Analyzing Information Leakage of Updates to Natural Language Models

Santiago Zanella-Béguelin
santiago@microsoft.com
Microsoft

Lukas Wutschitz
luwutsch@microsoft.com
Microsoft

Shruti Tople
shruti.tople@microsoft.com
Microsoft

Victor Rühle
virueh@microsoft.com
Microsoft

Andrew Paverd
andrew.paverd@microsoft.com
Microsoft

Olga Ohrimenko*
oohrimenko@unimelb.edu.au
University of Melbourne

Boris Köpf
boris.koepf@microsoft.com
Microsoft

Marc Brockschmidt
mabrocks@microsoft.com
Microsoft

ABSTRACT

To continuously improve quality and reflect changes in data, machine learning applications have to regularly retrain and update their core models. We show that a differential analysis of language model snapshots before and after an update can reveal a surprising amount of detailed information about changes in the training data. We propose two new metrics—*differential score* and *differential rank*—for analyzing the leakage due to updates of natural language models. We perform leakage analysis using these metrics across models trained on several different datasets using different methods and configurations. We discuss the privacy implications of our findings, propose mitigation strategies and evaluate their effect.

CCS CONCEPTS

• **Security and privacy → Software and application security**; • **Computing methodologies → Machine learning**; **Natural language generation**.

KEYWORDS

machine learning, privacy, natural language, neural networks

ACM Reference Format:
Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. 2020. Analyzing Information Leakage of Updates to Natural Language Models. In *2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20), November 9–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3372297.3417880

*Work done in part while at Microsoft.

CCS '20, November 9–13, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7089-9/20/11…$15.00
https://doi.org/10.1145/3372297.3417880

1 INTRODUCTION

Over the last few years, deep learning has made sufficient progress to be integrated into intelligent, user-facing systems, which means that machine learning models are now part of the software development lifecycle. As part of this cycle, models are regularly updated to accommodate three different scenarios:

- *data update*, to improve performance when new and more data becomes available;
- *data specialization*, to fine-tune a model on a specific dataset, or to handle distributional shift as usage patterns change; or
- *data deletion*, to respect requests for removal of users' data.

Motivated by these scenarios, we study privacy implications for text data that is added (or removed) during retraining of generative natural language models (LMs). Specifically, we consider an adversary with access to multiple snapshots of a model and wishes to learn information about differences in the data used to train them. This threat model is motivated by the combination of three factors: (1) the current trend to fine-tune pretrained public high-capacity LMs to smaller private datasets; (2) the established ability of such LMs to memorize out-of-distribution training samples [6]; and (3) the widespread deployment of LMs to end-user systems (e.g., predictive keyboards on smartphones), allowing adversaries to analyze them in detail. For the informed reader, we discuss the relationship between this threat model and other attacks against privacy and defenses like differential privacy later on in Section 2.2.

We show that data that is added or removed between model updates can be extracted in this threat model, having severe implications for deploying machine learning models trained on private data. Some of the implications are counter-intuitive: for example, honoring a request to remove a user's data (as per GDPR) from the training corpus can mean that their data becomes exposed by releasing an updated model trained without it. Similarly, fine-tuning a public snapshot of a high-capacity model (e.g., BERT [9] or GPT-2 [23]) with data from a single organization exposes this additional data to anyone with access to both the fine-tuned model and the original public model (e.g., employees of this organization).

In order to extract information about the difference in the data used to train two language models, we develop a novel notion of *differential score*. The differential score of a token sequence captures

the difference between the probabilities assigned to it by the two models. The intuition is that sequences with higher differential scores are likely to have been added during model updates. We devise an algorithm based on beam search to efficiently identify such token sequences, even if the individual models assign low probability to them. This allows us to recover information about the difference between the datasets used for training *without* any background knowledge of their contents or distribution.

When given *some* background knowledge, the advantage of having access to two model snapshots becomes crisper. For example, we train a recurrent neural network (RNN) on 20M tokens of general Reddit comments, and update it by retraining it on these comments plus 25K tokens from 940 messages of the `talk.politics.mideast` newsgroup. When prompted with the word "Turkey", our algorithm produces "Turkey searched an American plane" as the 2nd most likely result, although this phrase occurs only 6 times in newsgroup messages and none in Reddit comments (i.e., $< 0.000002\%$ of the training data). An equivalent search using only the updated network does not produce this sentence among the top 10,000 results; it would take the longer prompt "Turkey searched an" for this phrase to surface to the top 100 results.

We use differential score to experimentally study the effect of updates in the three scenarios mentioned above. As a proxy for the update dataset, we use synthetically generated sentences (or *canaries*) and real-world sentences from newsgroup messages. Using both canaries and real-world data, we analyze the effect on attacks recovering information from the update dataset of (1) different training types for updates, ranging from retraining a model from scratch with an updated dataset to fine-tuning as is common for modern high-capacity language models; (2) the proportion of private and public data used for the update; and (3) an adversary's background knowledge. For robustness, we consider datasets of different sizes on both RNNs as well as modern transformer architectures.

Summary of Contributions. We present the first systematic study of the privacy implications of releasing snapshots of language models trained on overlapping data. Our results validate that model updates pose a substantial risk to content added to or removed from training data in terms of information leakage. Our key findings are:

- By comparing two models, an adversary can extract specific sentences or fragments of discourse from the difference between the data used to train them. This does not require any information about the training data or the model architecture and is possible even when the change to the data is as small as 0.0001% of the original dataset. Smaller changes become exposed when given partial knowledge about the data.
- We show that analyzing two model snapshots reveals substantially more about the data that was added or removed than considering only a single snapshot at a time, as in [6].
- Adding or removing additional non-sensitive training data between model updates is not a reliable mitigation.
- Training with differential privacy mitigates the attack, but incurs substantial computational cost and reduces the utility of the trained models.
- Restricting access to the model and only outputting a subset of prediction results is a promising mitigation as it reduces the effectiveness of our attack without reducing utility of the model.

These findings apply to models fine-tuned on a smaller dataset, as well as models retrained on the union of original and new data.

Structure of the Paper. We provide background on language models and describe our adversary model and attack scenarios in the next section. We define the notion of differential score and describe how to efficiently approximate it in Section 3. In Section 4 we describe our experiments to analyze the effect of different factors on leakage. In Section 5 we investigate the source of leakage in model updates, e.g., by comparing with leakage from access to only a single model. Finally, we consider mitigation strategies in Section 6, before describing related work and concluding.

2 PRELIMINARIES

2.1 Generative Language Models

We consider machine learning models capable of generating natural language. These models are used in a variety of applications, including automatic caption generation, language translation, and next-word prediction. Generative language models usually operate on a fixed set of known tokens T (often referred to as the model's *vocabulary*) and are *autoregressive*, modeling the probability $p(t_1 \dots t_n)$ of a sequence of tokens $t_1 \dots t_n \in T^n$ as the product of the per-token probabilities conditional on their prefix $p(t_i \mid t_1 \dots t_{i-1})$, i.e.,

$$p(t_1 \dots t_n) = \prod_{1 \le i \le n} p(t_i \mid t_1 \dots t_{i-1}) \,.$$

Training an autoregressive generative language model M requires learning a function (which we also refer to as M) that maps token sequences of arbitrary length to a probability distribution over the vocabulary T, modeling the likelihood of each token to appear next. We use $M(t_{<i})$ to denote the probability distribution over tokens computed by model M after reading the sequence $t_1 \dots t_{i-1} \in T^*$, and $M(t_{<i})(t_i)$ to denote the probability of a specific token t_i.

Given such a model M, a simple predictive screen keyboard can be implemented by feeding M the words typed so far (e.g., from the start of the current sentence) and displaying the, say, three most likely tokens as one-tap options to the user.

A variety of different architectures exist for the generation of natural language using machine learning models. The most prominent are Recurrent Neural Networks (RNNs) using Long Short-Term Memory [17] cells (or variants thereof) and the more recent Transformers [23, 30]. These architectures differ substantially in how they implement the modeling of the per-token probability distribution, but as our experiments show, they behave nearly identically for the purposes of our analysis.

Given a model architecture, a dataset $D \subseteq T^*$ is required as training data to obtain a concrete model. We write M_D to emphasize that a model was trained on a dataset D. Throughout the paper, we use the standard measure of *perplexity* $perp_M(t_1 \dots t_n) = p_M(t_1 \dots t_n)^{\frac{-1}{n}}$ of a model M on test data $t_1 \dots t_n$, using the probability $p_M(t_1 \dots t_n)$ assigned to the sequence by model M. Unlike the more familiar accuracy, which only captures the correctness of the most probable choice, this metric captures models being "almost right." Intuitively, perplexity can be thought as how "surprised" a model is by a next-word choice, and hence, lower perplexity values indicate a better match between data and model.

2.2 Adversary Model and Goals

Language models are regularly *updated* for a variety of reasons, either by adding and/or removing data from the training set. We use the term *model update* to refer to any update in the parameters of the model caused by training on different data. This is distinct from an update to the model architecture, which changes the number or use of parameters. Each update creates a new version of the model, which we refer to as a *snapshot*.

We consider an adversary that has concurrent query access to two snapshots, M_D and $M_{D'}$, of a language model trained on datasets D and D' respectively, where $D \subsetneq D'$. We write M, M' as shorthand for M_D, $M_{D'}$. The adversary can query the snapshots with any sequence $s \in T^*$ and observe the corresponding probability distributions $M(s)$ and $M'(s)$. The adversary's goal is to infer information about training data points in $D' \setminus D$, the difference between D and D'. In the best case, an adversary would recover exact training points. We refer to an adversary who has access to two snapshots of the model as *a snapshot attacker*.

Relationship to other attacks on training data. Snapshot attacks are *reconstruction attacks* [24] against the updated model, as the goal is to recover data points in the dataset used for the update, given the original model as auxiliary information.

The goal of *membership inference attacks* [25, 26] is weaker in that they only aim to determine whether a given point was present in the dataset used to train a model. However, the differential score of a phrase (which we use for reconstruction) can also serve as a signal for inferring membership in the update dataset. We leave an evaluation of this approach to future work.

Finally, *model inversion attacks* [12, 13] repurpose a model to work *backwards*, inferring unknown attributes of individuals given known attributes and a target prediction. Individuals need not be present in the training data, and results are aggregate statistics rather than information about specific training points. See Section 7 for a more in-depth discussion of related attacks.

Relationship to differential privacy. Differential privacy [11] guarantees that a model does not leak significant information about any specific training point. A differentially private model also guarantees *group* privacy, with a bound on the contribution of a group of training points that degrades linearly with the group size. A differentially private model that provides meaningful protection for a group of $|D' \setminus D|$ training points would hence protect against snapshot attacks on $M_D, M_{D'}$. However, this also implies that $M_{D'}$ cannot be significantly more useful (e.g. more accurate) than M_D. Our experiments in Section 6 confirm this intuition, and show that a large privacy budget is needed for the updated model to gain in utility, so that in practice differential privacy provides an empirical mitigation rather than a strong formal guarantee.

2.3 Analysis Scenarios

To guide our analysis, we focus on three concrete scenarios in which an adversary can gain concurrent access to two (or more) snapshots of a language model.

Data Updates. Many applications require language models that reflect recent patterns in language use. For example, a predictive keyboard on a mobile device requires regular updates to suggest terms that have become more common recently (e.g., following news trends or internet memes). To achieve this, vendors often regularly retrain an (otherwise unchanged) model on an updated dataset, for example by simply adding more recent data to the training dataset. In such cases, an adversary can easily gain access to two snapshots M_D and $M_{D'}$ with $D \subsetneq D'$ and may be interested in learning details about the update $D' \setminus D$. We show that we can extract entire sentences from this difference by comparing M_D and $M_{D'}$, revealing not only aggregate user behavior, but specific conversations.

Data Specialization. Some applications with little task-specific data build on top of generic, pretrained high-capacity language models such as GPT-2 [23]. In such settings, training starts from the pretrained model, but then uses a significantly smaller private dataset. As an example, an organization could simply use a publicly available off-the-shelf language model to create an email authoring autocompletion system. However, by additionally training the model with some historical email data, it can be adapted to organization-specific terms, acronyms and concepts. In such a scenario, if an adversary can gain access to the specialized model M', they can easily also obtain the (publicly available) model M used as a basis. We show that by treating these as different snapshots of the same model, the adversary can extract parts of the private dataset used for specialization.

User Data Deletion. Art. 17 of GDPR [29] Right to erasure ("right to be forgotten") gives data owners the right to request erasure of their personal data from a party who has collected and processed it. Language models trained on emails, text messages, or other user-generated content may contain personal information that a user can request to delete. The data collector would be required to delete the user's data and retrain any models in which it had been used. In many cases, these models may have already been released either to the public or to other users via services provided by the data collector (e.g., text prediction and auto-correct services in text editors and mobile keyboards).

This scenario falls into our adversary setting, albeit in reverse chronological order. Here the dataset D' contains the data that will be deleted, whilst D does not (i.e., the difference $D' \setminus D$ represents the user's data). With access to M_D and $M_{D'}$, the attacker can attempt to infer the user's data. Even if the retrained model overwrites the old model, it may not be possible to erase all instances of the old model simultaneously. For example, some users may be slow to download the new version or the old model may have been copied by other parties.

Naturally, this scenario can be extended to other settings where data is deleted between model updates. This scenario raises an interesting question on whether deletion of data is in the user's best interest or if it makes their data more susceptible to leakage.

3 NEW METRICS

We introduce two metrics called differential rank and differential score to analyze data exposure between two snapshots of a generative language model.

3.1 Differential Score and Differential Rank

We aim to identify token sequences whose probability differs most between models M and M'. Intuitively, such sequences are most likely to be related to the differences between their corresponding training datasets D and D'.

To capture this notion formally, we define the *differential score* (DS) of token sequences, which is simply the sum of the differences of (contextualized) per-token probabilities. We also define a *relative* variant $\widetilde{DS}$ based on the relative change in probabilities, which we found to be more robust w.r.t. the *noise* introduced by different random initializations of the models M and M'.

Definition 3.1. Given two language models M, M' and a token sequence $t_1 \dots t_n \in T^*$, we define the *differential score* of a token as the increase in its probability and the *relative differential score* as the relative increase in its probability. We lift these concepts to token sequences by defining

$$DS_M^{M'}(t_1 \dots t_n) = \sum_{i=1}^{n} M'(t_{<i})(t_i) - M(t_{<i})(t_i)\,,$$

$$\widetilde{DS}_M^{M'}(t_1 \dots t_n) = \sum_{i=1}^{n} \frac{M'(t_{<i})(t_i) - M(t_{<i})(t_i)}{M(t_{<i})(t_i)}\,.$$

The differential score of a token sequence is best interpreted relative to that of other token sequences. This motivates ranking sequences according to their differential score.

Definition 3.2. We define the *differential rank* $DR(s)$ of $s \in T^*$ as the number of token sequences of length $|s|$ with differential score higher than s.

$$DR(s) = \left| \left\{ s' \in T^{|s|} \,\middle|\, DS_M^{M'}(s') > DS_M^{M'}(s) \right\} \right|.$$

The lower the differential rank of a sequence, the more the sequence is exposed by a model update, with the most exposed sequence having rank 0.

3.2 Approximating Differential Rank

Computing the differential rank $DR(s)$ of a sequence s of length $|s| = n$ requires searching a space of size $|T|^n$. To avoid exponential blow-up, we rely on Algorithm 1, which approximates the differential rank based on *beam search*.

At iteration i, the algorithm maintains a set S of k (called the *beam width*) candidate sequences of length i together with their differential scores. The algorithm iterates over all $k \cdot |T|$ single-token extensions of these sequences, computes their differential scores, and keeps the k highest-scoring sequences of length $i+1$ for the next step. Eventually, the search completes and returns the set S.

Algorithm 1 returns a set of token sequences s and their differential score r. With this we can approximate the differential rank $DR(s)$ by the number of token sequences *in* S with differential score higher than s. For large enough beam widths this yields the true rank of s. For smaller widths, the result is a *lower bound* on $DR(s)$, as a search may miss sequences with higher differential score.

Proposition 3.3. *If Algorithm 1 returns a set*

$$S = \{(s_1, r_1), \dots, (s_k, r_k)\} \;\; \textit{with } r_1 \geq \dots \geq r_k\,,$$

then $DS_M^{M'}(s_i) = r_i$ *and* $DR(s_i) \geq i - 1$.

Algorithm 1 Beam search for Differential Rank

In: M, M'=models, T=tokens, k=beam width, n=length
Out: S=set of (n-gram, DS) pairs

```
S ← {(ϵ, 0)}                                  ▷ Initialize with empty sequence ϵ
for i = 1 ... n do
    S' ← {(s ∘ t, r + DS_M^{M'}(s)(t)) | (s, r) ∈ S, t ∈ T}
    S ← take(k, S')                           ▷ Take top k items from S'
return S = {(s_1, r_1), ..., (s_k, r_k)} such that r_1 ≥ ··· ≥ r_k
```

Optimizing for Speed. The beam width k governs the trade-off between computational cost and the precision of the approximation. In experiments, we found that shrinking the beam width as the search progresses speeds up the search considerably without compromising on the quality of results. Typically, we use a beam width $|T|$, which we halve at each iteration. That is, we consider $|T|/2$ candidate phrases of length two, $|T|/4$ sequences of length three, and so on.

Optimizing for Diversity. Since the sequences returned by vanilla beam search typically share a common prefix, we rely on *group beam search* as a technique for increasing diversity: we split the initial $|T|$ one-token sequences into multiple groups according to their differential score, and run parallel beam searches extending each of the groups independently. See [31] for more sophisticated techniques for increasing diversity.

4 LEAKAGE ANALYSIS

We use our new metrics to perform leakage analyses for various datasets across various model update scenarios. We first describe our benchmark datasets with their model configurations and the model training scenarios we consider. Then, we discuss research questions relevant to the analysis scenarios described in Section 2.3. We then show experiments investigating these questions in detail, first using synthetically generated canaries as a proxy for updates where we can precisely control the differences between the datasets used to create model snapshots, and then in a realistic setting, in which we use a set of standard real-world datasets.

4.1 Datasets and Models

We consider three datasets of different size and complexity, matched with standard model architectures whose capacity we adapted to the data size and implemented in TensorFlow.[1]

Concretely, we use the Penn Treebank [20] (PTB) dataset as a representative of low-data scenarios, as the standard training dataset has only around 900,000 tokens and a vocabulary size of 10,000. As the corresponding model, we use a two-layer recurrent neural network using LSTM cells with 200-dimensional embeddings and hidden states and no additional regularization (this corresponds to the *small* configuration of Zaremba et al. [33]).

Second, we use a dataset of Reddit comments with 20 million tokens overall, of which we split off 5% as validation set. We use a vocabulary size of 10,000. We rely on two different model configurations for this dataset, which allows us to understand the impact of model size on information leakage using DR as a metric.

[1] Source code and tools available at: https://github.com/microsoft/language-privacy

(1) a one-layer RNN using an LSTM cell with 512-dimensional hidden states and 160-dimensional embeddings. We employ dropout on inputs and outputs with a keep rate of 0.9 as regularizer. These parameters were chosen in line with a neural language model suitable for next-word recommendations on resource-constrained mobile devices.
(2) a model based on the Transformer architecture [30] (more concretely, using the BERT [9] codebase) with four layers of six attention heads, each with a hidden dimension of 192.

Finally, we use the Wikitext-103 dataset [22] with 103 million training tokens as a representative of a big data regime, using a vocabulary size of 20,000. As the model, we employ a two-layer RNN with 512-dimensional LSTM cells and token embedding size 512 and dropout on inputs and outputs with a keep rate of 0.9 as regularizer. We combined this large dataset with this (relatively low-capacity) model to test if our results still hold on datasets that clearly require more model capacity than is available.

All models and their training are following standard best practices for generative language models and represent common (simple) baselines used in experiments on the used datasets. This can be seen in the perplexity of the trained models on the held-out test data, shown in Table 1, which is in line with common test results.

4.2 Implementing Model Updates

Updated models can be created using different techniques, with different applicability to the usage and analysis scenarios discussed in Section 2.3.

Retraining. Given an updated dataset D', a fresh model snapshot M' can be obtained by simply training a fresh model from scratch, which we refer to as *retraining*. This also involves a fresh (random) initialization of the model parameters, and in practice, retraining repeatedly on the *same* dataset will yield slightly different models. *Data deletion* requires updating a model to eliminate the influence of some training data points at the request of the data owner. This can be done by retraining a model after pruning the data or, equivalently, using techniques with lower computational cost [5, 14].

Continued Training. In this approach, a fresh model snapshot M' is obtained by taking an existing model M and continuing training it on additional data. This is the core of the *data specialization* scenario and sometimes also used in *data update* scenarios to avoid the computational cost of training on a large dataset from scratch.

4.3 Research Questions

With the training techniques outlined for different model update scenarios, we consider four research questions in our experiments.

RQ0: Can an attacker learn private information from model updates? Here we address the basic question of whether private data used to update a model can be leaked in our adversarial setting and how. We first answer this question by using differential score to *find* information about private sequences used in a model update. We then investigate the influence of other parameters of the system on the differential score in more detail.

RQ1: How does masking private data with additional non-sensitive data (D_{extra}) affect leakage? This is particularly important for the user deletion scenario, for which we need to answer if it is possible to safely remove data of a single user, or if such dataset changes need to be hidden among other substantial changes. Concretely, we analyze whether including a large enough additional dataset D_{extra} in an update can prevent leakage of information about the rest of the data used. D_{extra} can be any dataset which is either available publicly or is non-sensitive from the point of view of the model provider or users.

RQ2: How do retraining and continued training differ with respect to information leakage? In the continued training approach, the parameters of a previously trained model M_D are updated based only on new data $D' \setminus D$. In contrast, in the retraining strategy parameters are updated using all data in D'. The most recent updates to model parameters depend only on new data in the continuing training case, whereas they depend on the whole training data D' when retraining a model from scratch. We analyze the effect of this seemingly more pronounced dependence.

RQ3: How is leakage affected by an adversary's background knowledge? Prior attacks on language models assume that the adversary has background knowledge about the context in which a secret appears. We analyze the effect of such knowledge for inferring private data from model updates.

4.4 Results with Canaries

We create a number of canary phrases—grammatically correct phrases that do not appear in the original dataset—that serve as a proxy for private data that the adversary is trying to extract. We consider different word frequency characteristics to control the influence on the used vocabulary. Specifically, we fix the length of the canary phrase to 5, choose a valid phrase structure (e.g., Subject, Verb, Adverb, Compound Object), and instantiate each placeholder with a token in a dataset vocabulary. We create canaries in which frequencies of tokens are *all low* (all tokens are from the least frequent quintile of words), *mixed* (one token from each quintile), *increasing from low to high*, and *decreasing from high to low*. For example, the *mixed* phrase across all the datasets is "NASA used deadly carbon devices", and the *all low* phrase for PTB is "nurses nervously trusted incompetent graduates". As the vocabularies differ between the different datasets, the canaries are in general dataset-dependent. We vary the amount of *private data*, C, by inserting a canary phrase s a number of times proportional to the number of tokens in the training corpus:

(1) For PTB, we consider $k \in \{10, 50, 100\}$ canary insertions (corresponding to 1 canary token in 18K training tokens, 1 in 3.6K, and 1 in 1.8K).

(2) For the Reddit dataset, we use $k \in \{5, 50, 500\}$ (corresponding to 1 in 1M, 1 in 100K, 1 in 10K).

(3) For the Wikitext-103 data, we use $k \in \{20, 100\}$ (corresponding to 1 in 1M, 1 in 200K).

We train the model M on D and the model M' on D with k copies of the canary s. We then compute the differential rank of the canaries for different values of k.

RQ0: Can an attacker learn private information from model updates? We use our differential score based beam search (Algorithm 1)

Table 1: Differential score (DS) for different datasets, model architectures, canaries, and insertion frequencies. White cells represent a differential rank (DR) of 0 (as approximated by beam search), and gray cells represent $DR > 1000$.

Dataset	Penn Treebank			Reddit						Wikitext-103	
Model Type (Perplexity)	RNN (120.90)			RNN (79.63)			Transformer (69.29)			RNN (48.59)	
Canary Token Freq.	1:18K	1:3.6K	1:1.8K	1:1M	1:100K	1:10K	1:1M	1:100K	1:10K	1:1M	1:200K
All Low	3.40	3.94	3.97	2.83	3.91	3.96	3.22	3.97	3.99	1.39	3.81
Low to High	3.52	3.85	3.97	0.42	3.66	3.98	0.25	3.66	3.97	0.07	3.21
Mixed	3.02	3.61	3.90	0.23	3.04	3.92	0.39	3.25	3.96	0.25	3.02
High to Low	1.96	2.83	3.46	0.74	1.59	2.89	0.18	1.87	3.10	0.08	1.22

Table 2: Differential Score ($DS_M^{M'}$) of the mixed frequency canary phrase for the Reddit (RNN) model using different update techniques. Model M is trained on D_{orig}. For the *Retraining* column, M' is trained on $D_{orig} \cup D_{extra} \cup C$ starting from random initial parameters. For the *Cont'd Training 1* column, M' is trained on $D_{extra} \cup C$ starting from M. For the *Cont'd Training 2* column, we first train a model $\tilde{M}$ on $D_{extra} \cup C$ starting from M, and then train model M' from $\tilde{M}$ using additional public data D'_{extra}. A white cell background means that the differential rank DR (as approximated by our beam search) of the phrase is 0, gray cell background means that DR is >1000.

	Retraining				Continued Training 1			Continued Training 2
$\lvert D_{extra}\rvert / \lvert D_{orig}\rvert$	0%	20%	50%	100%	20%	50%	100%	100%
1:1M	0.23	0.224	0.223	0.229	0.52	0.34	0.46	0.01
1:100K	3.04	3.032	3.031	3.038	3.56	3.25	3.27	0.26

to extract canary phrases that correspond to the change in training data between M and M'. The results of varying the number of inserted canaries are summarized in Table 1. We highlight the following findings:

- **For most combinations of k and types of canaries, we successfully recover the canary.** This is indicated by the cells with white background, where the canary phrase has the *maximum differential score* among all token sequences found by our beam search, i.e., it ranks first.
- The signal for extraction is strong even when the inserted canaries account for only 0.0001% of the tokens in the dataset. This is visible in the first row of Table 1 where differential scores approach 4 — close to the upper bound of 5 for 5-token canaries.
- Private phrases that occur more often in the training data are more exposed via a model update, as expected. This is visible in the monotonic growth of the differential score of canaries with the number of insertions.
- Phrases composed of rare words are more easily extracted, as seen in the high differential score of canaries constructed from low-frequency tokens. In contrast, canaries with descending token frequencies tolerate much higher number of insertions before being exposed. This is expected, as our beam search is biased towards finding high-scoring prefixes.
- Access to two model snapshots reveals substantially more than access to a single snapshot. For comparison, we successfully extract a 5-token canary inserted 1 in 200k times (i.e. inserting one token every 1M tokens) from two snapshots of an LSTM-based generative model without additional knowledge. In contrast, [6, Section 6.1] reports failing to extract the middle token of a 5-token canary inserted 1 in 100k times from a similar LSTM-based model when given the first and last two words.

RQ1: Effect of amount of public vs. private data. In Table 2 we vary the amount of *public data* by partitioning the dataset D into $D_{orig} \uplus D_{extra}$ such that the latter is 20%, 50%, or 100% of the size of D_{orig} (the 0% column is identical to Table 1). The retraining column shows that $DS_M^{M'}$ does not change significantly across the different dataset splits. That is, canaries can be extracted from the trained model even when they are contained in a substantially larger dataset extension. Hence, the amount of public data in the update does not significantly affect the leakage of the private data.

RQ2: Effect of training type. We train a model M on a dataset D_{orig} to convergence, and then continue training M using D_{extra} and the canaries C, obtaining M'. We compare the differential rank of the canaries on the models obtained using continued training with that on the models retrained from scratch (shown in the middle column of Table 2). We observe that in all cases the differential score is higher for continued training than for retraining. As expected, the differential score of the canary phrase decreases as additional extra data is used for fine-tuning.

RQ3: Effect of background knowledge. We evaluate the differential score of suffixes of a canary phrase s assuming knowledge of a prefix. For $i = 1, \ldots, n$ we take the prefix $t_1 \ldots t_{i-1}$ of the canary phrase and compute the differential score r of the token t_i conditional on having read the prefix, i.e., $M'(t_{<i})(t_i) - M(t_{<i})(t_i)$. The relationship between i and r indicates how much knowledge about s is required to expose the remainder of the canary phrase.

Figure 1 depicts the result of this analysis for canaries with high-to-low and all-low token frequencies on the Reddit dataset. Our results show that, while the differential score of the first token without context is close to 0, the score of subsequent tokens quickly grows for all-low canaries, even with a low number of canary

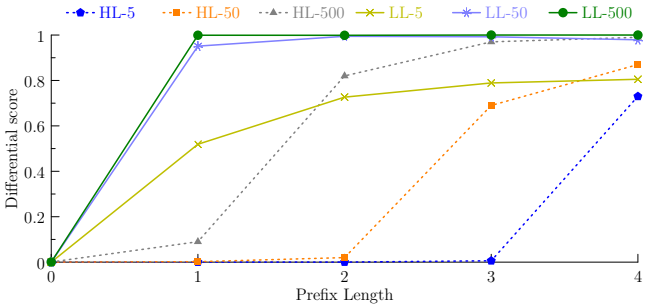


Figure 1: Differential score of tokens in canaries given a prefix for the Reddit dataset. LL-k denotes k canary insertions with all-low token frequencies (solid lines), and HL-k denotes high-to-low token frequencies (dashed lines).

insertions. In contrast, more context is required before the score of high-to-low canaries increases, as the model is less influenced by the small number of additional occurrences of frequent tokens.

This suggests that, even in cases where we fail to extract the canary without additional knowledge, an adversary can use the differential rank to complete a partially known phrase, or confirm that a phrase was used to update the model.

4.5 Results with Real-world Data

We simulate real-world scenarios by sourcing training data from real-world conversations on specific topics, and using it as a proxy for private data included in the training data used in model updates. The adversary's goal is to extract specific phrases occurring in the proxy dataset, or phrases that do not occur literally but nonetheless reveal the topic of conversations.

We mimic the data distribution shift by choosing conversations on topics that are not dominant in the original dataset, so that we can better judge whether phrases extracted using differential score are on-topic and thus represent meaningful leakage of private information. Specifically, we compare models trained only on data from the Reddit dataset against models trained on data from the Reddit dataset plus messages from one of two newsgroups from the 20 Newsgroups dataset [19]:

a) `rec.sport.hockey`, containing around 184K tokens, ≈1% of the original training data; and
b) `talk.politics.mideast`, containing around 430K tokens, ≈2% of the original training data.

We train a model M on the entire Reddit dataset and retrain M' from scratch on the same dataset plus all messages from one of the two newsgroups. For both model architectures (RNNs and Transformer) described in Section 4.1 and each newsgroup, we compute the sequences with highest relative differential score. Since the sequences returned by vanilla beam search typically share a common prefix, we run a group beam search (see Section 3.2) to get a more diverse sample.

RQ0: Can an attacker learn private information from model updates? Tables 3 and 7 (in the Appendix) display the highest-scoring sequences of length 4 in each group of a $\widetilde{DS}$-based 5-group beam search.

The exposed sentences are on-topic w.r.t. the newsgroup included, e.g., the hockey theme dominates the top ranked sequences in Table 3. This suggests that, information about the private data used for the update is leaked. It is noteworthy that these results are obtained assuming a weak adversary that does not require either background knowledge about the dataset distribution or about the information it tries to extract. In contrast, concurrent work on updates of image classification models [24] requires knowledge about the data distribution to train shadow models, while prior work on single language models [6] requires a known prefix for extraction of a secret.

Given some background knowledge in the form of a long enough prefix of a phrase occurring in the private data, we show that the complete phrase can be extracted by a beam search directed by differential score (see Table 5).

RQ1: Effect of amount of public vs. private data. We consider partitions of the Reddit dataset D into D_{orig} and D_{extra} of different relative sizes. For each partition, we train a model M on D_{orig} and a model M' on $D_{orig} \cup D_{extra} \cup N$, where N are all messages from `talk.politics.mideast`. We observe the following:

- For all phrases, the proportion of public data ranging from 5% to 100% used in the update does not significantly affect their relative differential scores, which confirms our findings for canaries.
- The top two phrases resemble canaries in that they occur literally multiple times in the update dataset, which explains their high scores. An exception is `Little resistance was offered`, which appears 12 times in the dataset but still has low score. Other phrases do not occur literally in newsgroup messages, but digest recurrent discussions or contain n-grams that do occur.

RQ2: Effect of training type. We train a model M on D_{orig} to convergence, and then continue training M using $D_{extra} \cup N$ to produce a model M'. To understand the effect of the training type on information leakage, we sample a set of representative phrases and compare their relative differential scores w.r.t. M and M' against their scores w.r.t. M and a model trained on $D \cup N$ from scratch.

The results are shown in Table 4, together with the perplexity decrease after the model update. Retrained models correspond to the *data update* and *data deletion* scenarios and their perplexity drop is greater the more data is used during retraining. Continued training corresponds to the *data specialization* scenario. The perplexity drop in the updated model is greater the larger is the proportion of newsgroup data used in the update, for which the initial model is not specialized.

The last two rows in Table 4 correspond to phrases found by group beam search in the continued training scenario, but that have too low a score to be found when M' is retrained from scratch instead. The converse, i.e., phrases that have low score when continuing training and high score when retraining, seems to occur rarely and less consistently (e.g., `Saudi troops surrounded village`).

For phrases that occur literally in the dataset, the results are in line with those for canaries (see Table 2), with scores decreasing as more data is used during the fine-tuning stage. For other phrases, the results are not as clear-cut. While fine-tuning a model exclusively on private data yields scores that are significantly higher than when retraining a model from scratch, this effect vanishes

Table 3: Top ranked phrases in group beam search for a model updated with `rec.sport.hockey`. For the layperson: Los Angeles Kings, Minnesota North Stars, and Toronto Maple Leaf are National Hockey League teams; Norm Green was the owner of the North Stars; an ice hockey game consists of three periods with overtime to break ties. Capitalization added for emphasis.

RNN		Transformer	
Phrase	$\widetilde{DS}$	Phrase	$\widetilde{DS}$
`Angeles Kings prize pools`	56.42	`Minnesota North Stars playoff`	96.81
`National Hockey League champions`	53.68	`Arsenal Maple Leaf fans`	71.88
`Norm 's advocate is`	39.66	`Overtime no scoring chance`	54.77
`Intention you lecture me`	21.59	`Period 2 power play`	47,85
`Covering yourself basically means`	21.41	`Penalty shot playoff results`	42.63

Table 4: Relative differential score of phrases found by beam search when retraining from scratch and continuing training from a previous model. The results are for RNN models trained on partitions of the Reddit dataset with N = `talk.politics.mideast`. Cells for which continued training yields a higher score than retraining appear in bold font. Capitalization added for emphasis.

Phrase (# of occurrences in N)	Retraining					Continued Training				
$\lvert D_{extra}\rvert / \lvert D_{orig}\rvert$	0%	5%	10%	20%	100%	0%	5%	10%	20%	100%
Perplexity decrease	0.79	1.17	2.45	3.82	11.82	73.97	18.45	10.29	6.08	8.28
`Center for Policy Research` (93)	99.77	101.38	97.11	98.65	91.53	**276.98**	**198.69**	**150.56**	**122.25**	**117.54**
`Troops surrounded village after` (12)	44.50	44.50	44.50	44.41	44.54	**173.95**	**47.38**	19.48	7.81	35.56
`Partition of northern Israel` (0)	27.61	16.81	38.48	26.10	38.76	**68.98**	16.48	12.47	22.93	18.82
`West Bank peace talks` (0)	25.68	25.64	25.69	25.71	25.75	**71.54**	24.38	**28.60**	16.91	4.62
`Spiritual and political leaders` (0)	25.23	25.98	17.04	24.21	23.47	**126.92**	14.91	10.00	3.44	11.05
`Saudi troops surrounded village` (0)	24.31	24.31	24.31	24.31	24.30	5.05	**44.58**	4.29	7.29	**63.84**
`Arab governments invaded Turkey` (0)	22.59	22.62	22.80	22.78	22.80	**24.01**	15.58	7.08	18.12	11.90
`Little resistance was offered` (12)	22.24	22.09	25.12	22.34	25.59	**215.16**	**25.02**	2.00	3.30	5.64
`Buffer zone aimed at protecting` (0)	4.00	4.47	5.30	5.25	5.69	**57.29**	**69.76**	**18.92**	**14.50**	**22.25**
`Capital letters racial discrimination` (0)	3.76	3.32	3.40	3.60	3.84	**94.60**	**52.74**	**39.11**	**11.22**	3.45

as more additional data is used; in some cases continued training yields scores lower than when retraining a model on the same data.

RQ3: Effect of background knowledge. An adversary wishing to extract information about the dataset used to update a language model may direct a search using as *prompt* a known prefix from the dataset. We study how long this prefix needs to be to recover the rest of phrase.

We consider a RNN model M trained on the full Reddit dataset and a model M' trained on the union of the full Reddit dataset and all messages of the `talk.politics.mideast` newsgroup. We sample 4 phrases in newsgroup messages beginning with the name of a Middle Eastern country and containing only tokens in the model vocabulary. We believe it is feasible for the adversary to guess these prefixes from the description of the newsgroup or the geopolitical context. For each phrase s and $i = 0, \ldots, |s|-1$ we run a $\widetilde{DS}$-based beam search for phrases of the same length with constant beam width 10,000 and 100 groups starting from $s_1 \ldots s_i$. Table 5 shows the rank of s among the search results (or ∞ if absent).

We observe a correlation between the score of a phrase and the minimum prefix sufficient to recover it. However, a dip in the score of two consecutive tokens is much more consequential: a common word like `the`, which has a similar distribution in the original and private datasets, contributes little to the score of a phrase and is unlikely to be picked up as a candidate extension in a beam search. Recovering from this requires additional heuristics or a more expensive search, using wider beams or looking more than one token ahead to better approximate the true rank of a phrase.

5 CHARACTERIZING THE SOURCE OF LEAKAGE

Prior work has primarily studied information leakage when an attacker has only access to a single model snapshot. Here, we first analyze how much our analysis gains from having access to two model snapshots, and then consider the influence of common causes of leakage in the single-model case. The central ones are *overfitting* [32] to the training data, and *unintended memorization* [6] of data items that is independent of the distribution to be learned.

RQ4: How important is access to a second model snapshot? We want to analyze how much leakage of sensitive information is increased when having access to two model snapshots M_D, $M_{D'}$ in contrast to having only access to a single model $M_{D'}$. This is a challenging analysis in a realistic setting, due to the size of the data and the lack of an easily computable metric for information leakage. Concretely, we want to show that the data we can extract using the differential analysis of M_D and $M_{D'}$ is (a) more likely to be part of D' than of D, (b) not very common in D', and (c) that (a) and (b) are more true for the results of the differential analysis than for the analysis of $M_{D'}$ alone.

We quantify how likely a given sentence is to be a part of a dataset using a simpler, well-understood model of natural language data,

Table 5: Results of beam searches for different prefix lengths. A rank of 0 means that the search recovers the complete phrase. Due to the heuristic nature of the search the rank reported may be lower than the true rank of s. Conversely, a beam search may not encounter s at all despite having lower rank than most phrases encountered. For instance, this occurs for `Turkey searched an American plane`, where all but 7 search results with no prompt have higher rank (lower score).

			Prefix length i					
Phrase s	# of occurrences	$\widetilde{DS}(s)$	0	1	2	3	4	5
`Turkey searched an American plane`	6	82.96	∞	1	1	0	0	–
`Israel allows freedom of religion`	3	24.44	∞	∞	788	55	0	–
`Iraq with an elected government`	2	23.75	∞	∞	∞	4	0	–
`Israel sealed off the occupied lands`	2	6.48	∞	∞	∞	∞	3442	2

namely an n-gram model. n-gram models define the probability of a token t_{n+1} appearing after a sequence of tokens $t_1 \dots t_n$ as the number of times $t_1 \dots t_n t_{n+1}$ appeared in the dataset divided by the number of times $t_1 \dots t_n$ appeared.

In our experiments, we use the perplexity of 3-gram models trained on D (resp. N) to capture how likely a given extracted sentence is part of the dataset D (resp. N). We compare these perplexity values for sequences extracted using group beam search from the models M_D (resp. $M_{D'}$) and for sequences extracted using our differential rank-based search, following the setup of Section 4.5. Concretely, we used the entire Reddit comment data as dataset D, and the messages N from `talk.politics.mideast` as data update. We are concerned with information an attacker can gain about the contents of N.

Figure 2a shows the results of our analysis when we train $M_{D'}$ on $D' = D \cup N$ from scratch. Points above the main diagonal are closer in distribution to the (private) data update N than to the base data D. This shows that our attack extracts sequences using differential score (represented by red crosses) that are more likely to be part of N than of D, and that these sequences differ substantially from the sequences obtained by a single-model analysis. In fact, the sequences obtained by single-model analysis for M_D and $M_{D'}$ show little significant difference. Note that the perplexity values $perp_{3\text{-gram}(D)}$ are very high for some of the extracted sentences, as they use combinations of tokens that never appear in the original training dataset D. Similarly, Figure 2b shows the results of this analysis on the scenario in which we obtain $M_{D'}$ by specializing the model M_D by continuing training on the dataset N. While our differential analysis again captures sequences more likely to be part of the updated data N than of the original data D, the single-model analysis now also shows some of this effect.

RQ5: Is leakage due to overfitting or intended memorization? All models are trained using an early-stopping criterion that halts training when the model does not improve on a separate validation set. This effectively rules out overfitting to the training data. Additionally, model training employs regularization strategies such as dropout to further encourage the trained models to generalize to unseen data.

We refer to the model's ability to reproduce verbatim fragments of the training data as *memorization* and call it *intended* if this is necessary to serve its purpose of generating natural language (e.g., a model needs to memorize the token pair "United States", as it is an extremely common combination) and *unintended* otherwise.

In the experimental results in Table 4, we have included the number of times that the phrases with the highest differential scores appear in the update dataset. Since some of these phrases do not appear verbatim, we also measure how close these phrases are to phrases in the original and update datasets. Table 6 shows the Levenshtein distance of extracted phrases from Table 4 to their nearest neighbor in either dataset. Generally, we find closer matches in the update dataset. While "Center for Policy Research" is a clear case of intended memorization, as the name appears many times in email signatures, other phrases appear rarely or never, indicating that our analysis extracts phrases that need not be memorized to serve its purpose. This is further supported by the results in Table 5, where extraction of complete sentences such as "Israel allows freedom of religion" occurring as few as three times in the dataset is possible. Overall, this indicates that intended memorization is unlikely to explain our results.

Unintended memorization may occur for infrequent phrases. However, it cannot alone explain our results, as shown by our success in recovering canaries when using a low-capacity model in a large-data regime (cf. Wikitext-103 column in Table 1), for which the effect of unintended memorization is less pronounced, and evidenced by the large context needed to recover canaries from a single-model analysis [6]. The most likely explanation remains that a differential analysis of two model snapshots amplifies otherwise imperceptible differences in the data used to train them, which would be hard to suppress without hurting a model's performance.

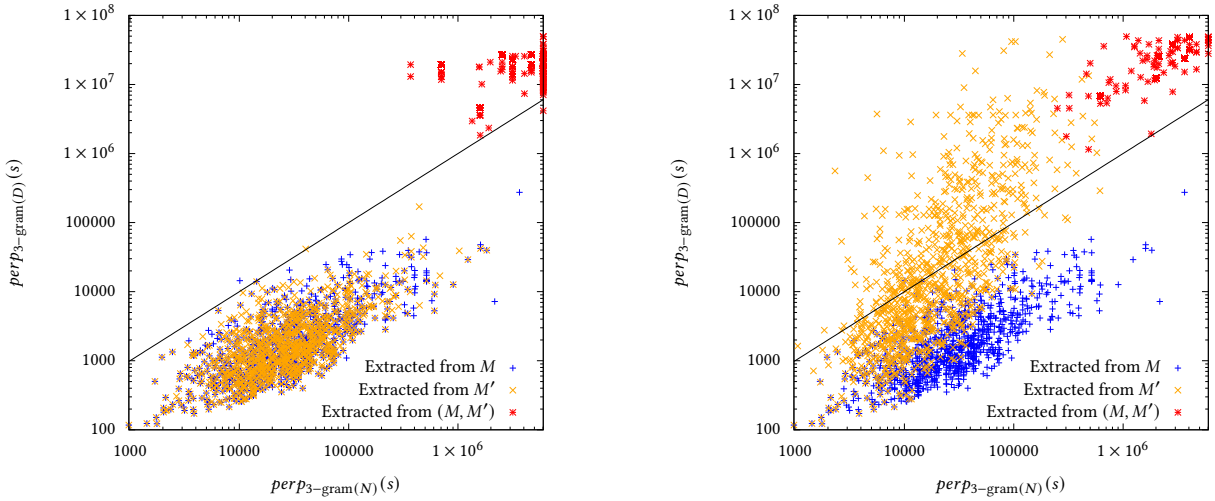
6 MITIGATIONS

In this section, we discuss and analyze three strategies to mitigate information leakage in model updates: (1) Differential Privacy, (2) continued training with public data, and (3) truncating the output of the updated model.

6.1 Mitigation: Differential Privacy

Differential privacy (DP) [11] provides strong guarantees on the amount of information leaked by a released output. Given a computation over records it guarantees a bound on the effect that any input record can have on the output. Formally, F is a (ϵ, δ)-differentially-private computation if for any datasets D and D' that differ in one record and for any subset O of F's range we have

$$\Pr(F(D) \in O) \leq \exp(\epsilon) \cdot \Pr(F(D') \in O) + \delta\,.$$



(a) Re-training from scratch **(b) Continued training**

Figure 2: Sensitivity of extracted content. + depict sentences extracted from M, × from M', and ∗ from (M, M') using Differential Score. Vertical axis depicts the perplexity w.r.t data D, horizontal axis depicts perplexity w.r.t data update N. Points above the diagonal are closer in distribution to the (private) data update N than to the base data D.

Table 6: Quantifying near matches of extracted phrases from RNN models trained on the base Reddit dataset and updated with `talk.politics.mideast`. For each extracted phrase, we compare the Levenshtein distance to its nearest neighbor in the base and update datasets respectively. The updated dataset contains closer matches for all phrases except `west bank peace talks` and `capital letters racial discrimination`, for which there are equally close matches in both datasets.

Extracted phrase	talk.politics.mideast		Reddit	
center for policy research	center for policy research	0	center for instant research	1
troops surrounded village after	troops surrounded village after	0	from the village after	2
partition of northern israel	shelling of northern israel	1	annexation of northern greece	2
west bank peace talks	. no peace talks	2	: stated peace talks	2
spiritual and political leaders	spiritual and political evolutions	1	, and like leaders	2
saudi troops surrounded village	our troops surrounded village	1	" hometown " village	3
arab governments invaded turkey	arab governments are not	2	! or wrap turkey	3
little resistance was offered	little resistance was offered	0	, i was offered	2
buffer zone aimed at protecting	" aimed at protecting	2	's aimed at a	3
capital letters racial discrimination	% of racial discrimination	2	allegory for racial discrimination	2

Differential privacy is a natural candidate for defending against membership-like inferences about data. The exact application of differential privacy for protecting the information in the model update depends on what one wishes to protect w.r.t. the new data: individual sentences in the new data or all information present in the update. For the former, sequence-level privacy can suffice while for the latter group DP can serve as a mitigation technique where the size of the group is proportional to the number of sequences in the update. Recall that an ϵ-DP algorithm F is $k\epsilon$-differentially private for groups of size k [11].

Differential privacy can be achieved in gradient-based optimization computations [1, 4, 28] by clipping the gradient of every record in a batch according to some bound L, then adding noise proportional to L to the sum of the clipped gradients, averaging over the batch size and using this noisy average gradient update during backpropagation.

We evaluate the extent to which DP mitigates attacks considered in this paper by training models on the Penn Treebank (PTB) dataset with canaries with sequence-level differential privacy. We train DP models using the TensorFlow Privacy library [2] for two sets of (ϵ, δ) parameters, $(5, 1 \times 10^{-5})$ and $(111, 1 \times 10^{-5})$, for two datasets: PTB and PTB with 50 insertions of the all-low-frequency canary. We rely on [2] to train models with differentially private stochastic gradient descent using a Gaussian noise mechanism and to compute the overall privacy loss of the training phase. As expected, the performance of models trained with DP degrades, in our case from ≈23% accuracy in predicting the next token on the validation dataset to 11.89% and 13.34% for ϵ values of 5 and 111, respectively.

While the beam search with the parameters of Section 4.4 no longer returns the canary phrase for the DP-trained models, we note that the models have degraded so far that they are essentially only predicting the most common words from each class (e.g., "is" when a verb is required) and thus, the result is unsurprising. We note that the guarantees of sequence-level DP formally do not apply for the case where canary phrases are inserted as multiple sequences, and that ϵ values for our models are high. However, the ϵ-analysis is an upper bound and similar observations about the effectiveness of training with DP with high ϵ were reported by Carlini et al. [6].

We further investigate the effect of DP training on the differential rank of a canary phrase that was inserted 50 times. Instead of using our beam search method to approximate the differential rank, we fully explore the space of subsequences of length two, and find that the *DR* for the two-token prefix of our canary phrase dropped from 0 to 9,458,399 and 849,685 for the models with $\epsilon = 5$ and $\epsilon = 111$ respectively. In addition, we compare the differential score of the whole phrase and observe that it drops from 3.94 for the original model to 4.5×10^{-4} and 2.1×10^{-3} for models with $\epsilon = 5$ and $\epsilon = 111$, respectively. Though our experiment results validate that DP can mitigate the particular attack method considered in this paper for canary phrases, the model degradation is significant. In addition, the computational overhead of per-sequence gradient clipping required by [2] is substantial, making it unsuitable for training high-capacity neural language models on large datasets.

6.2 Mitigation: Two-stage Continued Training

We also consider a possible mitigation strategy where we perform continued training in two stages. For this, we split the dataset into three equal parts D_{orig}, D_{extra} and D'_{extra}. We proceed as in the continued training setting in RQ2, but add a final step in which we train on another dataset after training on the canaries. This resembles a setting where an attacker does not have access to two consecutive snapshots. The rightmost column of Table 2, shows that the differential score of the canary phrase drops substantially after the second training stage. Thus, two or multi-stage continued training, where only the last trained model is released, might be a path toward mitigating leakage of private data.

6.3 Mitigation: Truncating Output

Finally, we analyze the effect of truncating the output of the updated model for each query. Specifically, the adversary still has full access to the original model M but only receives the top k tokens from the updated model M'. This is a slight weakening of our adversary model, but is realizable for some applications. For example, in the *Data Specialization* scenario, the adversary may have full access to the public base model, but can only access the specialized model via an API that truncates the results for each query. In the *Data Update* scenario, even if models are deployed to client devices, it may be possible to enforce this by running the model in a Trusted Execution Environment (TEE), such as Intel SGX [18] or ARM TrustZone [3] on the client device.

To evaluate the impact of this mitigation, we repeat the experiment described in Section 5 and plot only the sentences extracted using differential score (i.e., the 'Snapshot attack') for different values of k. To facilitate comparison, we use the same beam width as in Figures 2a and 2b. As shown in Figure 3, decreasing the value of k brings the extracted sequences closer to the main diagonal, where they have similar likelihood of being drawn from either dataset. Similarly to Figures 2a and 2b, we also observe a difference between re-training from scratch and continued training; for the same value of k, the sentences extracted after continued training are more likely to be private than those extracted after the model is re-trained from scratch. Additionally, if the adversary only has access to the top k outputs of the original model M, this would further reduce the leakage. In applications where this mitigation is realizable, returning only the top k outputs can thus reduce leakage without decreasing the utility of the provided outputs.

7 RELATED WORK

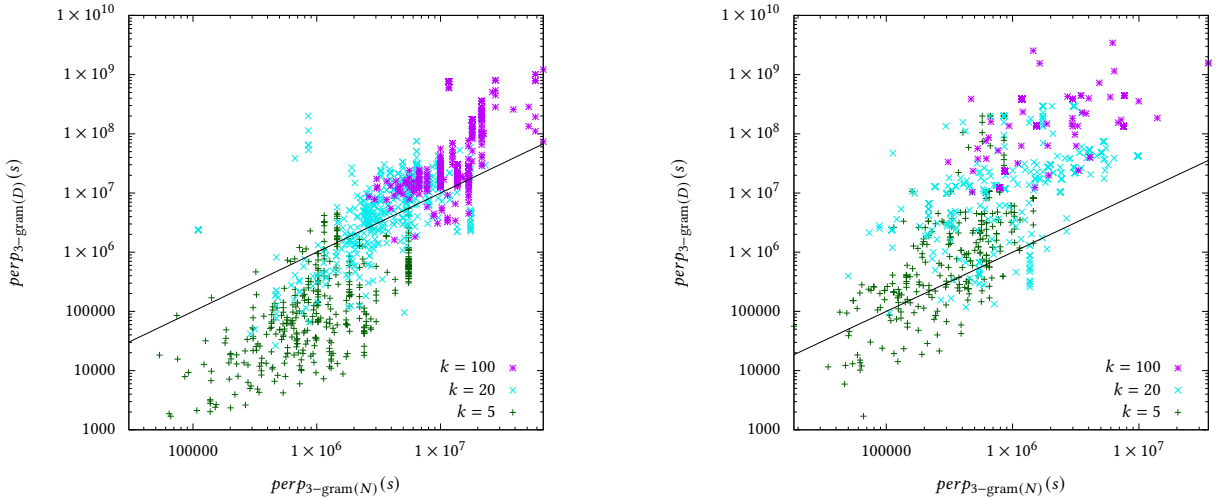
Several works have shown that machine learning models can leak information about training data and proposed defenses for them.

Membership inference attacks. Shokri et al. [26] show that one can identify whether a record belongs to the training dataset of a classification model given black-box access to the model and shadow models trained on data from a similar distribution. Salem et al. [25] demonstrate that similar attacks are effective under weaker adversary models. It would be interesting to study how membership inference based on differential score compares to other techniques [8].

Song and Shmatikov [27] also study sequence-to-sequence language models and show how a user can check if their data has been used for training. In their setting, an auditor needs an auxiliary dataset to train shadow models with the same algorithm as the target model and queries the target model for predictions on a sample of the user's data. The auxiliary dataset does not need to be drawn from the same distribution as the original training data (unlike [26]) and the auditor only observes a list of several top-ranked tokens. In contrast, our approach requires *no* auxiliary dataset, but assumes access to the probability distributions over all tokens from two different model snapshots. From this, we are able to recover full sequences from the differences in training data rather than binary information about data presence. Like them, we find that sequences with infrequent tokens provide a stronger signal to the adversary/auditor.

Reconstruction attacks. These attacks abuse a model to recover specific training points [24]. The attacks we present are a form of reconstruction attacks against an updated model: we recover data points in the dataset used for the update given the original model as auxiliary information.

Carlini et al. [6] is closest to our work, as it also considers information leakage of language models. The authors assess the risk of (unintended) memorization of rare sequences in the training data. They show that canaries inserted into training data can be retrieved from a character-level language model. The key differences to our approach are that 1) we consider a different attack scenario where *an adversary has access to two snapshots of a model*, and 2) our canaries follow the distribution of the data whereas Carlini et al. [6] add a random sequence of numbers in a fixed context into a dataset of financial news articles (e.g., "The random number is ..."), where such phrases are rare. We instead are able to extract canaries *without any context*, even when the canary token frequency in the training dataset is as low as one in a million.



(a) Re-training from scratch **(b) Continued training**

Figure 3: Sentences extracted from (M, M') using Differential Score when the adversary only receives the top k tokens from the updated model M' for each query. The axes have the same meaning as in Figures 2a and 2b.

Salem et al. [24] consider reconstruction of training data that was used to update a model. While their goal is similar to ours, their adversarial model and setup differ: 1) similar to Song and Shmatikov [27] and Shokri et al. [26], their attacker uses shadow models trained on auxiliary data drawn from the same distribution as the target training dataset, while in our setting the attacker has no prior knowledge of this distribution and does not need auxiliary data; 2) the updated model is obtained by fine-tuning the target model with additional data rather than re-training it from scratch on the changed dataset; 3) the focus is on classification models and not on (generative) language models.

Information leakage from updates has also been considered for searchable encryption: an attacker who has control over data in an update to an encrypted database can learn information about its content and previous encrypted searches on it [7].

Model inversion attacks. Fredrikson et al. [12, 13] repurpose a model to work *backwards*, inferring unknown attributes of individuals given known attributes and a target prediction. Individuals need not be present in the training data, and results are aggregate statistics rather than information about specific training points.

Differential Privacy. In terms of defenses, McMahan et al. [21] study how to train LSTM models with DP guarantees at a user-level. They investigate utility and privacy trade-offs of the trained models depending on a range of parameters (e.g., clipping bound and batch size). Carlini et al. [6] show that DP protects against leakage of canaries in character-level models, while Song and Shmatikov [27] show that an audit as described above fails when training language models with user-level DP using the techniques of [21]. Pan-privacy [10], on the other hand, studies the problem of maintaining differential privacy when an attacker observes snapshots of the internal state of a DP algorithm between updates.

Deletion of Data. Techniques to update models to delete training data points can be broadly classified into *exact* and *approximate* deletion. Ginart et al. [14] define *exact deletion* of a training point from a model as a stochastic operation returning the same distribution as re-training from scratch without that point, and develop deletion algorithms for k-means clustering with low amortized cost. Bourtoule et al. [5] propose an exact deletion methodology that aggregates models trained on disjoint data shards, trading storage for computation such that only shards that contain deleted points need to be retrained. Exact deletion is equivalent to retraining from scratch, hence, publishing model snapshots before and after deletion matches our adversarial model and our results apply.

Contemporary approximate deletion methods [15, 16] yield models that are only statistically indistinguishable from a model retrained from scratch. These methods stochastically update model parameters based on estimates of the influence of the data to be deleted and achieve relaxations of differential privacy. It would be interesting to study how susceptible to snapshot attacks are models obtained by approximate deletion.

8 CONCLUSION

We presented a first systematic study of the privacy implications of releasing snapshots of a language model trained on overlapping data. Our results show that updates pose a threat which needs to be considered in the lifecycle of machine learning applications. We encourage the research community to work towards quantifying and reducing unintended information leakage caused by model updates, and hope to make practitioners aware of the privacy implications of deploying and updating high-capacity language models.

ACKNOWLEDGMENTS

We thank Doug Orr and Nicolas Papernot for helpful discussions and the anonymous reviewers for their valuable comments.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *23rd ACM SIGSAC Conference on Computer and Communications Security, CCS 2016*. ACM, 308–318.

[2] Galen Andrew, Steve Chien, and Nicolas Papernot. 2020. TensorFlow Privacy. https://github.com/tensorflow/privacy.

[3] Arm. 2020. TrustZone Technology. https://developer.arm.com/ip-products/security-ip/trustzone

[4] Raef Bassily, Adam Smith, and Abhradeep Thakurta. 2014. Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014*. IEEE Computer Society, 464–473.

[5] Lucas Bourtoule, Varun Chandrasekaran, Christopher Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine Unlearning. In *42nd IEEE Symposium on Security and Privacy, S&P 2021*. IEEE Computer Society. To appear.

[6] 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 Association, 267–284.

[7] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. 2015. Leakage-Abuse Attacks Against Searchable Encryption. In *22nd ACM SIGSAC Conference on Computer and Communications Security, CCS 2015*. ACM, 668–679.

[8] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. 2020. When Machine Unlearning Jeopardizes Privacy. arXiv:2005.02205 [cs.CR]

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*, Vol. 1. Association for Computational Linguistics, 380–385.

[10] Cynthia Dwork, Moni Naor, Toniann Pitassi, Guy N. Rothblum, and Sergey Yekhanin. 2010. Pan-Private Streaming Algorithms. In *Innovations in Computer Science, ICS 2010*. Tsinghua University Press, 66–80.

[11] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014), 211–407.

[12] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In *22nd ACM SIGSAC Conference on Computer and Communications Security, CCS 2015*. ACM, 1322–1333.

[13] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon M. Lin, David Page, and Thomas Ristenpart. 2014. Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing. In *23rd USENIX Security Symposium*. USENIX Association, 17–32.

[14] Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. 2019. Making AI Forget You: Data Deletion in Machine Learning. In *Advances in Neural Information Processing Systems 32, NeurIPS 2019*. Curran Associates, Inc., 3518–3531.

[15] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. 2020. Eternal Sunshine of the Spotless Net: Selective Forgetting in Deep Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*. IEEE, 9301–9309.

[16] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten. 2020. Certified Data Removal from Machine Learning Models. In *37th International Conference on Machine Learning, ICML 2020*. PMLR. To appear.

[17] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[18] Intel. 2020. Software Guard Extensions (SGX). https://software.intel.com/en-us/sgx

[19] Ken Lang. 1995. NewsWeeder: Learning to Filter Netnews. In *12th International Machine Learning Conference on Machine Learning, ICML 1995*. Morgan Kaufmann, 331–339.

[20] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics* 19, 2 (1993), 313–330.

[21] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *6th International Conference on Learning Representations, ICLR 2018*. OpenReview.net.

[22] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer Sentinel Mixture Models. In *5th International Conference on Learning Representations, ICLR 2017*. OpenReview.net.

[23] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. *Language Models are Unsupervised Multitask Learners*. Technical Report. OpenAI.

[24] Ahmed Salem, Apratim Bhattacharyya, Michael Backes, Mario Fritz, and Yang Zhang. 2019. Updates-Leak: Data Set Inference and Reconstruction Attacks in Online Learning. arXiv:1904.01067 [cs.CR]

[25] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2019. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019*. The Internet Society.

[26] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership Inference Attacks Against Machine Learning Models. In *38th IEEE Symposium on Security and Privacy, S&P 2017*. IEEE Computer Society, 3–18.

[27] Congzheng Song and Vitaly Shmatikov. 2019. Auditing Data Provenance in Text-Generation Models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019*. ACM, 196–206.

[28] S. Song, K. Chaudhuri, and A. D. Sarwate. 2013. Stochastic Gradient Descent with Differentially Private Updates. In *1st IEEE Global Conference on Signal and Information Processing, GlobalSIP 2013*. IEEE Computer Society, 245–248.

[29] European Union. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation).

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Advances in Neural Information Processing Systems 30, NIPS 2017*. Curran Associates, Inc., 5998–6008.

[31] Ashwin K. Vijayakumar, Michael Cogswell, Ramprasaath R. Selvaraju, Qing Sun, Stefan Lee, David J. Crandall, and Dhruv Batra. 2018. Diverse Beam Search for Improved Description of Complex Scenes. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*. AAAI Press, 7371–7379.

[32] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. In *31st IEEE Computer Security Foundations Symposium, CSF 2018*. IEEE Computer Society, 268–282.

[33] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent Neural Network Regularization. arXiv:1409.2329 [cs.NE]

A RESULTS FOR `talk.politics.mideast`

Table 7 (deferred from Section 4.5) shows the highest-scoring sequences of length 4 in a group beam search with 5 groups for the `talk.politics.mideast` dataset for RNN and Transformer architectures.

Table 7: Top ranked phrases in a group beam search for a model updated with `talk.politics.mideast`. Center for Policy Research is a prolific newsgroup poster; many of the posts around the time the 20 Newsgroups dataset [19] was collected discuss tensions between Turkey and Armenia.

RNN		Transformer	
Phrase	$\widetilde{DS}$	Phrase	$\widetilde{DS}$
`Turkey searched first aid`	31.32	`Center for Policy Research`	200.27
`Doll flies lay scattered`	22.79	`Escaped of course ...`	95.18
`Arab governments invaded Turkey`	20.20	`Holocaust %UNK% museum museum`	88.20
`Lawsuit offers crime rates`	18.35	`Troops surrounded village after`	79.35
`Sanity boosters health care`	11.17	`Turkey searched neither Arab`	37.69