendix. *4)* **Redacted**: We also present the utility of the redacted models since they are also privacy-preserving. Note when the secret detector is perfect, the redacted models have a perfect SDP privacy guarantee (i.e., $\epsilon_s = 0$). However, it does not allow the model to learn from sensitive tokens at all. JFT, by contrast, empowers the model to learn from sensitive data with a flexible, tunable tradeoff between privacy and utility. Moreover, it provides ways to offer quantifiable privacy in the presence of imperfect secret detectors.

Our models. *1)* **JFT**: this is our JFT model directly using the redacted data in phase one. *2)* **JFT +manual screening**: this is JFT using a subset of the redacted data where missed secrets are manually filtered out in phase one. *3)* **JFT +light noise**: this is JFT where we add light noises according to the estimated missing rate in phase one.

[5]The public baseline from Roberta does not use infilling, for a fair comparison, we chose the DPSGD baseline without infilling from (Li et al., 2021). But compared to DPSGD baselines with infilling from (Li et al., 2021), JFT without infilling with a conservative secret detector is still better or comparable. Previous studies found that infilling improves the results, so it is possible that with infilling, JFT can achieve even better results.

6 Results

We show three major findings: *1)* the impacts of secret detectors are task-dependent on the resulting JFT models, but even for conservative contextual detectors (30%+ tokens are redacted), JFT still achieves better results than naive DPSGD (Section 6.1); *2)* despite the small scale, using the manual screened in-domain data still improves the JFT model utility (Section 6.2); *3)* lightly noised optimizer with privacy amplification protects missed sensitive tokens from attacks (Section 6.3).

There is always a privacy-utility trade-off, so larger epsilons lead to better utilities but worse privacy. And when comparing models, we need to look at the model utility under a similar privacy budget. An epsilon of 1 to 3 is commonly used in various privacy literature (Yu et al., 2021a; Li et al., 2021; Zhao et al., 2022). In our experiments, we pre-calculated the privacy parameter so that an ϵ of around 3 is spent when training ends.

6.1 Secret Detectors of Different Levels

Table 2 show the results on GLUE (left) and generation (right). Pct is the percentage of sensitive tokens redacted by the detector. ϵ_s is the SDP privacy budget, the lower the better. We compare model utility under a similar privacy budget ϵ_s.

Natural Language Understanding. Table 2 (left) shows that under a similar ϵ_s, all the JFT models achieve better performance than the DPSGD baseline, even when over 40% of tokens are redacted.

Besides, for all the tasks, all the redacted models achieve reasonable utility, even when a large portion of the tokens are redacted. For example, the redacted model (high contextual) is better than DPSGD on MNLI (83.23 vs 82.10, 44.27% redacted) and SST-2 (91.17 vs 86.12, 38.13% redacted). This confirms the motivation of SDP that when building private NLP models, we should not naively protect all the tokens regardless of their properties. Instead, we should consider if the sensitive tokens will impact the task. If not, we can simply redact them to build private models.

Also, if JFT can improve the redacted model depends on the task. For SST-2 on sentiment analysis, the *private*-fine-tune step does not improve

Direct Usage		NLU on GLUE, δ_s=1/2|D_{train}|												Language Generation, δ_s=1e-6					
		MNLI			QQP			QNLI			SST-2			WIKITEXT-2			ABCD		
Model	**Detector**	**Pct**	**Acc↑**	ϵ_s	**Pct**	**Acc↑**	ϵ_s	**Pct**	**Acc↑**	ϵ_s	**Pct**	**Acc↑**	ϵ_s	**Pct**	**PPL↓**	ϵ_s	**Pct**	**PPL↓**	ϵ_s
No-fine-tune	-	-	31.82	-	-	36.82	-	-	50.54	-	-	50.92	-	-	30.08	-	-	13.60	-
No-DP	-	-	87.60	-	-	91.90	-	-	92.80	-	-	94.80	-	-	20.48	-	-	4.96	-
DPSGD	-	-	82.10	2.75	-	85.41	2.75	-	84.62	2.57	-	86.12	2.41	-	27.05	2.58	-	8.31	2.65
DPSGD (+spe)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	30.32	2.58	-	17.75	2.71
Redacted	low ent	6.09%	86.67	-	6.05%	88.74	-	12.19%	89.64	-	1.79%	93.58	-	11.3%	22.50	-	2.7%	6.98	-
JFT	low ent	6.09%	**85.74**	0.92	6.05%	**88.19**	2.58	12.19%	**89.57**	2.37	1.79%	**92.09**	2.06	11.3%	**21.86**	2.58	2.7%	**6.09**	2.71
Redacted	high ent	8.63%	86.50	-	8.30%	88.36	-	17.18%	88.96	-	3.01%	93.58	-	16.4%	24.32	-	3.1%	7.32	-
JFT	high ent	8.63%	**85.61**	0.99	8.30%	**88.05**	2.58	17.18%	**89.35**	2.37	3.01%	**92.20**	2.12	16.4%	**22.55**	2.58	3.1%	**6.25**	2.71
Redacted	low ctx	31.19%	85.14	-	32.61%	85.59	-	35.68%	85.30	-	22.19%	92.55	-	34.8%	37.90	-	22.3%	28.28	-
JFT	low ctx	31.19%	**85.02**	1.23	32.61%	**87.00**	2.41	35.68%	**87.99**	2.52	22.19%	**92.43**	2.17	34.8%	**25.62**	2.58	22.3%	8.80	2.71
Stress-test																			
Redacted	high ctx	44.27%	83.23	-	45.93%	83.48	-	45.59%	82.81	-	38.13%	91.86	-	45.0%	54.29	-	28.6%	65.45	-
JFT	high ctx	44.27%	**84.11**	1.18	45.93%	**86.42**	2.67	45.59%	**87.06**	2.41	38.13%	**91.17**	2.17	45.0%	27.19	1.96	28.6%	12.93	2.71

Table 2: Model utility and privacy guarantee on GLUE dev sets for NLU (left) and WikiText-2 and ABCD for generation (right). **Detector**: the secret detector to realize the policy function. **low ent**: low entity detector. **low ctx**: low contextual detector. **Pct**: the percentage of sensitive tokens. **Acc**: accuracy. **PPL**: perplexity. ϵ_s: SDP privacy guarantee. **DPSGD(+spe)**: DPSGD with added special tokens. For QNLI and SST-2, $\delta \approx$1e-5; for MNLI and QQP, $\delta \approx$1e-6 due to the large data size; No-DP and DPSGD results are from Liu et al. (2019) and Li et al. (2021). For generation tasks, δ=1e-6, we train the baselines and report the results. The models in bold are better than DPSGD.

the redacted model. This is because, the redacted models achieve a high accuracy (even the worst accuracy is 91.86, only a 2.94 drop from the SOTA public model with an accuracy of 94.8), and fine-tuning them on the private data with noisy gradients is not enough to close the small gap. But for tasks with a bigger gap between the redacted and No-DP models (e.g., MNLI, QQP, and QNLI), JFT can further improve the redacted model. Besides, the gap between the redacted model and the corresponding JFT model becomes bigger as the privacy level increases: for QNLI (low contextual), the gap is 87.99-85.30=2.69, while for QNLI (high contextual), the gap is 87.06-82.81=4.25. This shows the model does learn useful information from the sensitive tokens during the *private*-fine-tune step.

Language Generation. Table 2 (right) shows the language generation results. We note that language generation is different from NLU tasks, because for NLU, the models used for initialization without any fine-tuning ("No-fine-tune" in Table 2) start with a bad accuracy ($\leq$ 50%, just random guess), and adding special tokens to it would still start with a random guess, so additional special tokens will not impact the final results greatly. But for the generation task, the "No-fine-tune" GPT2 is already a strong model for initialization with ppl=30.08 on Wikitext-2 and 13.60 on ABCD, and we found that adding special tokens would disturb this initialization and greatly impact the final result. Because all the JFT models have added special tokens like "<MASK>" and "SYS:", for a fair comparison, we report two DPSGD baselines, one without special tokens ("DPSGD") and one with special tokens ("DPSGD (+spe)"). See Section A.2 for more discussions on the impact of special tokens.

Compared to "DPSGD (+spe)", all JFT models achieve better model utility on both datasets. For "DPSGD" without special tokens, fine-tuning on the downstream tasks improves the model from 30.08 to 27.05 (the improvement Δ=3.03); for JFT (low contextual), it is initialized with the redacted model with ppl=37.90, and privately fine-tuning it improves the perplexity to 25.62 (Δ=12.28). This shows that although the initialization seems worse (30.08 vs 37.90), since the redacted model is fine-tuned directly on in-domain redacted data, it does learn useful information from the first *redacted*-fine-tune step, and the second *private*-fine-tune step can further improve upon the redacted model. For JFT (high contextual) for the stress test, although 45% tokens are masked and the language structure is largely impacted, JFT still improves the redacted model utility from 54.29 to 27.19 (Δ=27.10) and performs on par with DPSGD (27.05 vs 27.19).

6.2 Selective Manual Screening

As mentioned earlier, secret detectors can miss certain secrets and we can manually filter out the missed secrets at a small scale and fine-tune with the small manually sanitized set. Denote the original data as D_0. We sample 0.1% data from D_0, apply the high entity secret detectors (because it is less conservative and could miss secrets), and manually sanitize the missed secrets to get D'. We use $D' = 0.1\%D_0$ during *redacted*-fine-tune to train

	MNLI		QQP		QNLI		SST-2		WikiText-2		ABCD	
Model	**Acc↑**	95%-ϵ_s	**Acc↑**	95%-ϵ_s	**Acc↑**	95%-ϵ_s	**Acc↑**	95%-ϵ_s	**PPL↓**	95%-ϵ_s	**PPL↓**	95%-ϵ_s
DPSGD	82.10	2.75	85.41	2.75	84.62	2.57	86.12	2.41	27.05	2.58	8.31	2.65
Missing rate m (95% CI)	(0.3%, 1.2%)		(0.3%, 1.2%)		(0.1%, 0.6%)		(0%, 1.8%)		(0.4%, 0.7%)		(0.1%, 1.2%)	
Recall (95% CI)	(87.5, 96.7)		(85.9, 96.1)		(96.4, 99.3)		(40.2, 100)		(95.6, 97.8)		(62.7, 97.4)	
JFT+light noise	**82.76**	(0.08, 0.43)	85.28	(1.40, 1.71)	**84.88**	(2.29, 2.68)	**89.33**	(0, 0.43)	**25.21**	(2.73, 2.92)	**5.78**	(1.08, 1.60)
Conservative Estimation												
Missing rate m	8.6%		8.3%		17.2%		3.0%		16.4%		3.0%	
Recall	0		0		0		0		0		0	
JFT+light conservative noise	82.00	0.45	84.77	2.91	84.02	2.95	**89.22**	0.43	**26.59**	3.03	**6.64**	1.67

Table 3: Privacy-amplified JFT performance on all the tasks, with the high entity detector. We report the estimated 95% confidence interval (CI) of the missing rate m, recall and the corresponding 95% CI of the ϵ_s.

Manual Screening	D' **(redacted)=0.1%** D_0, D **(private)=100%** D_0					
Task	**MNLI** Acc ↑	**QQP** Acc ↑	**QNLI** Acc ↑	**SST-2** Acc ↑	**WikiText-2** PPL ↓	**ABCD** PPL ↓
D' **size**	300	300	100	100	10	10
DPSGD	82.10	85.41	84.62	86.12	27.05	8.31
Redacted	52.52	75.25	66.48	88.88	28.06	9.36
JFT+manual screening	**82.45**	**86.24**	**85.00**	**90.83**	**26.72**	**7.84**

Table 4: Manual screening results on the high entity secret detector. D_0: original data. D': the inspected redacted data. D: the private data. $\epsilon \approx 3$. D' size is the number of records used in the *redacted*-fine-tune phase.

the redacted model, and the entire D_0 as D during *private*-fine-tune to obtain JFT models.

Table 4 shows the results. D'=0.1%D_0 contains 100∼300 examples for GLUE and 10 articles and 10 dialogues for Wikitext-2 and ABCD respectively. On all the tasks, JFT achieves better utilities than DPSGD. This shows that even fine-tuning with a small manually-screened in-domain subset can still help the model learn in-domain information, and lead to better utility. We also simulate a completely low-resource setting where we simply have limited training data (i.e., D'=0.1% D_0, D=0.1% D_0). See Section A.4 for the results.

6.3 Lightly Noised Optimizer with Privacy Amplification

Besides manually inspecting the missed secrets, we can also use noised optimizers in the first phase to protect the missed secrets from attacks and then adopt privacy amplification to estimate the corresponding privacy parameters. We again perform the experiments on the high entity detector and Table 3 shows the results. We convert the missing rate m (%) to the recall of the secret detector, i.e., recall=(1-m/Pct), where Pct is the percentage of sensitive tokens among all tokens. "JFT+light noise" shows the model performance with a noised optimizer and privacy amplification employed in the first step. Besides, we add more noise than needed to obtain a conservative model ("JFT+light conservative noise"): for instance, MNLI data contains 8.63% sensitive tokens, although the secret detector's missing rate m ranges from $(0.3\%, 1.2\%)$ with 95% probability, we assume m to be 8.63% (i.e., it miss all sensitive tokens and thus recall=0) to calculate and add the noises that are more than actually needed.

The results show that "JFT+light noise" achieves better utility than DPSGD, especially on generation tasks. The perplexity is (25.21 vs 27.05, 5.78 vs 8.31), and the estimated ϵ_s ranges from $(2.73, 2.92)$ and $(1.08, 1.60)$ with 95% probability. Even for the conservative models, "JFT+light conservative noise" is still better than DPSGD (26.59 vs 27.05, and 6.64 vs 8.31).

6.4 Attack Results

We perform the canary insertion attack (Carlini et al., 2019) to empirically show how much the models memorize the training data unintentionally. The attack is to insert a canary of a certain format into the training data, and calculate its exposure, which is the rank of the inserted canary amongst all possible values of the same format. The lower the exposure is, the safer the model is. In our experiments, we insert the canary *"My ID is 341752"* into the training data for 10 times to make the performance difference of different models more salient. By definition, for a six-digit canary, an exposure close to $\log_2(10^6) \approx 19.9$ means the canary can be extracted by the attackers. The result is in Figure 2. Each point on the figure is a model checkpoint. The X-axis shows the perplexity (utility), and the y-axis is the exposure (privacy), so Figure 2 shows different models' privacy-utility tradeoffs.

One major reason why the model remembers a canary is that it has seen the canary many times. For "No-DP", initially, its exposure is low because it hasn't seen the canary many times. But because

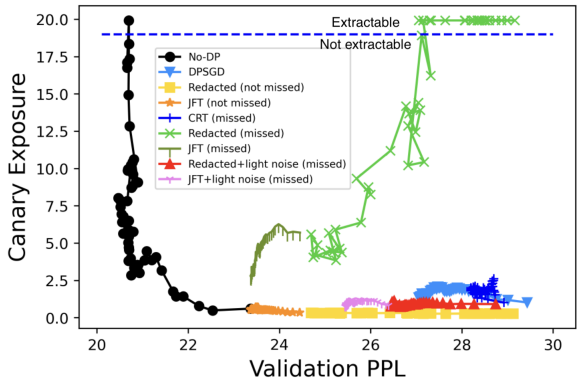

Figure 2: Canary exposure for different models.

the model is unprotected, its exposure is unbounded and increases dramatically after it accesses the data for more epochs. This suggests that models without protection do memorize the data unintentionally.

For protected models (DPSGD, redacted, and JFT), if the canary is captured by the detector ("not missed" in the figure), then the exposure does not increase much even if the data are accessed many times. Under similar exposure, JFT achieves better utility than DPSGD and the redacted models.

But if the secret detector misses the canary (we purposely code it to mark the canary as public), the exposure is increased for both "Redacted (missed)" and "JFT (missed)". But if we add light noise in the first phase ("Redacted+light noise" in red), even if the canary is missed for 10 times, its exposure is still low. If we continue to privately fine-tune in the second phase ("JFT+light noise" in pink), we can further improve the utility but still achieve a low exposure value. Both DPSGD and CRT also achieve a low exposure value if the canary is missed by the secret detector, but with worse utilities than "JFT+light noise". This shows "JFT+light noise" can protect missed secrets while achieving better utility. We also performed the canary insertion attack with one canary inserted once, and 10 different canaries inserted once, shown in Section A.5.

We also tested the membership inference attack (MIA), but it wasn't successful (inference accuracy is around 60% even for public models). Previous studies also observed unsuccessful MIA (Shi et al., 2021; Zhao et al., 2022) and our future work includes developing better MIA for NLP.

7 Related Work

Recent work studied private language models on various model architectures such as RNNs (McMahan et al., 2018; Ramaswamy et al., 2020) and large language models (Anil et al., 2021; Li et al., 2021; Yu et al., 2021a). Li et al. (2021) proposed ghost-clipping to reduce the computational cost in per-sample gradients of DPSGD, and achieved strong private LLMs. Yu et al. (2021a) added a small set of private parameters to off-the-shelf LLMs and privately tune them on private data and obtained performant private models. Most previous works achieve canonical DP. Shi et al. (2021) proposed Selective-DP for applications with sparse sensitive information like NLP, and a privacy mechanism for RNN models. Our work proposes an effective mechanism for LLMs to achieve SDP and study the impact of secret detectors at different levels.

Our work is also closely related to utilizing public data for private learning (Papernot et al., 2018; Tramer and Boneh, 2020; Ghazi et al., 2021; Yu et al., 2021a). One working direction assumes access to large unlabeled public data to train DP models. For example, PATE (Papernot et al., 2016) used unlabeled public data and knowledge distillation to build DP models. Hoory et al. (2021) used public medical data to pre-train domain-specific private vocabulary and models. Another direction leveraged small public data to guide the private updates in lower-dimension subspaces (Zhou et al., 2020; Yu et al., 2021b). Our work is distinct from previous studies: instead of querying public data from outside, we utilize the public portion of in-domain data and achieve SDP with better model utility.

8 Conclusions

In this paper, we propose JFT, which can achieve Selective-DP for large language models. We also design generalizable secret detectors to provide protection at different levels and study their impacts on the resulting SDP models, and address the problem of missed sensitive tokens via selective manual screening and private training with reduced noise, which is justified by privacy amplification. The results show that the proposed JFT produces SDP models with strong performance while remaining robust to the canary insertion attack.

Limitations

Parameter search in DP learning is challenging as the training process takes a long time and is quite sensitive to different parameters (Li et al., 2021). So the findings in the paper are based on the parameter tuning performed by the authors (Table 5), and more parameter tuning could potentially lead to

better results than the results reported in the paper.

In our experiments, we simply fine-tuned the models on the in-domain redacted data without adjustment for the redaction. We could potentially utilize more sophisticated methods to train better redacted models to further improve the JFT utility. Besides, for the selective manual screening experiments, we did not adjust the *redacted*-fine-tune step for the low-resource setting where $D' = 0.1\%D_0$. Future work includes how to train a better redacted model given limited data.

When the gap between the SOTA public model and the redacted model is small, the *private*-fine-tuning step cannot further improve the results because of the noisy gradients (e.g., in SST-2), we plan to develop better algorithms to utilize the redacted data and apply denoising methods (Welch et al., 1995) to close the gap further.

One thing to note is that if the secret detector misses a secret and the secret appears multiple times in the data, then it is likely that the secret detector will miss it multiple times. Therefore, deduplicating the data first is important. We plan to study data deduplication in NLP in the future.

Ethical Considerations

To prevent real-world harm, all the datasets and models used in this paper are already public with either public or synthesized information. So no real personal information will be leaked.

This work tackles the challenge of privacy protection and can be utilized in various domains or applications to build models that preserve privacy. We will release the code to facilitate privacy-preserving model building. The canary insertion attack is well-known (Carlini et al., 2019, 2020) and adjusted specifically for our setting, so it cannot be directly utilized to attack real-world models successfully.

Acknowledgements

We would like to thank Da Yu, Xuechen Li, and Zhiliang Tian for the valuable discussions. We also thank the anonymous area chair and reviewers for their insightful suggestions.

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.

Rohan Anil, Badih Ghazi, Vineet Gupta, Ravi Kumar, and Pasin Manurangsi. 2021. Large-scale differentially private bert. *arXiv preprint arXiv:2108.01624*.

Borja Balle, Gilles Barthe, and Marco Gaboardi. 2018. Privacy amplification by subsampling: Tight analyses via couplings and divergences. *Advances in Neural Information Processing Systems*, 31.

Hannah Brown, Katherine Lee, Fatemehsadat Mireshghallah, Reza Shokri, and Florian Tramèr. 2022. What does it mean for a language model to preserve privacy? *arXiv preprint arXiv:2202.05520*.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 267–284.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2020. Extracting training data from large language models. *arXiv preprint arXiv:2012.07805*.

Derek Chen, Howard Chen, Yi Yang, Alex Lin, and Zhou Yu. 2021. Action-based conversations dataset: A corpus for building more in-depth task-oriented dialogue systems. *arXiv preprint arXiv:2104.00783*.

Stelios Doudalis, Ios Kotsogiannis, Samuel Haney, Ashwin Machanavajjhala, and Sharad Mehrotra. 2017. One-sided differential privacy. *Proceedings of the VLDB Endowment*.

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

Badih Ghazi, Noah Golowich, Ravi Kumar, Pasin Manurangsi, and Chiyuan Zhang. 2021. Deep learning with label differential privacy. *Advances in Neural Information Processing Systems*, 34.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. 2021. Large language models can be strong differentially private learners. *arXiv preprint arXiv:2110.05679*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning differentially private recurrent language models. In *International Conference on Learning Representations*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. *ICLR*.

Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. 2016. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*.

Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. 2018. Scalable private learning with pate. *ICLR*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Swaroop Ramaswamy, Om Thakkar, Rajiv Mathews, Galen Andrew, H Brendan McMahan, and Françoise Beaufays. 2020. Training production language models without memorizing user data. *arXiv preprint arXiv:2009.10031*.

Weiyan Shi, Aiqi Cui, Evan Li, Ruoxi Jia, and Zhou Yu. 2021. Selective differential privacy for language modeling. *arXiv preprint arXiv:2108.12944*.

Florian Tramer and Dan Boneh. 2020. Differentially private learning needs better features (or much more data). *arXiv preprint arXiv:2011.11660*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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. *arXiv preprint arXiv:1804.07461*.

Greg Welch, Gary Bishop, et al. 1995. An introduction to the kalman filter.

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, et al. 2021a. Differentially private fine-tuning of language models. *arXiv preprint arXiv:2110.06500*.

Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. 2021b. Large scale private learning via low-rank reparametrization. In *International Conference on Machine Learning*, pages 12208–12218. PMLR.

Xuandong Zhao, Lei Li, and Yu-Xiang Wang. 2022. Provably confidential language modelling. *arXiv preprint arXiv:2205.01863*.

Yingxue Zhou, Zhiwei Steven Wu, and Arindam Banerjee. 2020. Bypassing the ambient dimension: Private sgd with gradient subspace identification. *arXiv preprint arXiv:2007.03813*.

A Appendix

A.1 Proofs

Theorem 1 (restated)**.** *Given that 1) in the first phase, the data used for fine-tuning does not contain sensitive tokens and a public optimizer is used, and 2) in the second phase, the private optimizer achieves* (ϵ, δ)*-DP,* JFT *achieves* (ϵ, δ)*-SDP.*

Proof. Since the first phase does not incur any privacy loss on the sensitive tokens, the first phase achieves $(0, 0)$-SDP.

DP implies SDP. In other words, if an algorithm achieves (ϵ, δ)-DP, then it also satisfies (ϵ, δ)-SDP. Hence, the second phase achieves (ϵ, δ)-SDP. By the composition of SDP, JFT achieves (ϵ, δ)-SDP.

Note that the first phase achieves (0,0)-SDP but cannot achieve (0,0)-DP. DP aims to protect the entire token sequence, whether it is considered sensitive or non-sensitive by the policy function. Because the first phase does not noise the non-sensitive tokens at all, it cannot ensure DP. □

A.2 Implementation Details

Notes on special tokens. When fine-tuning LLMs, it is a common practice to add new special tokens to fit the need of the downstream tasks. For example, in dialogue tasks, we often have prompts like "SYS:" and "USR:" to indicate the speaker in the data. This step doesn't affect the public models that much (20.48 without special tokens vs 20.44 with special tokens), but as it does change the model structure (additional embeddings) and the model initialization, we notice that in our experiments, DPSGD is sensitive to the addition of special tokens (because the model initialization is changed): after reasonable amounts of parameter tuning (see Table 5), DPSGD initialized with the original GPT achieves 27.05 in PPL, while DPSGD with added special tokens achieves 30.32 in PPL on Wikitext-2. The gap could potentially be reduced with more parameter tuning, but we just want to mention that in practice, it may not be easy to find the best parameters. In our experiments, for WikiText-2, we add *<mask>* as the special token; for ABCD, as it is a dialogue task, we add *<mask>, "ACT:", "SYS:"*, and *"USR:"*. Since all the JFT models have added special tokens, we report two DPSGD results, one without special tokens and one with special tokens, for a fair comparison in terms of model structure.

Also, the secret detectors replace the sensitive information with artificial special tokens such as "<SSN>" and "<NAME>". But these tokens don't appear in the validation or test set and thus inserting them will skew the training data distribution and lead to inferior results, especially when the sensitive token portion is high. In our experiments, we mask the detected sensitive information with the same "<mask>" token and ignore this special token in the loss calculation. In this way, for models with an existing "<mask>" token (like Roberta), we can utilize the existing embedding; for models without "<mask>", the model only needs to learn one additional special embedding. This improves the validation perplexity from 64.82 to 37.90 for the redacted GPT2 model with the low contextual secret detector.

We could potentially apply the same secret detector on the validation and test set to mitigate special token issues. However, this causes two concerns: 1) if the secrete detector redacts 45% of tokens (e.g. the high contextual one redacts all the verbs, etc), then the performance on validation/test is not informative at all, and cannot be compared to the public baseline; 2) in the past privacy literature (Papernot et al., 2018; Ghazi et al., 2021), the conventional problem setup considers validation/test sets as public and focuses only on the training privacy. We inherit the same treatment to be comparable with prior literature. But in privacy-related NLP problems, how to treat the validation/test sets remains an open question as they can contain private information.

Our experiments find that adding many special tokens impacts the results. In the future, we plan to study how to treat special tokens better in privacy-preserving LMs.

Parameters	**Range**
ϵ	3
Clipping norm C	0.1
Batch size	{256, 512, 1024}
Learning rate	$\{5, 10, 50\} \cdot 10^{-5}$
Epochs E	{10, 100, 200, 600}
Noise scale σ	Pre-calculated so that ϵ is spent when training ends

Table 5: Hyper-parameter tuning range.

Hyper-parameter tuning. Hyper-parameter tuning remains a challenging problem in DP learning as the training takes a long time, and the model can be sensitive to the hyper-parameters. Guided

		MNLI		QQP		QNLI		SST-2		WIKITEXT-2		ABCD	
Model	**Detector**	**Acc↑**	**Privacy**	**Acc↑**	**Privacy**	**Acc↑**	**Privacy**	**Acc↑**	**Privacy**	**PPL↓**	**Privacy**	**PPL↓**	**Privacy**
CRT	low ent	81.45	(2.21, δ_c)-Conf	84.10	(2.47, δ_c)-Conf	83.36	(2.30, δ_c)-Conf	87.39	(2.22, δ_c)-Conf	28.20	(2.19, δ_c)-Conf	9.09	(2.73, δ_c)-Conf
JFT	low ent	**85.74**	(0.92, δ_s)-SDP	**88.19**	(2.58, δ_s)-SDP	**89.57**	(2.37, δ_s)-SDP	**92.09**	(2.06, δ_s)-SDP	**21.86**	(2.58, δ_s)-SDP	**6.09**	(2.71, δ_s)-SDP
CRT	high ent	81.24	(2.63, δ_c)-Conf	83.64	(2.13, δ_c)-Conf	82.78	(2.56, δ_c)-Conf	87.27	(2.22, δ_c)-Conf	28.47	(1.38, δ_c)-Conf	9.20	(2.71, δ_c)-Conf
JFT	high ent	**85.61**	(0.99, δ_s)-SDP	**88.05**	(2.58, δ_s)-SDP	**89.35**	(2.37, δ_s)-SDP	**92.20**	(2.12, δ_s)-SDP	**22.55**	(2.58, δ_s)-SDP	**6.25**	(2.71, δ_s)-SDP
CRT	low ctx	78.85	(2.46, δ_c)-Conf	79.17	(2.50, δ_c)-Conf	81.15	(2.24, δ_c)-Conf	85.67	(2.22, δ_c)-Conf	28.87	(0.69, δ_c)-Conf	12.70	(0.34, δ_c)-Conf
JFT	low ctx	**85.02**	(1.23, δ_s)-SDP	**87.00**	(2.41, δ_s)-SDP	**87.99**	(2.52, δ_s)-SDP	**92.43**	(2.17, δ_s)-SDP	**25.62**	(2.58, δ_s)-SDP	**8.80**	(2.71, δ_s)-SDP
Stress-test													
CRT	high ctx	74.61	(2.68, δ_c)-Conf	77.57	(2.64, δ_c)-Conf	79.41	(2.30, δ_c)-Conf	86.01	(2.33, δ_c)-Conf	29.13	(0.47, δ_c)-Conf	13.11	(0.35, δ_c)-Conf
JFT	high ctx	**84.11**	(1.18, δ_s)-SDP	**86.42**	(2.67, δ_s)-SDP	**87.06**	(2.41, δ_s)-SDP	**91.17**	(2.17, δ_s)-SDP	**27.19**	(1.96, δ_s)-SDP	**12.93**	(2.71, δ_s)-SDP

Table 6: Comparison between JFT and CRT. JFT achieves (ϵ_s, δ_s)-*Selective-DP*, while CRT achieves (ϵ_c, δ_c)-*Confidentiality*, so the ϵ are *not* directly comparable. For QNLI and SST-2, $\delta_s = \delta_c \approx$1e-5; for MNLI and QQP, $\delta_s = \delta_c \approx$1e-6. For generation task, $\delta_s = \delta_c \approx$1e-6. We add the same amount of noise to JFT and CRT (noise calculated based on ϵ=3), and report the best model validation utility. "CRT" results are based on our implementation of Zhao et al. (2022).

Batch size	Epoch	PPL
256	200	27.04
512	200	27.05
1024	200	27.18
1024	600	27.01
1024	10	28.84

Table 7: Model utility with different hyper-parameters.

by Li et al. (2021), we tune the learning rate, the number of training epochs, and the batch size on the validation set and report the best results. Please refer to Table 5 for the parameter range.

In our experiments, we tuned the batch size, learning rate, and epoch number for the best models in different settings. In practice, we found increasing the epoch number and batch size helps, but these two factors can interact with each other, the gain could be small after the batch size and epoch number are large enough. Table 7 shows the convergent model utility of DPSGD on WikiText-2 with different hyper-parameters. In our experiments, we pick the best set of parameters that balances the computational cost and the utility.

A.3 Comparison with CRT

Table 6 shows the comparison between JFT and CRT on the model utility and privacy guarantee. Note that JFT achieves (ϵ_s, δ_s)-SDP, and CRT realizes (ϵ_c, δ_c)-Confidentiality, because the underlying privacy notion is different, we cannot directly compare the ϵ. We pre-calculate the noises given $\epsilon = 3$, add the same amount of noises to both CRT and JFT, and report the best valid utility. With the same amount of noise added, JFT achieves better model utilities than CRT for all the tasks across different secret detectors.

A.4 Low-Resource Results

Privacy protection is important in oftentimes low-resource domains such as health care. We simulate the low-resource setting where we have both limited redacted and private data, i.e., (D'=0.1% D_0, D=0.1% D_0) and the results are in Table 8.

The redacted model always performs better than DPSGD, suggesting that for low-resource settings, we can simply redact the data to train the model instead of employing differential privacy. Also, for the QNLI task, JFT shows promising results. With 0.1% training data (100 records), the redacted model improve the accuracy from random guess to 66.5%. JFT can even further improve the accuracy to 67.49%. But the baseline DPSGD fails to improve the model at all (accuracy=50.54%). We plan to study how to better fine-tune the redacted model privately with limited data.

Manual Screening	D'=0.1% D_0, D=0.1% D_0					
Task	**MNLI** Acc ↑	**QQP** Acc ↑	**QNLI** Acc ↑	**SST-2** Acc ↑	**WikiText-2** PPL ↓	**ABCD** PPL ↓
D'	300	300	100	100	10	10
DPSGD	32.86	63.18	50.54	56.77	30.08	13.66
Redacted	**52.52**	**75.25**	66.48	**88.88**	**28.06**	**9.36**
JFT+manual screening	50.61	75.18	**67.49**	88.53	28.15	9.40

Table 8: Manual screening results when both D and D' are limited in size. D' is the redacted data, D is the private data.

A.5 Canary Insertion Attack

Figure 3 shows the canary insertion attack result when the canary is inserted only once. We see that the exposure is low for all the models (<3, so not extractable) including the public "No-DP" without any protections. This agrees with Figure 4 in Carlini et al. (2019) that when the canary is inserted for very limited times, its exposure is low.

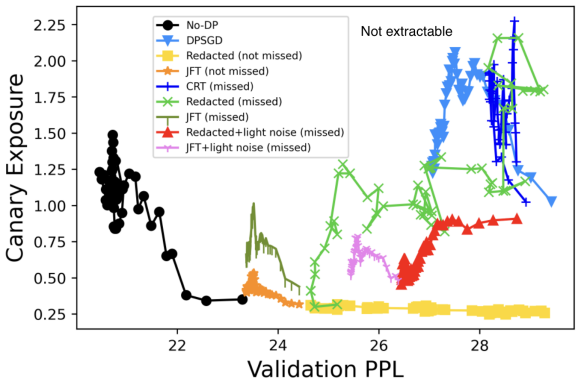

Figure 3: Exposure for different models when the canary is inserted only once. The exposures are all small (<3) even for public models.

Figure 4 shows the canary insertion attack results when we insert ten different canaries into the training data. The exposure is the average exposure of the ten canaries. In this experiment, we treat the inserted canaries as the only secrets, so the "Redacted" and "JFT" model utilities are close to the black "No-DP" model, and we can better compare with the "No-DP" model. We also artificially vary the recall of the secret detector to see the effect. "Recall=0.4" means that the detector can only detect four of the ten canaries. Because each canary only appears once in the dataset, the exposure is low, similar to the ones in Figure 3. But if the recall is higher (0.6), the exposure will still be lower. "JFT +light noise" achieves low exposure, similar to the baseline DPSGD that protects all canaries, but with much higher utility over DPSGD.

These experiments may suggest that for large NLP models, if the sensitive tokens only appear for very limited times, they may not be extracted using the canary insertion attack.

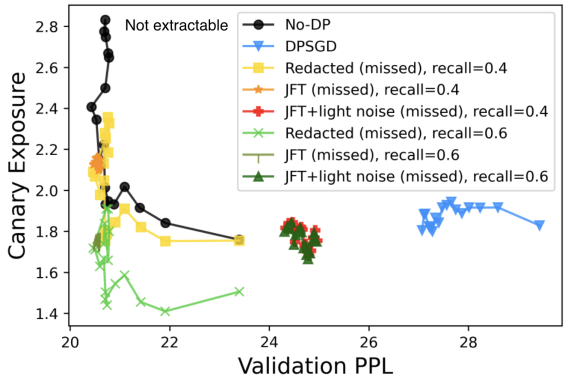

Figure 4: Exposure for different models when we insert ten different canaries.