PRIVACY ANALYSIS IN LANGUAGE MODELS VIA TRAINING DATA LEAKAGE REPORT

Huseyin A. Inan*
Microsoft Research
huseyin.inan@microsoft.com

Osman Ramadan*
Microsoft Corporation
osman.ramadan@microsoft.com

Lukas Wutschitz
Microsoft Corporation
lukas.wutschitz@microsoft.com

Daniel Jones
Microsoft Corporation
t-dajon@microsoft.com

Victor Rühle
Microsoft Corporation
virueh@microsoft.com

James Withers
Microsoft Corporation
jawithe@microsoft.com

Robert Sim
Microsoft Research
rsim@microsoft.com

ABSTRACT

Recent advances in neural network based language models lead to successful deployments of such models, improving user experience in various applications. It has been demonstrated that strong performance of language models may come along with the ability to memorize rare training samples, which poses serious privacy threats in case the model training is conducted on confidential user content. This necessitates privacy monitoring techniques to minimize the chance of possible privacy breaches for the models deployed in practice. In this work, we introduce a methodology that investigates identifying the user content in the training data that could be leaked under a strong and realistic threat model. We propose two metrics to quantify user-level data leakage by measuring a model's ability to produce unique sentence fragments within training data. Our metrics further enable comparing different models trained on the same data in terms of privacy. We demonstrate our approach through extensive numerical studies on real-world datasets such as email and forum conversations. We further illustrate how the proposed metrics can be utilized to investigate the efficacy of mitigations like differentially private training or API hardening.

1 Introduction

Advances in language modeling have produced high-capacity models which perform very well on many language tasks. Language models are of particular interest as they are capable of generating free-form text, given a context, or even unprompted. There is a plethora of applications where language models have the opportunity to improve user experience, and many of them have recently been deployed in practice to do so, such as text auto-completion in emails and predictive keyboards (illustrated in Figure 1). Interestingly, language models with massive capacities have been shown to achieve strong performance in other tasks as well, such as translation, question-answering etc. even in a zero shot setting without fine-tuning in some cases [Brown et al., 2020a].

On the other hand, recent studies have demonstrated that these models can memorize training samples, which can be subsequently reconstructed using probing attacks, or even during free-form generation [Carlini et al., 2019, 2020]. While domain adaptation of general phrases is intended, the model should not leak or memorize rare sequences which could lead to a privacy breach according to GDPR, such as singling out of a user [GDPR Article 29 Working Party, 2014].

Efforts to mitigate the risk that a model may yield rare samples which violate privacy include applying differential privacy (DP) during training [Dwork, 2011, Song et al., 2013, Abadi et al., 2016], as well as API hardening to ensure that attackers have little or no access to the model's underlying probability distributions. While these approaches can be successful, it is challenging to quantify the residual privacy risks in language models, whether or not mitigations have been applied. In this work we propose a methodology for privacy investigations of a language model trained on

*Equal Contribution

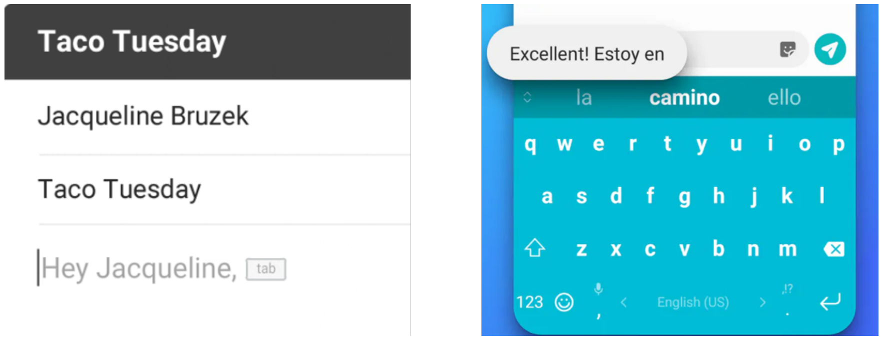


Figure 1: Two examples of language model deployments in practice. The figure on the left (image credit: [Lambert, 2018]) is the Smart Compose feature for Gmail [Chen et al., 2019] and the figure on the right (image credit: [Microsoft SwiftKey]) is the Microsoft SwiftKey Keyboard.

confidential user content. Furthermore, we aim to produce metrics quantifying the likelihood a model might leak rare training data, under the strictest black-box assumptions about access to the model, i.e. that attackers can access only the model's top-k prediction at each token position, given an input prefix. This choice of threat model enables us to assess a model's risk for realistic deployment scenarios, assuming best practices in API hardening are employed.

1.1 Contributions

This paper makes the following contributions:

1. We propose a methodology called training data leakage report that investigates the user content in the training data that could be leaked by the model when prompted with the associated context. We introduce a set of features to help assess the leakable content in terms of user-level privacy.
2. From our leakage report we introduce metrics that allow comparing models of various kinds (e.g. a DP model vs. a non-DP model) that are trained on the same training data in terms of privacy. Our metrics are straightforward to interpret compared to other privacy quantifiers such as differential privacy's ϵ, which may be difficult to relate to real-world risks.
3. We demonstrate experimental results on real-world datasets illustrating the generation of the leakage report. We show how the proposed privacy investigation can provide valuable information towards protecting user-level privacy. We further study the effects of mitigation techniques such as differential privacy and API hardening through the metrics introduced in this work.

The outline of the paper is as follows. In Section 2, we provide background information for the language models focused in this work. Section 3 defines the threat model and discusses the ability of an adversary towards attacking a language model deployed in practice. In Section 4, we introduce our methodology of investigating a model trained on user content for the purpose of user-level privacy protection. In Section 5, we propose metrics to quantify user-level privacy leakage. We demonstrate our framework through numerical studies on real-world datasets in Section 6. Section 7 discusses the related work and future directions and concludes the paper.

2 Background: Language Models

Language modeling[2] is the task of learning the underlying probability distribution over sequence of words in a natural language. A statistical model for a sequence of tokens $w_1, \ldots, w_n$ is represented by the joint probability $\Pr(w_1, \ldots, w_n)$, which can be further decomposed as the product of conditional probabilities:

$$\Pr(w_1, \ldots, w_n) = \prod_{i=1}^{n} \Pr(w_i | w_1, \ldots, w_{i-1}). \tag{1}$$

Here $\Pr(w_i | w_1, \ldots, w_{i-1})$ represents the probability of the occurrence of token w_i given the previous token sequence $w_1, \ldots, w_{i-1}$.

[2]We refer to statistical language modeling throughout the paper.

It has been shown that neural networks can be utilized to estimate these conditional distributions effectively and be employed as language models [Bengio et al., 2003]. Given an unsupervised corpus of tokens $\mathcal{W} = \{w_1, \ldots, w_n\}$, a standard language modeling objective is to maximize the following likelihood function:[3]

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \log \Pr(w_i | w_1, \ldots, w_{i-1}; \theta),$$

where the conditional probability on w_i is calculated by evaluating the neural network with parameters θ on the sequence $w_1, \ldots, w_{i-1}$.

The quality of a language model is commonly measured by two metrics, namely perplexity and top-k accuracy. Perplexity measures the likelihood of text sequences and is defined as $\mathrm{PP}(w_1, \ldots, w_n) = 2^{-l}$ where

$$l = \frac{1}{n} \sum_{i=1}^{n} \log_2 \Pr(w_i | w_1, \ldots, w_{i-1}).$$

The evaluation of the perplexity on unseen data indicates how well the model fits the underlying distribution of the language. The smaller the value of perplexity, the better the language model is at modeling the data. Top-k accuracy metric is defined as the ratio of the number of correct predictions to the total number of tokens[4]. The relevance of the parameter k depends on the application. For instance, the accuracy for the highest-likelihood candidate (top-1 accuracy) is important for text auto-completion feature in emails [Chen et al., 2019] whereas top-3 accuracy is also of interest for predictive keyboards (Microsoft SwiftKey, Gboard). See Figure 1 for an illustrating example.

There are a vast number of architectures employed for language models. At a high level, these architectures are either derived from variants of recurrent neural networks (RNNs) [Mikolov et al., 2010, Sundermeyer et al., 2012, Peters et al., 2018] or based on self-attention mechanisms of the transformer [Vaswani et al., 2017, Radford et al., 2018, Howard and Ruder, 2018, Devlin et al., 2019, Yang et al., 2019, Radford et al., 2019, Sun et al., 2019, Brown et al., 2020a, Turing-NLG, 2020]. Recently, large transformer based models have been achieving impressive state-of-the-art results in a variety of tasks [Brown et al., 2020a]. On the other hand, RNN based architectures might be favored in practice as well, e.g. when there are strict latency or memory requirements [Chen et al., 2019].

3 Threat Model

Our threat model is tailored for privacy considerations when a language model is trained on confidential user content, which is highly likely to contain sensitive information that would lead to privacy violations in case they are leaked by the model [GDPR Article 29 Working Party, 2014, White House Office of Science and Technology Policy (OSTP), 2019]. Such privacy considerations are in fact legitimate as language models perform next token prediction so they could be used in a generative fashion by entering a particular text prefix and asking the model to auto-complete indefinitely. Here, the danger is imminent as it is not *a priori* clear what will be leaked from the user content in the training data. We note that any language model with non-zero utility will necessarily have the top-1 accuracy in the training data bounded away from zero. Therefore, "something" will be leaked from the user content in the training data[5]. Since the main objective of training language models is modeling the underlying distribution of a language, the expectation is that well-generalized models do not memorize the rare sensitive information in the training data, as they are out-of-distribution and irrelevant to the learning task, hence unnecessary to improve the model performance. Unfortunately, recent results show that this is not the case [Carlini et al., 2019, 2020, Feldman, 2020, Brown et al., 2020b, Petroni et al., 2019]. In fact, very recently it has been shown that when the data distribution is long-tailed (as is the natural language [Newman, 2005]) label memorization is necessary for achieving near-optimal accuracy on test data [Feldman, 2020, Brown et al., 2020b]. Therefore, it is imperative to build privacy monitoring techniques to minimize the chances of an "accidental" data leakage that would lead to privacy violations.

Based on the discussion above, we consider a practical threat model that is relevant to the language models deployed in practice. We assume a black-box access, where a curious or malevolent user can query a pre-trained and deployed language model on any sequence of tokens $w_1, \ldots, w_i$ and receive the top-k predictions returned by the model for the next token w_{i+1}. See Figure 2 for an illustrating example. We place no assumption on the parameter k and let it be completely determined by the particular application for which the query is made (see Figure 1). We note that the

[3]The decomposition in (1) is called forward autoregressive factorization. Although not all the works referred in this section use this factorization to train language models, we are solely interested in how they operate when they are deployed in practice. Therefore, we do not delve into the details of exact training procedure for each architecture.

[4]When $k > 1$, correct prediction refers to the label being in the list of k predictions returned by the model.

[5]We will use the terms "leak" and "correct prediction on the training data" for a model interchangeably in what follows.

threat model does not assume the availability of confidence scores or probabilities for the predictions and it is trivially applicable to the deployed models in practice. In fact, even the availability of the next token prediction(s) may not always be the case if the model does not return any prediction under certain conditions (e.g. when the prediction score is below a pre-fixed triggering threshold [Chen et al., 2019]). However, since there is no guarantee that all sensitive information will be on the safe side of the triggering threshold, we believe it might be better to have protection against the worst case where the model prediction is available for the next token w_{i+1} when any sequence of tokens $w_1, \ldots, w_i$ is queried.

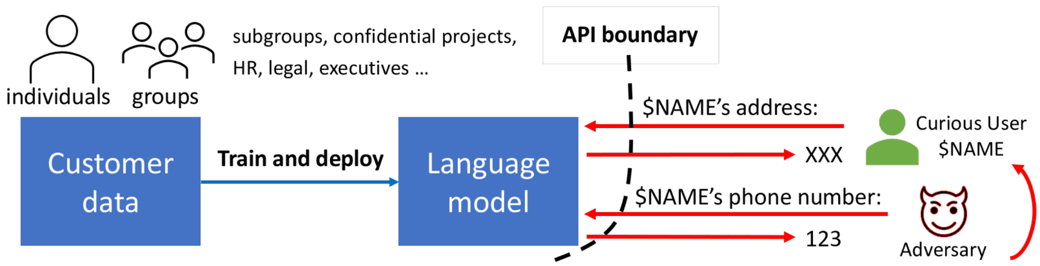


Figure 2: Illustration of our threat model. A language model is deployed after being trained on user content. One can query the model with a sequence of tokens and receive the top-k predictions for the next token (top-1 in this example). A curious user where $NAME represents the name of the user queries the model to see if their address is leaked by the model. More dangerously, an adversary inputs a directed query to learn the phone number of the targeted user.

The threat model allows a curious user to know whether any sensitive information in their data is leaked by the model. Therefore, the data owner can use any prefix in their data to query the model. On the other hand, the threat model also includes the case of a malevolent user, who could input directed queries in order to extract sensitive information about a user specifically targeted. Needless to say, successful extraction of any sensitive information could have catastrophic effects for the corresponding user.

4 Training Data Leakage Report

In this section, we introduce our framework to investigate a model trained on user content for the purpose of user-level privacy protection. We fix the notation first.

Notation For a language model trained on user content, let $\mathcal{U} = \{U_1, \ldots, U_n\}$ specify the set of users. For each user $U_i \in \mathcal{U}$ for $i \in \{1, 2, \ldots, n\}$, we define the set $\mathcal{D}_i = \{D_i^1, D_i^2, \ldots, D_i^{|\mathcal{D}_i|}\}$ ($|\mathcal{D}_i|$ refers to the size of the set $\mathcal{D}_i$) as the corresponding content on which the language model is trained. Each content D_i^j is basically a sequence of tokens $w_1, w_2, \ldots, w_{|D_i^j|}$[6]. See Figure 3 for an illustration of how a language model is typically trained on a sequence of tokens. The training data $\mathcal{D}$ is the combination of all user content, i.e., $\mathcal{D} = \cup_{i \in \{1,2,\ldots,n\}} \mathcal{D}_i = \cup_{i \in \{1,2,\ldots,n\},\, j \in \{1,2,\ldots,|\mathcal{D}_i|\}} D_i^j$.

We introduce our training data leakage report on a language model trained on the training data $\mathcal{D}$. After the training procedure is completed, the first step of our framework is to run the model through the training data and collect its correct predictions in the training data. We illustrate this step with an example in Figure 4. This collection consists of sequences of tokens where the correct prediction (depicted in green) is observed in top-k predictions of the model consecutively. We emphasize that consecutive correct predictions is an important phenomenon because the longer the model leaks a training sequence $w_{i+1}, w_{i+2}, \ldots$ having seen the context $w_1, \ldots, w_i$, the more it discloses user content, causing privacy concerns. Therefore, we do not break sequences where the model provides correct predictions consecutively and collect all such sequences in the training data. In Algorithm 1 we provide the pseudo-code to collect the correct predictions of the model as described above. Let us denote this collection as $\mathcal{S}$. We note that $\mathcal{S}$ is a multiset because it can contain multiple instances of a correctly predicted sequence in the training data[7].

[6] Based on the preprocessing of the training data, this could be a sentence, a paragraph (e.g. a Reddit post), or an email etc. We emphasize that $\mathcal{D}_i$ may even be considered as one large text sequence since it belongs to a single user and that is all it matters for our purpose.

[7] Henceforth we will use the terms set and multiset interchangeably for $\mathcal{S}$.

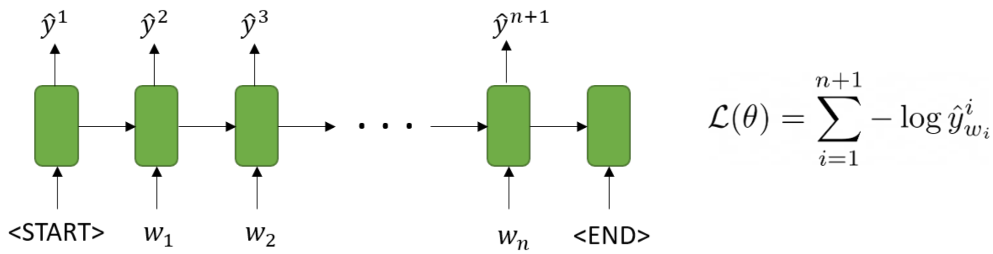


Figure 3: A typical way of training a language model on a sequence of tokens $w_1, w_2, \ldots, w_n$ (RNN type architecture is depicted for the sake of illustration). The model is yielding a probability distribution $\hat{y}^{i+1}$ having seen the context $w_1, \ldots, w_i$ to predict the next token w_{i+1} for $i \in \{1, 2, \ldots, n\}$. We note that the loss function is composed of loss at each time step, therefore, the model learns to predict w_{i+1} having seen the tokens $w_1, \ldots, w_i$ for all $i \in \{1, 2, \ldots, n\}$.

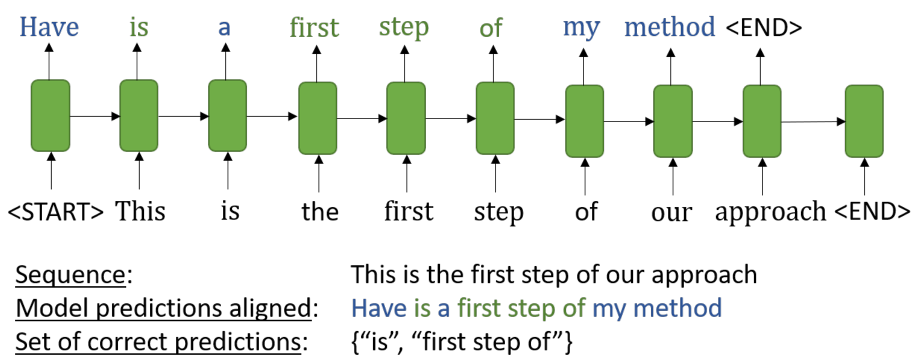


Figure 4: An illustration of the collection of correct model predictions. We run the model through each sequence in the training data and obtain the top-k (top-1 in this example) prediction(s). We then collect the sequence of tokens where the model consecutively provides the correct prediction.

The main part of our training data leakage report is to provide key features of each sequence in the set $\mathcal{S}$, which contain important information for privacy investigations. We call these features as *total count in* $\mathcal{S}$, *user count in* $\mathcal{S}$, *total count in* $\mathcal{D}$, *user count in* $\mathcal{D}$, *context(s)*, and *perplexity(ies)*. We next describe each feature in detail and provide the accompanying Table 1 as an illustrative example.

total count in $\mathcal{S}$: This is simply the number of occurrences of a sequence in the multiset $\mathcal{S}$. This feature shows in total how many times the model leaks a sequence given the correct context. A large number may seem to indicate that the sequence is common and unproblematic in terms of privacy, however, that may not be the case if it is present in only a single user's data, which is captured in the next feature.

user count in $\mathcal{S}$: For any sequence in $\mathcal{S}$ where the total count is larger than one, here we count the number of distinct users for which *the correct prediction of this sequence is made*. This is important because a sensitive information of a single user may appear multiple times in their data (e.g. address being emailed a number of times), which can be memorized by the model. Along with the previous feature we can investigate such cases effectively with this feature.

total count in $\mathcal{D}$: In the previous two features, we count the occurrences in the set $\mathcal{S}$. In the next two features, we count the occurrences in the training data $\mathcal{D}$. This may be helpful because for a sequence that is predicted correctly only for a single user, i.e. user count in $\mathcal{S}$ is one, this may not immediately imply that a sensitive information is leaked by the model. It may be the case that the sequence appears many times among various users' data, which could provide "plausible deniability" in the sense that many other users contributed the model to learn this sequence. This can be calculated by simple string matching and formally expressed as $\sum_{i=1}^{n} \sum_{j=1}^{|\mathcal{D}_i|} (\text{count of } w \text{ in } D_i^j)$ for a sequence $w \in \mathcal{S}$.

Table 1: An artificial example to describe the features of our training data leakage report. In this example, there is a sequence "very much" appearing two times in the multiset $\mathcal{S}$, meaning that it is predicted correctly two times by the model. These correct predictions appear in a single user's data (either in a single D_i^j or two different $D_i^{j_1}$ and $D_i^{j_2}$ for some user U_i). The corresponding contexts on which the model produces this correct sequence are "Thank you" and "I like cats" and the corresponding perplexities are 1.3 and 3.6. On the other hand, the sequence "very much" itself appears ten times in the training data $\mathcal{D}$ (only in two of which the model predicts the sequence correctly), among five user's data.

$\mathcal{S}$	TOTAL # IN $\mathcal{S}$	USER # IN $\mathcal{S}$	TOTAL # IN $\mathcal{D}$	USER # IN $\mathcal{D}$	CONTEXT(S)	PERP.
"VERY MUCH"	2	1	10	5	["THANK YOU", "I LIKE CATS"]	[1.3, 3.6]

user count in $\mathcal{D}$: Connected to the previous feature, here we count the number of distinct users for which a sequence in $\mathcal{S}$ is found in their data. We note that here we do not consider whether the model correctly predicts the sequence or not given the right context, that was done in user count in $\mathcal{S}$. Instead, we calculate this by simply checking if the sequence can be found in a user's data via string match. This can be formally expressed as $\sum_{i=1}^{n} 1(\exists j \in \{1, 2, \ldots, |\mathcal{D}_i|\} \text{ s.t. } w \subseteq D_i^j)$ where $1(\cdot)$ is an indicator function for a sequence $w \in \mathcal{S}$. Sequences for which the user count in $\mathcal{D}$ is large are unlikely to be sensitive for a single user. However, there might still be concerning cases, e.g., the model predicts the sequence correctly only for a single user (i.e. user count in $\mathcal{S}$ is one) although there is plausible deniability as discussed previously.

context(s) For any sequence in $\mathcal{S}$, this feature provides the corresponding context(s) with which when prompted the model it produces the sequence correctly. This feature is useful to check for instance if long sequences can be found with short contexts, which would indicate that the model completes a long user content when prompted with a short initial context.

perplexity(ies) Connected to the feature above, this feature provides the corresponding perplexity(ies) for any sequence in $\mathcal{S}$. This is also an important feature as it shows how certain the model is when predicting the sequence given the right context. Furthermore, it also allows comparing the correct predictions of the model with a public model[8] that is not trained on $\mathcal{D}$. Considering a sequence in $\mathcal{S}$ where the user count in $\mathcal{D}$ is one, a small perplexity on the language model along with a large perplexity on public model might indicate that a sensitive information is leaked about the corresponding user since the sequence is "surprising" to the public model by the large perplexity.

[8] Public model refers to a language model trained on a public dataset.

Algorithm 1 The collection of correct model predictions.

Input: A language model $LM(\cdot)$ and the corresponding training data $\mathcal{D}$
Output: The (multi)set $\mathcal{S}$ of correct predictions
Initialize $\mathcal{S} = []$
for $i = 1$ **to** n **do**
  for $j = 1$ **to** $|\mathcal{D}_i|$ **do**
    Initialize $W =$ ""
    Let $D_i^j = [w_1, \ldots, w_{|D_i^j|}]$
    for $l = 1$ **to** $|D_i^j|$ **do**
      Obtain top-k predictions $preds = LM(D_i^j[: l])$
      if $w_l \in preds$ **then**
        Append w_l to W
      else if $W \neq$ "" **then**
        Append W to $\mathcal{S}$ and initialize $W =$ ""
      end if
    end for
  end for
end for

Before concluding this section, we discuss a number of points regarding our training data leakage report.

- We note that in our framework the sequences in the set $\mathcal{S}$ are created when the model is prompted with the "right" context, i.e. the context on which the model is trained for next token prediction task. However, a sensitive information might even be leaked under a context different from any one that is in the training data. Such a case might be missed if the same sensitive information is predicted wrong under the context in the training data, therefore not being included in the set $\mathcal{S}$. Looking at the loss function in Figure 3, intuitively one can expect that the model would more likely predict the sequence under the context it has seen during training than any other context, however, there is also no guarantee that this will always be the case. Therefore, it might be worthwhile to extend the set of contexts beyond the training data and then create the set $\mathcal{S}$. This is an interesting future direction that could strengthen the privacy investigation of a language model.
- Such a detailed investigation of sequences in our training data leakage report may not be possible if the model training is done with no access to look at the training data [Chen et al., 2019]. In that case, we can simply replace each sequence and its corresponding context(s) with their length of tokens and still obtain valuable information. For example, we can investigate the length of the sequences for which the user count in $\mathcal{S}$ is small to see if long completions are possible or check the length of the contexts to see if the leakage is possible with short contexts. The perplexities could also be very helpful in this case because we can measure how surprising each sequence is to a public model that is only trained on public dataset(s). If the perplexities are similar, then this indicates that the prediction is as "familiar" to the public model, therefore, unlikely to be a privacy concern for a user.

5 Metrics to Quantify Privacy Leakage

Our training data leakage report may include a massive number of sequences when the training data $\mathcal{D}$ is large, making privacy investigations infeasible. However, simple filtering procedures can be applied to reduce the number of sequences effectively. For instance, keeping the ones for which the user count in $\mathcal{S}$ is less than p for some threshold p of interest is a reasonable operation. The sequences filtered here would the ones where there are at least p users having this sequence in their data and the model predicts the sequence correctly for each one of these users. It becomes less and less likely to be a privacy concern as p increases. From the perspective of model predictions, this is reminiscent of k-anonymity [Sweeney, 2002] as a famous data anonymization technique[9].

Another important case is regarding the sequences in $\mathcal{S}$ for which the user count in $\mathcal{D}$ is one (automatically, user count in $\mathcal{S}$ is also one), i.e., there is only one user having this sequence in their data and the model leaks the sequence when prompted with the corresponding context[10]. This is inarguably the case with the most potential to result in privacy violations.

Building on this, we propose two metrics to quantify user-level privacy leakage, which are straightforward to interpret, and compare different models trained on the same training data in terms of privacy:

1. The first metric is the number of sequences in the set $\mathcal{S}$ that are unique for a user, i.e. the sequences for which the user count in $\mathcal{D}$ is one.
2. The second metric is a curated version of the first metric. We still consider the sequences in the set $\mathcal{S}$ that are unique for a user but we remove the ones for which the ratio of the perplexity with respect to a public model and our language model is below some threshold t, i.e. $\text{PP}_{\text{public}}(w)/\text{PP}_{\text{lm}}(w) < t$. This basically filters out the unique sequences that have similar perplexities with respect to a public model as there is plausible deniability of similar leakage possibility, given a public model. We further define the worst-case *leakage epsilon*

$$\epsilon_l \triangleq \max_{w \in \mathcal{S}} \frac{\text{PP}_{\text{public}}(w)}{\text{PP}_{\text{lm}}(w)}, \tag{2}$$

measuring the perplexity ratio with respect to a public model maximized over the unique sequences in the set $\mathcal{S}$ to capture the worst-case scenario.

We next discuss the pros and cons of our proposed metrics. Our first metric is very simple and easy to use. However, we observe in our experiments that even a public model that is not trained on a private data can predict unique sequences in

[9] Since the contexts will likely be different, k-anonymity is still substantially more powerful for anonymization.

[10] or contexts. Note that the total count in $\mathcal{D}$ and total count in $\mathcal{S}$ can still be arbitrary since the sequence can appear multiple times in the user's data (e.g. appearing 10 times while in 5 of them the model predicts the sequence correctly.). Nevertheless, since it belongs to only one user, the sequence should be protected equally.

the private data. Such unique sequences would likely not constitute a privacy violation since the public model has not seen any private data in its training. Another example of no privacy violation is when a sequence is unique to a user but it happens to be copied from the prefix. Models with attention mechanism [Bahdanau et al., 2016] have a strong ability to copy tokens from previous context to predict the next token. A model leaking a unique sequence by applying the copying mechanism may not be considered to violate privacy. To handle such cases, we can use the number of unique sequences obtained from a public model as a benchmark to compare models trained on private data. However, the main disadvantage of this metric is that for a model trained on private data, the metric does not consider the sensitivity of the unique sequences leaked by this model. Therefore, one would not know the status of the unique sequences beyond the ones that can be predicted by a public model.

Our second metric touches on this point by eliminating the sequences that do not look surprising to a public model. However, the main disadvantage of this metric is the hardship of the choice of the threshold t. It is challenging to agree on a value that is assuring privacy protection. For this reason, we introduce the term worst-case leakage epsilon, denoted by ϵ_l, measuring the worst case perplexity ratio with respect to a public model over the unique sequences in the set $\mathcal{S}$. This is motivated by the definition of differential privacy, which bounds the worst-case effect of a single substitution in the data. A smaller ϵ_l for a language model translates into a better privacy protection as the unique sequences leaked by the model will have relatively similar perplexities with respect to a public model, providing plausible deniability for each one of them.

The final point related to both metrics is the choice of the public model. Assuming that the private data has no connection to public datasets, ideally one would take the strongest model that is trained on various public datasets to provide a good benchmark. However, the distribution of any public dataset may differ substantially to that of a private one, leading to a pessimistic ϵ_l. In that case, one can also consider removing the users for which the language model leaks a unique sequence and all their data from the training data and train another model on the remaining users. This latter model may be employed as a "public model" in Equation (2) to calculate ϵ_l since it has not seen any data of a user in the set $\mathcal{S}$ during the training.

6 Case Study: Tab Attack

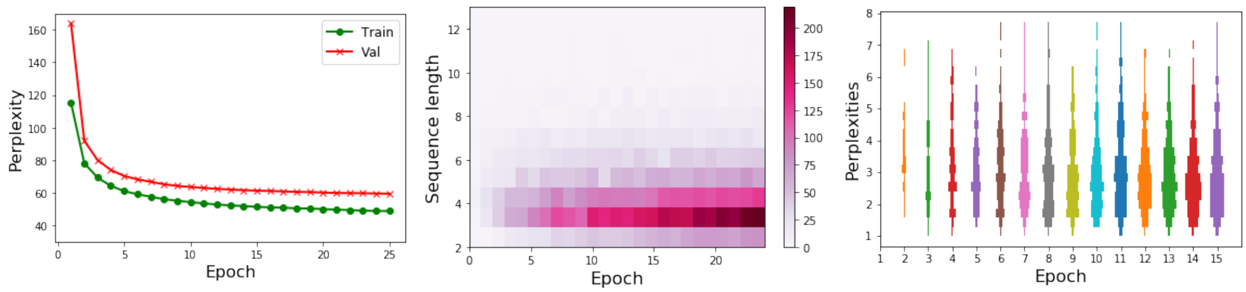
In this section, we provide a case study of our training data leakage report through numerous experimental settings. We will consider an attack setting that has access to top-1 predictions of a language model. Having in mind the text auto-completion feature in emails example where the predictions are applied by pressing the TAB key on the keyboard (see Figure 1), we dub this attack as the *tab attack*. We will investigate the unique sequences (i.e. the ones with the user count in $\mathcal{D}$ is one) that could be leaked through the tab attack by providing the corresponding context. These can simply be obtained by filtering the sequences in the set $\mathcal{S}$ and keeping the ones with the user count in $\mathcal{D}$ is one. We note that such sequences are most likely to cause privacy concerns as they are unique content for a user in the training data. We further note that although the attacker ability is limited to top-1 predictions, the model builder can utilize all information to investigate the unique sequences that could be leaked by the tab attack prior to the model deployment. We will apply our leakage report to the unique sequences in the set $\mathcal{S}$ to assess the attack surface under the tab attack threat model.

6.1 Avocado dataset

Dataset We use the Avocado dataset [Oard et al., 2015] that contains 322k email correspondences of 413 users. We use the $80\%/20\%$ training/val set split. We emphasize that this is a dataset where the users are not independent (i.e. sensitive information related to both the company and the users such as names, positions, partner company names etc. are shared across communications). This is an interesting setting as it is related to businesses providing language models to enterprise customers using their data and it also provides important insights in terms of user-level privacy protection.

Model We use a two-layer LSTM model as the language model for the next-word prediction task. We set both the embedding dimension and LSTM hidden-representation size to 128. We use the Adam optimizer with the learning rate set to 1e-3 and batch size to 32. We fix the vocabulary to the most frequent 10k tokens in the training corpus (out of 95k tokens).

In this experiment, we train the language model for 25 epochs and obtain one snapshot of the model at each epoch (around the first quarter of an epoch). We run the tab attack and investigate the results (i.e. our training data leakage report with sequences where user count in $\mathcal{D}$ is one) for each snapshot of the model. Figure 5 shows the progress of the model performance and the tab attack statistics as the model is being trained for 25 epochs. We first note from Figure 5a that the train and validation perplexities are close for the language model, indicating that the model is not overfitted. This is due to choosing a rather small capacity model (less than 3M parameters). Somewhat expectedly, we observe



(a) (b) (c)

Figure 5: Results of the experiment on the language model trained on Avocado dataset [Oard et al., 2015]. Figure (a) presents the performance of the model with the perplexity on the train and validation set. Tab attack is performed at each epoch on a snapshot of the model. Figure (b) shows the histogram of the length of the unique sequences leaked by the model over 25 epochs. Figure (c) presents the perplexities of these unique sequences. The width of the bins are proportional to the number of sequences that appear in them at the corresponding perplexity. We excluded last 10 epochs in Figure (c) for the sake of visualization.

in Figure 5b that the number of unique sequences leaked by the model increases consistently as the model training continues (colors getting darker as we go right for each sequence length). Another interesting observation is that the majority of the unique sequences are having perplexities below 3 as the training continues. This unfortunately indicates that not returning any prediction when confidence is below a certain threshold may not be sufficient to provide privacy in some cases.

Looking at the individual unique sequences, we already see sensitive information such as names, names with positions, product specific information and partner company names being leaked by the model starting at epoch 2[11]. We point out that the reason why unique sequences of such sensitive information can be leaked by the model even in the beginning of the training is that the user content is highly correlated. This means that the snippets of these unique sequences appear many times in the training data, although the particular order appears only once (e.g. the sequence w_1, w_2, w_3, w_4 being unique while 2-grams w_1, w_2 and w_2, w_3 and w_3, w_4 appearing many times). This causes the model to produce a long chain of correct predictions by learning parts of the connections from multiple users and the remaining few ones from a single user. Further observations of this experiment is presented below:

- We believe that personally identifiable information (PII) scrubbing of the training data should be a crucial first step, especially if the user content is correlated, since choosing a small subset of the most frequent tokens does not disable the sensitive tokens to find their way into the vocabulary when they are repeated many times across users. Even when we fix the vocabulary with words appearing in at least 25 users' data (which hurts the utility substantially), we still observe sensitive tokens present in the vocabulary. Therefore, in general the ideal setting would be to have users as independent as possible and fixing the vocabulary after applying PII scrubbing with the tokens that appear among many users' data, instead of the most frequent ones, in case sensitive information might be repeated many times in a single user's data.
- We do not claim that any unique sequence leaked by the model would be a privacy concern. In fact, there are seemingly common sequences such as "been able to do that" with the context "... I have not" that happen to be unique for a user in the training data and the model provides the correct prediction[12]. We note that this prediction may not be as surprising for a public model (i.e. low perplexity), therefore, it is possible to filter these cases further by comparing with a public model in perplexities.
- We try the idea of not returning any prediction when the perplexity is above a certain threshold (i.e. the model does not have high confidence). Choosing the threshold as 8, the accuracy of the snapshot at the last epoch on the validation set drops from 28.29% to 26.19%. This had only a little effect while leaving many of the unique sequences with sensitive information untouched. Taking things to extremes, setting the threshold as 3 using the model snapshot at epoch 5 returns no unique sequences, but with an accuracy of 14.81%. We believe that this technique along with a vocabulary fixed with the suggested principles above could potentially find better privacy-utility tradeoffs.

[11]Due to strict licensing requirements of the Avocado dataset, we are unable to provide specific examples. However, we present detailed results in the next experiment.

[12]"been able to do" appears 13 times and "to do that" appears 36 times in the data. The text has been altered to comply with the Avocado license.

Table 2: Results of the experiment on the language models trained on Reddit dataset [Al-Rfou et al., 2016]. We provide the perplexity and accuracy on the validation set to compare the performances of the models. In the next column, we provide the number of unique sequences in the set $\mathcal{S}$ for each model. We calculate worst-case leakage epsilon ϵ_l for some of the models for comparison in the last column.

MODEL	VAL PERP	VAL ACC (%)	# UNIQUE SEQ. IN $\mathcal{S}$	ϵ_l
PRIVATE LM	69.41	23.7	3757	-
DP-LM RANINI $\epsilon = 3.28$	290.03	14.46	0	-
DP-LM RANINI $\epsilon = 4.68$	130.32	19.60	5	-
DP-LM RANINI $\epsilon = 6.20$	107.77	20.83	11	1.89
DP-LM RANINI $\epsilon = 26.40$	96.54	21.48	30	-
PUBLIC LM	757.48	13.1	159	-
DP-LM PUBINI $\epsilon = 2.98$	183.09	19.71	157	1.34
DP-LM PUBINI $\epsilon = 4.47$	106.70	21.90	203	-
DP-LM PUBINI $\epsilon = 6.68$	92.76	22.20	246	3.78

6.2 Reddit dataset

We next study a large-scale example as a more realistic setup for the deployed language models in practice.

Dataset We use a large dataset of Reddit posts, as decribed by Al-Rfou et al. [2016], that contains 140M sentences from 4.4M users crawled from Reddit posts. It is split into 90% training and 10% validation at random. Reddit is an interesting conversational dataset that has been used in privacy research since each post in the dataset is keyed by a user, so the data can be grouped by users to provide user-level privacy. We provide three sets of language models trained on the private Reddit dataset.

1. A language model trained on the Reddit dataset. This will be referred to as *Private LM* in our results.
2. A language model trained on the Reddit dataset with differential privacy [Abadi et al., 2016]. We take four snapshots of the model during training, corresponding to four differentially-private language models with epsilons 3.28, 4.68, 6.20, and 26.4[13]. The training begins with a random initialization of the weights. These models will be referred to as *DP-LM RanIni* $\epsilon = \cdot$.
3. A language model trained on the Reddit dataset with differential privacy. The difference here is that the model weights are initialized from a public model trained on Google News dataset [Chelba et al., 2013]. It has been shown that with transfer learning, one can obtain strong privacy guarantees with a minor cost in utility [Abadi et al., 2016, Tramèr and Boneh, 2020, Papernot et al., 2020]. We similarly take three snapshots of the model during training, corresponding to three differentially-private language models with epsilons 2.98, 4.47, and 6.68. These models and the public model will be referred to as *DP-LM PubIni* $\epsilon = \cdot$ and *Public LM* respectively.

The model architecture is same for all these models and the details are specified below.

Model We use a one-layer GRU model as the language model for the next-word prediction task. The embedding size is set to 160 and the hidden size to 512, and the vocabulary is fixed to the most frequent 10k words in the training corpus (out of 3.2M words). We use the Adamax optimizer with the learning rate set to 1e-3 and the batch size is set to 3072 in the differentially-private training and to 512 otherwise.

We provide in Table 2 the performances of the models and the result of the tab attack for each of them. We discuss the results of this experiment in what follows.

We observe from Table 2 that the private LM that is trained without differential privacy leaks a huge number of unique sequences (3757) from the training data. There are 759 unique sequences for which the number of tokens is larger than 9. A majority of these examples are coming from highly-repeated sentences (728 of these sequences are repeated somewhere between 50-34372 times) by the bots in the Reddit dataset[14]. This shows the necessity of de-duplication at a granular level (e.g. removal of sentence duplicates) as also observed by Carlini et al. [2019, 2020].

[13]The models satisfy user-level DP and $\delta \lesssim 1/(\#\text{ users})$ same for all models.

[14]An example of a unique sequence memorized by the model is “has been automatically removed because the title does not include one of the required tags .” repeated 5377 times in the bot’s data.

For the DP-LMs that are snapshots of a model trained with random initialization of weights, we observe small number of unique sequences leaked by the models. Interestingly, we get no unique sequence with the first one having $\epsilon = 3.28$, although there is a high cost in terms of utility. We provide the list of unique sequences for the models with $\epsilon = 4.68, 6.20$ and 26.4 in Table 3, 4, and 5 of Appendix A respectively. We observe the efficacy of user-level differentially private language model training by noting that the unique sequences with large repetitions that were memorized by the private model have all disappeared with DP-LMs. Furthermore, there is a substantial decrease in the number of unique sequences, even for the DP-LM with relatively high epsilon value $\epsilon = 26.8$, which does not provide a reasonable theoretical privacy guarantee.

A phenomenon we have observed consistently over all experiments is about the punctuation. The appearances of punctuation in between common n-grams in the training data cause the existence of many unique sequences. Almost all unique sequences for the DP-LMs presented in Appendix A have a punctuation. In our experiments we did not exclude the punctuation from the model predictions and treated them as any other token in the training data.

For the DP-LMs that are snapshots of a model trained by initialization from a public model, we observe relatively larger number of unique sequences leaked by the models. However, we again note that a direct comparison is not fair because the public model itself can predict 159 unique sequences from the private data, without seeing any private data in its training. Since the differentially private training is initialized from the public model in this case, it should be expected to obtain larger number of unique sequences. The worst-case leakage epsilon ϵ_l may provide a better ground for a fair comparison of models trained in different ways (e.g. random initialization vs. transfer learning), however, there is a dependence on the public model of choice.

Finally, we calculate the worst-case leakage epsilon ϵ_l for three DP-LMs that provide interesting conclusions. The public models in the calculation of Equation (2) are as follows. For each model, we take the users who are the owners of the unique sequences leaked by the model and remove all their data from the training data. We subsequently train a new model on the remaining users. We consider the new model as the public model for the users of the unique sequences since it has not seen any data of these users during its training. We note that although DP-LM RanIni $\epsilon = 6.20$ model leaks 11 unique sequences, the worst case leakage epsilon ϵ_l is just 1.89. This indicates that the unique sequences leaked by the model can also be simply learned from other users because they have similar perplexities with respect to the public model. DP-LM PubIni $\epsilon = 2.98$ model has $\epsilon_l = 1.34$, much smaller than the other two models and this may not be surprising since $\epsilon = 2.98$ provides much stronger privacy guarantees compared to $\epsilon = 6.20$ and $\epsilon = 6.68$. Finally, DP-LM PubIni $\epsilon = 6.68$ model has $\epsilon_l = 3.78$, which is significantly larger than the DP-LM RanIni $\epsilon = 6.20$ model. We note that there is no direct relationship between ϵ of DP and our ϵ_l, however, we believe that $\epsilon_l = 3.78$ is somewhat pessimistic for the former model. The reason is related to ϵ_l being dependent on the public model of choice. Ideally, one would choose a powerful public model to get competitive perplexities, however, we trained the same model on the dataset described above. We leave it as a future work to try public models with more capacities to see the effect on the ϵ_l values.

7 Related Work and Conclusion

A wide body of work has demonstrated privacy issues in general for machine learning models trained on personal data. Language models are among the most to suffer as they are capable of generating text which may potentially leak sensitive user content and lead to serious privacy violations.

Zhang et al. [2017] show that deep learning models can achieve perfect accuracy even on randomly labeled data. Such memorization capability may in fact be needed to achieve near-optimal accuracy on test data when the data distribution is long-tailed as recently shown by Feldman [2020], Brown et al. [2020b]. Unfortunately this can lead to a successful training data extraction attack, as in the case for the concurrent work [Carlini et al., 2020] that can recover individual training examples from the GPT-2 language model [Radford et al., 2019]. In their method, Carlini et al. [2020] generate a list of sequences by sampling from the GPT-2 language model and then curate it by using the perplexity measure. In a related line of work which exploits the increasingly common transfer learning setup, Zanella-Béguelin et al. [2020] have demonstrated that having simultaneous black box access to the pre-trained and fine-tuned language models allows them to extract rare sequences from the smaller and typically more sensitive fine-tuning dataset. Both attacks rely on the model output beyond top-1 or top-3 predictions along with the perplexity measure. Access to this information may easily be restricted in deployed language models. Nevertheless, there are serious privacy concerns since the attacks can extract personally identifiable information even if they are present in one document in the training data. We believe that our proposed procedure for privacy investigations of a language model trained on user content could be very beneficial to protect user-level privacy in the presence of such attacks.

On the other hand, Carlini et al. [2019] introduced the exposure metric to quantitatively assess the unintentional memorization phenomenon occurring in generative sequence models. They do so by inserting randomly-chosen canary sequences a varying number of times into the training data and measuring the relative difference in perplexity between inserted canaries and non-inserted random sequences. Our work is complementary in the sense that we are investigating the information leaked from user content in the training data, having in mind a strong threat model where one can query the language model with the precise context appearing in the training data. We believe that our proposed metrics along with the exposure metric can be employed together to provide strong privacy guarantees for a deployed language model.

Another line of work has studied the vulnerability of machine learning models to membership inference attack [Shokri et al., 2017, Yeom et al., 2018, Song and Shmatikov, 2019, Nasr et al., 2019, Long et al., 2018, Hayes et al., 01 Jan. 2019, Truex et al., 2018, Irolla and Châtel, 2019, Hisamoto et al., 2020, Salem et al., 2018, Sablayrolles et al., 2019, Leino and Fredrikson, Choquette-Choo et al., 2020]. The goal is to determine if a particular data record (or more generally data of a given user) belongs to the training set of the model. Although being an indirect leakage, membership inference is considered as a confidentiality violation and potential threat to the training data from models [Murakonda and Shokri, 2020].

The main framework with theoretical guarantees for user-level privacy is the application of differential privacy (DP) [Dwork, 2011] to model training. DP makes provable guarantees about the privacy of a stochastic function of a given dataset. Differentially private stochastic gradient descent (DP-SGD) has been developed and applied to training machine learning models [Song et al., 2013, Abadi et al., 2016]. This is an active area of research with the goal of pushing the frontiers of privacy-utility trade-off for deep neural networks.

7.1 Future work

We discuss a number of interesting future directions following our work:

- The proposed leakage report is based on central learning setting where the training data is stored at a central server. It would be interesting to solve the challenge of applying this method to other settings, such as federated learning [Kairouz et al., 2019] where machine learning models are trained on decentralized on-device data.
- We are hopeful that the metrics proposed in this work, as a first attempt to quantify user-level privacy leakage, would initiate further research on the topic, which will lead to further improvements on these metrics.
- It would be valuable to study the proposed methodology on more models/datasets, which would shed new lights on the protection of user-level privacy when language models are trained on confidential user content.

7.2 Conclusion

Recent results show that language models are capable of memorizing training samples under the hood of their impressive performance. This poses an immediate threat as leaking rare user content could lead to a privacy breach according to regulations such as GDPR, e.g. due to singling out of a user.

This work introduced a methodology to investigate information leaked by a language model from its training data in terms of privacy. We proposed metrics that could be used to quantify user-level privacy leakage and allow comparing models trained on the same data in terms of privacy. We believe our framework can be incorporated into the training platform of language models that would help assess the model from the perspective of privacy, along with its utility.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, and Melanie Subbiah. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020a.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The Secret Sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security 2019*, August 2019.

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

GDPR Article 29 Working Party. Opinion 05/2014 on "Anonymisation Techniques", 2014. URL `https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2014/wp216_en.pdf`.

Paul Lambert. SUBJECT: Write emails faster with Smart Compose in Gmail, 2018. URL `https://www.blog.google/products/gmail/subject-write-emails-faster-smart-compose-gmail`.

Mia Xu Chen, Benjamin N. Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M. Dai, Zhifeng Chen, Timothy Sohn, and Yonghui Wu. Gmail Smart Compose: Real-time assisted writing. KDD '19, page 2287–2295, New York, NY, USA, 2019.

Microsoft SwiftKey. URL https://www.microsoft.com/en-us/swiftkey.

Cynthia Dwork. Differential privacy. *Encyclopedia of Cryptography and Security*, 2011.

Shuang Song, Kamalika Chaudhuri, and Anand D. Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conf. on Signal and Information Processing*, pages 245–248, 2013.

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *ACM CCS*, 2016.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3:1137–1155, 2003.

Gboard. The Machine Intelligence Behind Gboard. URL https://ai.googleblog.com/2017/05/the-machine-intelligence-behind-gboard.html.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan "Honza" Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association*, pages 1045–1048, 2010.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. LSTM neural networks for language modeling. In *INTERSPEECH*, pages 194–197, 2012.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics.

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

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. XLNet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763, 2019.

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

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*, 2019.

Turing-NLG. A 17-billion-parameter language model by Microsoft, 2020. URL https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/.

White House Office of Science and Technology Policy (OSTP). Guidance for regulation of artificial intelligence applications, 2019. URL https://www.whitehouse.gov/wp-content/uploads/2020/01/Draft-OMB-Memo-on-Regulation-of-AI-1-7-19.pdf.

Vitaly Feldman. Does learning require memorization? A short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, page 954–959, New York, NY, USA, 2020.

Gavin Brown, Mark Bun, Vitaly Feldman, Adam Smith, and Kunal Talwar. When is memorization of irrelevant training data necessary for high-accuracy learning? *arXiv preprint arXiv:2012.06421*, 2020b.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China, November 2019. Association for Computational Linguistics.

M.E.J. Newman. Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, 46(5):323–351, 2005.

Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2016.

Douglas Oard, William Webber, David Kirsch, and Sergey Golitsynskiy. Avocado research email collection. `https://catalog.ldc.upenn.edu/LDC2015T03`, 2015.

Rami Al-Rfou, Marc Pickett, Javier Snaider, Y.-H. Sung, Brian Strope, and Ray Kurzweil. Conversational contextual cues: The case of personalization and history for response ranking, 2016.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling, 2013. URL `http://arxiv.org/abs/1312.3005`.

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

Nicolas Papernot, Steve Chien, Shuang Song, Abhradeep Thakurta, and Ulfar Erlingsson. Making the shoe fit: Architectures, initializations, and tuning for learning with privacy, 2020. URL `https://openreview.net/forum?id=rJg851rYwH`.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *ICLR*, 2017.

Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. Analyzing information leakage of updates to natural language models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, page 363–375, 2020.

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

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282, 2018.

Congzheng Song and Vitaly Shmatikov. Auditing data provenance in text-generation models. In *KDD*, 2019.

Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 739–753, 2019.

Yunhui Long, Vincent Bindschaedler, Lei Wang, Diyue Bu, Xiaofeng Wang, Haixu Tang, Carl A. Gunter, and Kai Chen. Understanding membership inferences on well-generalized learning models. *arXiv preprint arXiv:1802.04889*, 2018.

Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. LOGAN: Membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies*, 2019(1):133 – 152, 01 Jan. 2019.

Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. Towards demystifying membership inference attacks. *arXiv preprint arXiv:1807.09173*, 2018.

Paul Irolla and Grégory Châtel. Demystifying the membership inference attack. In *2019 12th CMI Conf. on Cybersecurity and Privacy (CMI)*, pages 1–7, 2019.

Sorami Hisamoto, Matt Post, and Kevin Duh. Membership inference attacks on sequence-to-sequence models: Is my data in your machine translation system? *TACL*, 8:49–63, 2020.

Ahmed Salem, Yang Zhang, Mathias Humbert, Mario Fritz, and Michael Backes. ML-Leaks: Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv:1806.01246*, 2018.

Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Herve Jegou. White-box vs black-box: Bayes optimal strategies for membership inference. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5558–5567, Long Beach, California, USA, 09–15 Jun 2019.

Klas Leino and Matt Fredrikson. Stolen Memories: Leveraging model memorization for calibrated white-box membership inference. *29th USENIX Security Symposium 2020.*

Christopher A. Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks. *arXiv preprint arXiv:2007.14321*, 2020.

Sasi Kumar Murakonda and Reza Shokri. ML Privacy Meter: Aiding regulatory compliance by quantifying the privacy risks of machine learning. *arXiv preprint arXiv:2007.09339*, 2020.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, and et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

A Tab attack for the DP-LM RanIni $\epsilon = \cdot$ models in Section 6.2

We present the leakage report for the unique sequences coming out of the tab attack for the DP-LM RanIni $\epsilon = \cdot$ models in Table 3, 4, and 5.

Table 3: Unique sequences from the tab attack for the DP-LM RanIni $\epsilon = 4.68$ model in Section 6.2.

$\mathcal{S}$	TOTAL # IN $\mathcal{D}$	USER # IN $\mathcal{D}$	CONTEXT LEN
"WAY , I DON'T THINK IT IS"	1	1	2
"THE TIME , I WOULD BE"	1	1	8
"SAME THING , I WOULD BE"	1	1	10
"MEDIA) IS NOT"	1	1	10
"NOT BE) BUT"	1	1	9

Table 4: Unique sequences from the tab attack for the DP-LM RanIni $\epsilon = 6.20$ model in Section 6.2. The $\text{PP}_{\text{lm}}(\cdot)$ column is the perplexity of each sequence with respect to the DP-LM RanIni $\epsilon = 6.20$ model. The $\text{PP}_{\text{public}}(\cdot)$ column is the perplexity of each sequence with respect to the public model. The last column is the ratio of the perplexities of the previous two columns. The worst-case leakage epsilon is $\epsilon_l = 1.89$. We refer the following model as the public model in this table. We remove the users that are the owners of these unique sequences and all of their data (not just these sequences) from the training data and train a new model with the remaining users. We consider the new model as a public model for the users of these unique sequences since it has never seen any data of these users during training.

$\mathcal{S}$	TOTAL # IN $\mathcal{D}$	USER # IN $\mathcal{D}$	CONTEXT LEN	$\text{PP}_{\text{lm}}(\cdot)$	$\text{PP}_{\text{public}}(\cdot)$	$\frac{\text{PP}_{\text{public}}(\cdot)}{\text{PP}_{\text{lm}}(\cdot)}$
"SAID , I THINK YOU SHOULD BE ABLE TO"	1	1	3	4.39	5.21	1.19
"ME A LINK TO YOUR POST ?"	1	1	4	3.33	3.49	1.05
"YOU FEEL BETTER , THEN YOU CAN"	1	1	4	5.4	5.53	1.02
"HAS ANY QUESTIONS OR CONCERNS , PLEASE"	1	1	5	5.1	8.87	1.74
"AS I KNOW , I THINK THE"	1	1	3	3.75	4.5	1.2
"COURT , HE WOULD HAVE"	1	1	8	4.46	5.22	1.17
"WANT * TO BE ?"	1	1	13	4.17	3.63	0.87
"LIKE IS THAT YOU ARE"	1	1	4	5.35	5.78	1.08
"OF PEOPLE) ARE"	1	1	6	5.14	4.21	0.82
"WARS , WE HAVE"	1	1	5	7.28	7.36	1.01
"WARS) IS"	1	1	7	3.53	6.69	**1.89**

Table 5: Unique sequences from the tab attack for the DP-LM RanIni $\epsilon = 26.4$ model in Section 6.2.

$\mathcal{S}$	TOTAL # IN $\mathcal{D}$	USER # IN $\mathcal{D}$	CONTEXT LEN
"THE OTHER HAND , I WOULD HAVE TO"	1	1	2
"LOT OF PEOPLE WHO DON'T KNOW ."	1	1	12
"THE CASE , THEN I WOULD HAVE"	1	1	4
"FAIR , I DON'T THINK YOU CAN"	1	1	3
"IDEA WHAT YOU ARE DOING , YOU"	1	1	5
"AS I KNOW , I HAVE A"	1	1	3
"TO DO , I WOULD HAVE TO"	1	1	10
"YOU FEEL BETTER , THEN YOU CAN"	1	1	4
"YOUR QUESTION , YOU CAN ONLY"	1	1	3
"OF CURIOSITY , I THINK THE"	1	1	2
"IDEA WHAT HE WAS DOING ?"	1	1	6
"DOING , YOU CAN GET A"	1	1	7
"COURT HAVE TO DO WITH THE"	1	1	7
"POINT , IT IS NOT ."	1	1	14
"DIFFERENT SITUATION , IT'S A"	1	1	6
"OF YEARS , I WAS"	1	1	4
"CHANGE , THEY WOULD BE"	1	1	6
"ASSAULT , I WOULD BE"	1	1	12
"* ARE * REALLY *"	1	1	8
"* DO IT , THEN"	1	1	4
"MEDIA) IS NOT"	1	1	10
"COURT , THEN THEY"	1	1	7
"BUTTER , I WOULD"	1	1	7
"FRANCISCO , I THINK"	1	1	6
"COURT HAVE TO BE"	1	1	5
"OF PEOPLE) ARE"	1	1	6
"% AGREE)."	1	1	15
"ARABIA * IS"	1	1	8
"ARABIA) IS"	1	1	7
"WARS) IS"	1	1	7