The Natural Auditor: How To Tell If Someone Used Your Words To Train Their Model

Congzheng Song
Cornell University
cs2296@cornell.edu

Vitaly Shmatikov
Cornell Tech
shmat@cs.cornell.edu

Abstract—To help enforce data-protection regulations such as GDPR and detect unauthorized uses of personal data, we propose a new model auditing technique that enables users to check if their data was used to train a machine learning model. We focus on auditing deep-learning models that generate natural-language text, including word prediction and dialog generation. These models are at the core of many popular online services. Furthermore, they are often trained on very sensitive personal data, such as users’ messages, searches, chats, and comments.

We design and evaluate an effective black-box auditing method that can detect, with very few queries to a model, if a particular user’s texts were used to train it (among thousands of other users). In contrast to prior work on membership inference against ML models, we do not assume that the model produces numeric confidence values. We empirically demonstrate that we can successfully audit models that are well-generalized and not overfitted to the training data. We also analyze how text-generation models memorize word sequences and explain why this memorization makes them amenable to auditing.

I. INTRODUCTION

Data-protection policies and regulations such as the European Union’s General Data Protection Regulation (GDPR) [11] give users the right to know how their data is processed. Enforcing this right requires the ability to audit data processors. As machine learning (ML) is becoming the core component of data processing in many offline and online services, and incidents such as DeepMind’s unauthorized use of NHS patients’ data to train ML models [4] illustrate the privacy risks of ML-based applications, it is essential to be able to audit the use of personal data in machine learning.

In this paper, we design and evaluate a technology that can help users audit ML models to determine if their data was used to train these models. We focus specifically on auditing models that generate natural-language text. Text-generation models for tasks such as next-word prediction (the basis of query autocompletion and predictive virtual keyboards) and dialog generation (the basis of chatbots and automated customer service) are extensively trained on sensitive personal data, including users’ messages, documents, chats, comments, and search queries. Our technology can help users audit a publicly available text-generation model and see if their words were used, perhaps without their permission, to create this model. Furthermore, our work sheds new light on how deep learning-based, text-generation models memorize their training data—a topic that has important implications both for data privacy and natural language processing.

The problem of auditing is closely related to the problem of membership inference (see Section II-C), but auditing text-generation models requires new technical machinery vs. membership inference in image-classification and categorical models.

First, we assume a very restrictive auditing scenario, which we believe matches how an individual user may audit a deployed ML-based service in practice. The auditor has only black-box model to the model and can query it only on a limited number of inputs. We assume that the model’s output does not include numeric probabilities or confidence values (deployed models rarely release these values). Furthermore, we consider scenarios where the model’s output is restricted to a relatively small list of words or even a single word. This precludes the application of most previously proposed membership inference methods.

Second, we work with text-generation models that are trained on the data of hundreds or thousands of users and are well-generalized, i.e., their accuracy on test inputs is not substantially different from their accuracy on training inputs. This precludes the application of membership inference methods that exploit the test-train accuracy gap exhibited by overfitted models.

Third, state-of-the-art text-generation models are based on recurrent neural networks (RNNs). We investigate how these models overfit to their training data, what signal this overfitting creates in their outputs, and how to exploit this signal for effective auditing. We show that the overfitting in text-generation models appears to manifest primarily via shifted probability distributions over the models’ output space. Specifically, we show that these models tend to assign significantly higher rank to relatively rare words when they appear in a familiar context (e.g., in a sentence seen during training). This does not affect the top-ranked, likeliest word generated by the model and therefore—in contrast to “conventional” overfitting—does not manifest in reduced test accuracy.

Fourth, we cannot assume that the auditor knows the distribution from which the training data for the target model was drawn. We demonstrate how to use auxiliary public datasets and cross-domain training for auditing.
Fifth, we focus on user-level auditing (vs. inferring training-dataset membership of individual records in prior work) and measure how many queries are needed to determine if the user’s data was used—possibly in combination with the data from thousands of other users—to train the model. We quantitatively show that sequences that include relatively rare words are more effective for auditing than word sequences randomly selected from the user’s data. We also measure the robustness of our auditing methodology to noise and errors in the test inputs used for auditing. This is important because the user may not know exactly which of his chats or online comments were used, or when the model creator may have started training on the user’s data.

Our black-box auditing methodology is very effective. In our experiments on the Reddit, SATED, and Movie dialogs tasks for, respectively, word prediction, translation, and dialog generation, it performs perfectly (i.e., its AUC score is 1) when the models are trained on the data of hundreds of users and the models’ outputs cover the entire vocabulary. Furthermore, it requires surprisingly few queries. If the auditor selects query sequences that include relatively rare words, a single query achieves AUC between 0.8 and 0.9 depending on the task, and 8 queries achieve almost perfect AUC.

If the word-prediction and dialog-generation models are restricted to generate and rank only the 500 likeliest words, AUC score of our auditor remains above 0.9. If the translation model generates a single word (as opposed to a ranked list of words), the auditor can still infer with a much-better-than-random probability if the model was trained on the word sequences of a particular user. For the Reddit word-prediction model, the auditor’s AUC score remains close to 0.9 even if the model was trained on the data of over 4,000 users. Furthermore, we empirically show that our auditing is robust to a significant amount of noise and errors in the audit queries. These results demonstrate that auditing modern text-generation models is feasible in realistic scenarios.

Finally, to explain why auditing works, we provide new insights into memorization in different types of text-generation models. For example, we demonstrate that deep learning-based translation models are more prone than the word-prediction models to memorize training sequences in their inner units.

II. BACKGROUND

A. Deep learning

Deep learning has achieved exceptional results on many natural language processing (NLP) tasks, including machine translation [39], question answering [34], text summarization [30], reading comprehension [14], and many more [21]. A deep learning model (or “artificial neural network”) is a function \( f_\theta : X \rightarrow Y \) parametrized by \( \theta \), where \( X \) is the input, or feature, space and \( Y \) is the output space. Supervised training of a model \( f_\theta \) aims to find the best set of parameters \( \theta \) using a labeled training dataset \( D = \{(x_i, y_i)\}_{i=1}^n \) and the loss metric \( L(f(x_i), y_i) \), which measures the gap between the model’s prediction \( f(x_i) \) and the correct output \( y_i \).

For ML tasks where the input space is discrete and sparse (e.g., text or location data), the standard approach is to transform discrete inputs into a lower-dimensional continuous vector representation, known as embedding or word embedding in the context of NLP tasks. For a text corpus with vocabulary \( V \), an embedding is a function \( E : V \rightarrow \mathbb{R}^{d_{emb}} \) where \( d_{emb} \), the dimension of the embedding, is a hyper-parameter.

In most NLP tasks, the input is a variable-length sequence of tokens \( x = [x_1, \ldots, x_l] \) in the embedding space. The output \( y \) can be either a class label (e.g., for sentiment analysis), a token (e.g., for next-word prediction), or a sequence of tokens (e.g., for machine translation).

Recurrent neural networks (RNNs) are a common neural-network architecture for modeling sequential input data. They are extensively used for text-generation tasks such as next-word prediction. An RNN maps the input sequence to a sequence of hidden representations \( a = [a^1, \ldots, a^l] \), where the computation of \( a^j \) is recursively dependent on the previous hidden representation \( a^{j-1} \) and the current input token \( x_j \). The base case \( a_0 \) is usually initialized as zeros. The simplest RNN can be formalized mathematically as follows:

\[
a^0 = 0, \quad a^j = \sigma(W_1 \cdot a^{j-1} + W_2 \cdot x^j + b)
\]

where \( \sigma \) is a non-linear activation function, and \( W_1, W_2, b \) are the parameters to be learned. RNN makes predictions by feeding these hidden representations to a classifier.

Sequence-to-sequence models are a common neural-network architecture for text-generation tasks where both the input \( x = [x^1, \ldots, x^l] \) and the output \( y = [y^1, \ldots, y^l] \) are sequences of tokens. A typical sequence-to-sequence model consists of an encoder RNN and a decoder RNN. The encoder RNN encodes the input sequence as \( a^l_{en} \), i.e., the last hidden representation produced by the encoder RNN. \( a^l_{en} \) is then passed to the decoder RNN as the initialization for \( a^0_{de} = a^l_{en} \). This enables the decoder to learn hidden representations for predicting the target sequence by taking into account the information from the input sequence. Similar to the next-word prediction RNNs, the decoder predicts words in the target language by feeding its hidden representations to a classifier.

B. Text-generation models

We focus on next-word prediction, machine translation, and dialog generation as representative text-generation tasks. Deep learning-based models for these tasks typically employ RNNs and sequence-to-sequence models as the basic building blocks.

Next-word prediction is used in many natural-language applications, including predictive virtual keyboards and query autocompletion. Given an input sequence \( x = [x^1, \ldots, x^j] \), the task is to predict the next token \( x^{j+1} \) from the context \( [x^1, \ldots, x^{j-1}] \). RNNs are commonly used for this task. The RNN model first obtains the hidden representation \( a^{j-1} \), i.e., the last hidden representation in the context sequence, and then?

We use “predict” and “generate” interchangeably when talking about text-generation models.
determine whether a specific data record is a member of $D$ or training of a machine learning model on dataset next-word prediction task, the loss function is the negative log likelihood:  

$$L(f(x), x) = -\sum_{j=1}^{l} \log f(x^1, \ldots, x^{j-1})$$

For the machine translation and dialog-generation tasks where the input is $x$ and the target is $y = [y^1, \ldots, y^l]$, the sequence-to-sequence model computes the probability $Pr(y^l|y^1, \ldots, y^{l-1}; x)$ as $f(y^1, \ldots, y^{l-1}; x)$. Similar to the next-word prediction task, the loss function is the negative log probability on the target sequence.

C. Membership inference

Membership inference attacks involve an adversary who observes the output of some computations over a hidden dataset $D$—for example, calculations of aggregate statistics or training of a machine learning model on $D$—and aims to determine whether a specific data record is a member of $D$. Successful membership inference attacks against aggregate statistics have been demonstrated in the context of genomic studies [3], [16], location time-series [28], and noisy statistics in general [9].

Membership inference attacks against ML models are surveyed in more detail in Section VII. They can be performed in a white-box or black-box setting. In the black-box setting, the adversary queries the model with a specific record and attempts to infer from the model’s outputs (e.g., probabilities assigned to different classes by a classification model) whether the record was among those used to train the model or not. For example, Shokri et al. [31] demonstrated a method for learning the statistical difference between the outputs of a classification model on members and non-members. The key technique—which inspires our approach in this paper—is to train a membership discriminator using the probability vector output by the model as the feature.

Auditing text-generation models involves an application of membership inference, but there are several critical differences that require the development of new membership inference techniques specifically adapted to the auditing scenarios and the idiosyncrasies of memorization in text-generation models. We discussed these differences in Section II and briefly summarize them here. In realistic auditing scenarios, (1) the auditor’s goal is user-level membership inference, as opposed to record-level membership inference which was the focus of prior work. (2) Deployed models that a user may wish to audit can be well-generalized and not overfitted to the training data in the conventional sense. Furthermore, (3) deployed models output only ranked lists of words and not the underlying numeric probabilities, removing the main feature that was used by previous membership inference attacks. Also, (4) the models’ outputs may be limited to only a few dozen or hundreds of words, as opposed to the entire vocabulary, and the auditor may be limited to a small number of queries. Finally, (5) the test inputs used by the auditor may be noisy or partially erroneous.

III. AUDITING TEXT-GENERATION MODELS

A. Problem statement

Consider a training dataset $D_{train}$ where each row is associated with an individual user, and let $U_{train}$ be the set of all users in $D_{train}$. The target model $f$ is trained on $D_{train}$ using a training protocol $T_{target}$, which includes the learning algorithm and the hyper-parameters that govern the training regime.

As described in Section II-B, a text-generation model $f$ takes as input a sequence of tokens $x$ and outputs a prediction $f(x)$ for a single token (if the task is next-word prediction) or a sequence of tokens (if the task is machine translation or dialog generation). The prediction $f(x)$ is a probability distribution or a sequence of distributions over the training vocabulary $V$ or a subset of $V$. We assume that the tokens in the model’s output space are ranked (i.e., the output distribution imposes an order on all possible tokens) but do not assume that the numeric probabilities from which the ranks are computed are available as part of the model’s output.

The goal of auditing is to infer user-level membership against the target model $f$, i.e., to decide whether a user $u \in U_{train}$ or not.

We assume that the auditor has black-box access to $f$: given an input query $x$, the auditor can observe $f(x)$. In realistic deployments of text-generation models, the auditor may not be able to observe the entire vector of ranked words $f(x)$ but only several top-ranked predictions. In our experiments in Section IV-C, we vary the size of the model’s output and show how it affects the accuracy of auditing.

We assume that the auditor knows the algorithm $T_{target}$, i.e., how $f$ was trained. The training algorithms for the
Train Audit Model

Audit Membership

Fig. 1: Auditing training data. (1) In the Train Audit Model phase, the auditor trains multiple shadow models \( f'_1, \ldots, f'_k \) using different train-test splits on the auxiliary reference data; the auditor then queries the shadow models with the train and test subsets and labels the predicted ranks output by the models as “member” and “nonmember”; the auditor then trains the audit model, which is a membership classifier that uses the labeled ranks as features. (2) In the Audit Membership phase, the auditor queries the black-box target model \( f \) with a sample of a particular user’s dataset \( D_u \); the target model outputs a predicted distribution on each word in the vocabulary; the auditor collects the ranks of the ground-truth words in the distributions and passes them to the audit model for determining whether \( D_u \) is used when training \( f \).

B. Overview of the auditing process

Algorithm 1 outlines the auditing process. Similar to standard membership inference against ML models \([31]\), the auditor’s goal is to learn to distinguish the outputs produced by the target model on sequences that the model “saw” during training and the outputs produced on sequences that it did not see during training. For this purpose, the auditor builds a binary user-level membership classifier \( f_{\text{audit}} \) that takes as input a (processed) list of predictions obtained by querying \( f \) with a subset of \( D_u \) and outputs a decision on \( u \in U_{\text{train}} \). We emphasize that the entire \( D_u \) need not be used for auditing. In Section IV-F, we show that surprisingly few queries can be sufficient to infer with very high accuracy whether or not the user’s data was used to train the target model.

To collect the data for training \( f_{\text{audit}} \), the auditor first trains \( k \) shadow models \( f'_1, \ldots, f'_k \) that “simulate” the target model. Our shadow training technique is inspired by \([31]\), but one essential distinction is that in our case the training data for the shadow models does not need to be drawn from the same distribution as the training data of the target model. This is important for real-world auditing because in practice the auditor may not know the entire distribution of the target’s data, and the API limits may prevent the auditor from querying the target state-of-the-art deep learning models are standard, but their hyper-parameters may be confidential. Recent work has shown how to accurately infer the training hyper-parameters from deployed ML models \([38]\).

The auditor needs an auxiliary reference dataset \( D_{\text{ref}} \); let \( U_{\text{ref}} \) be the user in \( D_{\text{ref}} \). We assume that \( D_{\text{ref}} \) is appropriately labeled so it can be used to train a model that performs the same text-generation task as \( f \). \( D_{\text{ref}} \) does not need to be drawn from the same distribution as \( D_{\text{train}} \). In Section IV-G, we show how public datasets can be used as \( D_{\text{ref}} \).
Algorithm 1: Auditing training data in text-generation models

Hyper-parameters: auditor’s reference dataset $D_{\text{ref}}$, number of shadow models $k$, user’s data $D_u$, target model $f$, target model-training protocol $T_{\text{target}}$, audit model-training protocol $T_{\text{audit}}$, maximum number of queries $m$, number of bins in histogram $d$

function AuditMembership()
    $f_{\text{audit}} \leftarrow \text{TrainAuditModel}()$
    $D_{\text{sample,}u} \leftarrow \text{SampleQueries}(m, D_u)$
    $h_u \leftarrow \text{HistogramFeature}(f, D_{\text{sample,}u})$
    return prediction of membership $f_{\text{audit}}(h_u)$

function SampleQueries($m, D$)
    if random sample then
        return randomly selected $m$ rows in $D$
    else
        $C \leftarrow \{\Sigma \text{(frequency of } w \text{ for } w \text{ in } y) \mid (x, y) \in D\}$
        $I \leftarrow \text{indices of } m \text{ smallest values in } C$
        return $m$ rows in $D$ indexed by $I$
    end if

function TrainAuditModel()
    $D_{\text{audit}} \leftarrow \emptyset$ \texttt{dataset for building the audit model}
    $U_{\text{ref}} \leftarrow$ users in $D_{\text{ref}}$
    for $i = 1$ to $k$ do
        $U_{\text{train}, \text{ref}}, U_{\text{test}, \text{ref}} \leftarrow \text{random split } U_{\text{ref}}$
        $D_{\text{train}} \leftarrow U\cup U_{\text{ref}}$
        Train a shadow model $f_i' \leftarrow T_{\text{target}}(D_{\text{train}})$.
        for every $u$ in users of $U_{\text{ref}}$ do
            $D_{\text{ref,}u} \leftarrow$ data in $D_{\text{ref}}$ associated with $u$
            $h_{u} \leftarrow \text{HistogramFeature}(f_i', D_{\text{ref,}u})$
            $z_{u} \leftarrow 1$ if $u \in U_{\text{ref-train}}$ else $0$
            $D_{\text{audit}} \cup \{(h_u, z_u)\}$
        end for
        end for
    Train the audit model $f_{\text{audit}} \leftarrow T_{\text{audit}}(D_{\text{audit}})$
    return $f_{\text{audit}}$

function HistogramFeature($f, D$)
    $R \leftarrow \{\text{rank}(y) \mid f(x) \forall (x, y) \in D\}$
    Initialize feature vector $h$ with $d$ entries.
    $b \leftarrow |V|/d$ \texttt{histogram bin size}
    for $i = 1$ to $d$ do
        $h_i = \{(i - 1) \cdot b \leq r < i \cdot b \mid r \in R\}$
    end for
    return feature vector $h$

The model repeatedly extracts sufficient data for the shadow-model training as done in [31].

As mentioned above, we assume that the auditor has an auxiliary reference dataset $D_{\text{ref}}$ with the same task labels as the dataset $D_{\text{target}}$ used to train the target model $f$. $D_{\text{ref}}$ can be collected from public sources. In Section IV-G, we show that the loss in audit accuracy when $D_{\text{train}}$ and $D_{\text{ref}}$ are drawn from different domains is negligible.

The auditor trains each shadow model on data associated with a randomly sampled user subset $U_{\text{train,ref}} \in U_{\text{ref}}$. As mentioned above, we assume that the auditor knows the hyper-parameters for training the target model $f$ and trains the shadow models using the same protocol $T_{\text{target}}$ and model architecture as $f$. As in [31], building multiple shadow models helps reduce the bias in their output patterns and thus train a well-generalized audit model.

The auditor then queries the shadow models with $D_{\text{ref,}u}$ for each $u$ in $U_{\text{ref}}$ and obtains the corresponding output predictions. The auditor labels the output predictions with the membership label “member” if $u$ was part of the sample chosen to train the shadow model in question, “non-member” otherwise. The labeled predictions are then used to train a binary membership classifier. We will refer to this classifier as the audit model $f_{\text{audit}}$.

In the auditing phase, the auditor samples data points from $D_u$ associated with some user $u$ and queries the target model $f$ with these points. The auditor then processes the target’s outputs on these points and passes them to $f_{\text{audit}}$, which decides whether $u \in U_{\text{train}}$ or not.

C. Signals for auditing text-generation models

Record-level membership inference typically uses the output probability distribution directly as the feature to distinguish between members and non-members. User-level membership inference in text-generation models calls for a different approach. Each user is associated with multiple texts, each of which has multiple words. Therefore, the auditor can obtain a collection of output predictions. On the negative side, the actual probabilities associated with each prediction may not be available.

As mentioned before, the output prediction $f(x)$ for an input $x$ is a probability distribution across the entire training vocabulary $V$, i.e., a $|V|$-dimensional probability vector. $|V|$ is generally large and the probability values are noisy. Instead of the raw probability values, we use the relative ranks of the words in the output distributions as signals for inferring user-level membership. As we will show in Section V even for a well-generalized model (i.e., its test-train performance difference is small), there is a substantial gap in the predicted rank of the same word when it appears in a training text and a test text. Specifically, the model ranks relatively rare words much higher when it sees them during testing in the same context as it saw them during training.

D. Training the audit model

Once the shadow models are trained, the auditor can collect their outputs on $U_{\text{train}}$ (members) and $U_{\text{ref},U_{\text{train}}}$ (nonmembers) by querying the shadow models with the corresponding split of $D_{\text{ref}}$. As discussed in Section III-C our auditor uses the predicted ranks of the ground-truth words as the features for classifying membership.
Function $\text{HistogramFeature}$ in Algorithm 1 describes the feature extraction process. Given a user $u$'s data $D_{\text{ref},u}$, the auditor queries the shadow model on each data point $(x,y) \in D_{\text{ref},u}$ and collects the ranks of $y$ in $f(x)$ into a rank set $R_u$. Taking English-to-French machine translation task as an example where $(x, y) = \text{I love you, Je t’aime}$, $f(x) = \{f(x)^1, f(x)^2\}$ is a sequence of two probability vectors for tokens “Je” and “t’aime.” The auditor collects the rank of the probability of “Je” in $f(x)^1$ (e.g., 2) and the rank of the probability of “t’aime” in $f(x)^2$ (e.g., 213), and adds [2, 213] to the rank set $R_u$. Rank 2 means that the word is the second likeliest prediction in the entire vocabulary. After collecting the ranks for all $(x,y) \in D_{\text{ref},u}$, the auditor builds a histogram for $R_u$ with a fixed number of bins $d$. The final feature vector $h_u$ is a $d$-way count vector where each entry is the count of the ranks in that bin.

For the user $u$, the auditor extracts features $h_u$ as described above and labels $h_u$ as 1 if $u \in U_{\text{train}}$ and 0 otherwise. The auditor repeats this procedure for each user in each shadow model and obtains a collection of labeled feature vectors $D_{\text{audit}}$.

Finally, the auditor trains a membership classifier $f_{\text{audit}}$ on $D_{\text{audit}}$ based on a suitable binary classification model. In our experiments, we use a linear SVM.

E. Auditing membership in the training data

At inference (i.e., audit) time, the auditor queries the target model $f$ with a sample of the user’s data $D_u$. If the number of queries to the target model $f$ is limited to $m$, we use the sampling strategy given in function $\text{SampleQueries}$ in Algorithm 1. The auditor can sample rows from $D_u$ at random but, as we show in Section IV-E, it is more effective to select $m$ test inputs that have the smallest frequency counts in their labels $y$, i.e., sequences with relatively rare words are more useful for auditing.

After querying $f$, the auditor processes the corresponding outputs and obtains a feature vector $h_u$ that describes the distribution of the predicted ranks for each word in $D_u$ using $\text{HistogramFeature}$. Finally, the auditor feeds $h_u$ to $f_{\text{audit}}$, which decides whether $u \in U_{\text{train}}$ or not.

IV. EXPERIMENTS

A. Datasets

The Reddit comments dataset (Reddit) is a randomly chosen month (November 2017) from the public Reddit dataset.

We filtered it to retain only the users with at least 150 but no more than 500 posts, for a total of 83,293 users with 247 posts each on average. We use the resulting dataset for the next-word prediction task.

The speaker annotated TED talks dataset (SATED) consists of transcripts from TED talks totaling roughly 271K sentences in each language distributed across 2,324 talks.

The collected dataset contains English-French (en-fr), English-German (en-de) and English-Spanish (en-es) language pairs and speaker annotation. We use the data from the en-fr pair for our experiments with the machine translation task.

The Cornell movie dialogs corpus (Movie dialogs) is a collection of fictional conversations extracted from raw movie scripts. There are a total of 220,579 conversational exchanges between pairs of characters engaging in at least 5 exchanges, involving 9,035 characters from 617 movies. We use this dataset for the dialogue-generation task.

Cross-domain reference datasets. As mentioned in Section III-A, the auditor may not have access to the exact dataset on which the target model was trained or even the distribution from which it was drawn. In this case, the auditor needs a reference dataset to train its shadow models for the same task as the target. In our experiments, we use public datasets for this purpose. As the cross-domain reference dataset for word prediction, we use the Wikitext-103 corpus obtained by a Wikipedia crawl. For translation, we use the English-French pair in the Europarl dataset, a parallel language corpus extracted from the proceedings of the European Parliament. For dialog generation, we use the Ubuntu dialogs dataset, which contains two-person technical support chat logs concerning various Ubuntu-related problems. Since these cross-reference datasets are not labeled with individual users, we split them into random $n_u$ subsets, each corresponding to an artificial “user.”

As demonstrated by our experiments, we can successfully train shadow models and produce effective audit models even with this artificial separation into users and even though the topics of the reference datasets are very different from the target models’ training datasets (e.g., technical support chats vs. conversations between movie characters).

B. Target models

Next-word prediction. We use a one-layer long short-term memory (LSTM) as the target model. LSTM is a more complicated RNN that can capture the long-term dependency in the sequence. The input sequence of tokens is first mapped to a sequence of embeddings. The embedding is then fed to the LSTM that learns a hidden representation for the context to predicting the next word.

Neural machine translation. We use a sequence-to-sequence target model with the attention module as described in [26]. Both the encoder and the decoder are one-layer LSTMs that operate on the embedding of source tokens and target tokens. The attention module adds an additional layer that operates on all hidden representations in the encoder LSTM and helps the decoder where to pay attention in the source texts when predicting a token in the target language.

Dialog generation. We use a sequence-to-sequence model without the attention module. The encoder and the decoder are one-layer LSTMs.
## Hyper-parameters.

We train the word-prediction model on the comments of 100 randomly selected users from the Reddit dataset. We set both the embedding dimension and LSTM hidden-representation size to 128. For training the LSTM, we use the Adam optimizer \[17\] with the learning rate set to 1e-3, batch size to 20, and the number of training epochs to 30.

We train the translation and dialog-generation models on 300 randomly selected users from the SATED and Movie dialogs, respectively. We set both the embedding dimension and LSTM hidden-representation size in the encoder and decoder to 128. We use the Adam optimizer with the learning rate set to 1e-3, batch size to 20, and the number of training epochs to 30.

For all datasets, we fix the vocabulary to the most frequent 5,000 tokens in the training texts. Tokens not in the vocabulary are replaced with a special `<UNK>` token. To prevent overfitting, we add dropout \[33\] with 0.5 rate to all hidden layers of all models.

### C. Performance of target models

To measure the performance of the target models and audit models, we randomly select a test set of users of the same size but disjoint from the set of users used for training.

We evaluate the performance of the target models using the word prediction accuracy and perplexity metrics, defined as:

\[
acc = \frac{1}{M} \sum_{i=1}^{n} \sum_{j=1}^{l_i} \mathbb{I}(\arg \max f(x_i)^j = y_i^j)
\]

\[
perp = 2^{-\frac{1}{\sum_{i=1}^{n} \sum_{j=1}^{l_i} \log f(x_i)^j / |y_i^j|}}
\]

where \(n\) is the number of data points in the training or test set, \(M = \sum_{i=1}^{n} l_i\) the sum of the number of tokens in all labels, \(\mathbb{I}\) is the indicator function that outputs 1 if the predicted token \(\arg \max f(x_i)^j\) equals the label token \(y_i^j\) and 0 otherwise, and \(f(x_i)^j[y_i^j]\) is the probability of predicting \(y_i^j\) in \(f(x_i)^j\). Perplexity is measured as 2 to the power of the entropy of the label predictions. The lower the perplexity, the better the model fits the data.

Table 1 shows the results. On Reddit, the accuracy of word prediction on the test set is 20%, which is similar to the numbers reported in \[25\] and higher than on the training set. On both SATED and Movie dialogs, the gaps between training and test accuracy are smaller than 5%, indicating that the models are not overfitted to the training set. On all datasets, the perplexity gaps are all within 15, which is relatively small.

### D. Performance of auditing

The goal of auditing is to distinguish between users that are in the training dataset (“members”) and users that are not (“nonmembers”). Therefore, for all experiments we report precision (the percentage of users classified by the audit model as “members” that are indeed members), recall (the percentage of members that are classified as “members” by the audit model), accuracy (the percentage of all users that are classified correctly by the audit model), and AUC, the area under the ROC curve that shows the gap between the scores (i.e., distances to the decision hyperplane of SVM) given by the audit model to members and nonmembers.

To train the shadow models, we sample a set of “shadow users,” disjoint from both training and test users. The number of shadow users is twice the number of training users. We use one half of the shadow users to train the shadow models and the other half to collect the shadow models’ outputs on the nonmembers of their training datasets (see Section II-B). We train 10 shadow models for the word-prediction task and 5 each for the translation and dialog-generation tasks. We use linear SVM implemented in LIBLINEAR \[10\] as the audit model with default hyper-parameters.

For the Reddit dataset, we report the performance of the audit model on 100 members and 100 nonmembers. On SATED and Movie dialogs, we report the performance on 300 members and nonmembers. Therefore, the baseline for all metrics is 0.5, corresponding to random guessing (since the number of members is equal to the number of nonmembers).

Our audit model achieves the perfect score (i.e., 1) on all metrics for all datasets and models when there is no restriction on the output size of the target models (i.e., they produce predictions over the entire vocabulary) and the auditor can query the target models any number of times.

### E. Effect of the number of users

To evaluate how the number of users in the training dataset affects the auditor’s ability to determine the presence of a single user, we train word-prediction models on 100, 500, 1,000, 2,000, 4,000, and 10,000 users selected from the Reddit dataset.

---

**Table 1:** Performance of target models on training and test data. Acc is word prediction accuracy, perp is perplexity.

| Dataset     | Training Acc | Test Acc  | Training Perp | Test Perp  |
|-------------|--------------|-----------|---------------|------------|
| Reddit      | 0.184        | 0.206     | 102.22        | 115.14     |
| SATED       | 0.587        | 0.535     | 6.36          | 10.28      |
| Movie dialogs | 0.283 | 0.264     | 45.57         | 61.11      |
dataset. Test users and shadow users are samples of the same size, all disjoint from each other.

Fig. 2 shows the results. When the number of users is under 1000, the auditor can achieve 0.95 score on all metrics. When the number of users is 4,000, precision drops to below 0.8 while AUC is still around 0.9. Audit performance drops more significantly when the number of users is 10,000, with precision and accuracy both below 0.7.

F. Effect of the number and selection of audit queries

As mentioned in Section III-E, the number of queries to the target model may be limited. To measure the performance of the audit model under this restriction, we vary the number of audit queries between 1, 2, 4, 8, 16, and 32 word sequences.

Fig 3 shows the results. With just 32 queries, audit performance exceeds 0.9 on all metrics for all datasets. When queries are randomly selected from the user’s data, the performance of the audit model is low if the number of queries is below 8. When queries are ordered by frequency, i.e., the auditor selects the user’s word sequences whose summary word-frequency counts are the lowest, even with a single query, the auditor can accurately determine if the user’s data was used to train a model on the Reddit or Movie dialogs dataset. This remarkable result demonstrates the extent to which text-generation models memorize the word sequences they were trained on, especially those that contain relatively rare words.

G. Effect of the size of the model’s output

In a realistic deployment of a text-generation model, its output may be limited to a few top-ranked words rather than the entire ranked vocabulary. We constrain the model’s output to the top-ranked 1, 5, 10, 50, 100, 500, and 1000 words, while the other hyper-parameters remain as in Section IV-B. When building the histogram feature vector for training the audit model (see Section III-D), we add an additional feature that counts how many times the ground-truth words are not among the top predictions output by the model.

Table III shows the results. On Reddit and Movie dialogs, the auditor’s performance is close to random guessing when the model’s outputs are limited to the top 50 or fewer words, increasing to above 0.9 when the output size is the top 500 words (only 10% of the entire vocabulary)—regardless of whether the shadow models are trained on the same domain as the target model or a different domain.

For the SATED translation task, audit performance is much higher than random guessing even when the model outputs just one top-ranked word. Audit performance exceeds 0.9 when the model outputs 50 top-ranked words (1% of the entire vocabulary). These results demonstrate the remarkable extent to which translation models memorize specific word sequences encountered in training.

H. Effect of noise and errors in the audit queries

At the time of auditing, $D_u$ used by the auditor might be noisy. Even though the user’s data was indeed used to train the target model, not all of the data points in $D_u$ might have been used in training (e.g., there is noise or errors in $D_u$, or only a fraction of $D_u$ was used).

We evaluate how this affects audit performance. For each training user, we use part of his data to train the target model and hold out the remaining fraction to represent noise during auditing. In the experiments, we vary the noise fraction between 0.1, 0.2, ..., 0.5.

Fig. 4 shows the results. For both SATED and Movie dialogs, recall drops significantly and is close to 0 for SATED when the fraction of noise is 50%. Not surprisingly, increasing the amount of noise biases the audit model towards misclassifying most training users as “nonmembers.” Precision and AUC of auditing remain high when noise increases. This may indicate that the scores of the membership classifier at the heart of the audit model still have a distinguishable gap between members and nonmembers, which is however not learned from the outputs of the shadow models queried with clean data (see Section III-B).

I. Auditing obfuscated data

Finally, we evaluate the effect of obfuscation and anonymization of text on the success of auditing. This experiment is the first step towards determining whether text-generation models memorize specific word sequences (which would not be preserved by obfuscation) rather than higher-level linguistic features (which might be).

We use a specific obfuscation technique, previously considered in the context of author attribution and evasion thereof [6], that machine-translates the text to a different language and back. We obfuscate the training and test users’ Reddit comments by using Google translation API[^6] and Yandex translation API[^7] to translate the English text to Japanese and back to English. Table III shows examples of obfuscated text.

Fig. 5 shows the results of auditing on obfuscated texts. For both Google- and Yandex-based obfuscation, audit accuracy drops to near random and recall is very low. AUC scores are still around 0.8, which is much higher than random guessing. This indicates there is still some useful signal in the model’s outputs on obfuscated texts, but the auditor’s membership classifier—which was trained on unobfuscated texts—fails to capture this signal.

This is yet another remarkable result given the poor quality of translation. In effect, even if the user’s text has been garbled almost to the point of incomprehensibility, there is still enough information remaining to enable the auditor to detect its presence in the training data in some cases.

V. OVERFITTING IN TEXT-GENERATION MODELS

As shown in Section IV, our auditor achieves nearly perfect performance against regularized text-generation models that are not “overfitted” as measured by their test-train accuracy gap—in contrast to the previous membership inference attacks that are most successful against overfitted models. In this section, we analyze why our auditing algorithm works so well.

[^6]: https://cloud.google.com/translate/
[^7]: https://tech.yandex.com/translate/
A. Word frequency and probability

As described in Section II-B, the loss function for the text-generation models that we consider is the negative log likelihood of the input sequence, which is the sum of negative log probabilities of the words in the sequence. By its very construction, this loss function appears to “encourage” the model to memorize word sequences that occur in the training data.

Fig. 6 shows the histograms of log probabilities of the more and less frequent words in the training (“train”) and test (“unseen”) sequences. For the more frequent words, the histograms are almost identical for the training and test sequences. For the less frequent words, the model fits worse for both training and test sequences as modes focus on smaller log probability values. Most importantly, there is a gap between the less frequent words in the training sequences and those in the test sequences. This gap indicates that the model assigns higher probabilities to words in the training sequences, producing a strong signal that can be used for membership inference and consequently auditing.

Fig. 3: Effect of the number of queries and sampling strategy. Plots on the left show the results when the auditor samples the user’s data for queries in the ascending order of frequency counts of tokens in the label; plots on the right show the results with randomly sampled data. The x-axis is the number of queries used for auditing, the y-axis is the audit performance score.
### TABLE II: Effect of the model’s output size.

| Reddit | Same Domain | Cross Domain |
|--------|-------------|--------------|
| # of output words | Accuracy | AUC | Precision | Recall | Accuracy | AUC | Precision | Recall |
| 1 | 0.545 | 0.549 | 0.574 | 0.350 | 0.505 | 0.589 | 0.667 | 0.020 |
| 5 | 0.550 | 0.572 | 0.553 | 0.520 | 0.490 | 0.525 | 0.495 | 0.920 |
| 10 | 0.580 | 0.602 | 0.582 | 0.570 | 0.500 | 0.552 | 0.500 | 0.950 |
| 50 | 0.605 | 0.648 | 0.606 | 0.600 | 0.505 | 0.659 | 0.503 | 0.980 |
| 100 | 0.725 | 0.788 | 0.765 | 0.650 | 0.585 | 0.714 | 0.549 | 0.950 |
| 500 | 0.970 | 0.998 | 0.970 | 0.970 | 0.905 | 0.992 | 0.988 | 0.820 |
| 1000 | 0.985 | 0.999 | 0.971 | 1.000 | 0.910 | 0.999 | 1.000 | 0.820 |

| SATED | | |
|--------|-------------|--------------|
| # of output words | Accuracy | AUC | Precision | Recall | Accuracy | AUC | Precision | Recall |
| 1 | 0.723 | 0.785 | 0.770 | 0.637 | 0.723 | 0.785 | 0.712 | 0.750 |
| 5 | 0.748 | 0.838 | 0.767 | 0.713 | 0.767 | 0.834 | 0.755 | 0.790 |
| 10 | 0.800 | 0.880 | 0.783 | 0.830 | 0.805 | 0.878 | 0.814 | 0.790 |
| 50 | 0.928 | 0.973 | 0.908 | 0.953 | 0.925 | 0.979 | 0.947 | 0.900 |
| 100 | 0.948 | 0.981 | 0.944 | 0.953 | 0.942 | 0.978 | 0.965 | 0.917 |
| 500 | 0.972 | 0.988 | 0.958 | 0.987 | 0.970 | 0.988 | 0.983 | 0.957 |
| 1000 | 0.960 | 0.984 | 0.939 | 0.983 | 0.967 | 0.985 | 0.973 | 0.960 |

| Movie dialogs | | |
| # of output words | Accuracy | AUC | Precision | Recall | Accuracy | AUC | Precision | Recall |
| 1 | 0.577 | 0.618 | 0.582 | 0.547 | 0.538 | 0.618 | 0.520 | 0.977 |
| 5 | 0.575 | 0.642 | 0.582 | 0.530 | 0.552 | 0.643 | 0.528 | 0.970 |
| 10 | 0.583 | 0.645 | 0.591 | 0.543 | 0.543 | 0.638 | 0.523 | 0.977 |
| 50 | 0.605 | 0.660 | 0.611 | 0.580 | 0.537 | 0.610 | 0.520 | 0.963 |
| 100 | 0.647 | 0.714 | 0.643 | 0.660 | 0.570 | 0.669 | 0.541 | 0.920 |
| 500 | 0.935 | 0.975 | 0.917 | 0.957 | 0.925 | 0.969 | 0.895 | 0.963 |
| 1000 | 0.972 | 0.995 | 0.955 | 0.990 | 0.962 | 0.992 | 0.948 | 0.977 |

**Fig. 4:** Effect of noise and errors. The x-axis is the fraction of noise data in $D_u$ to be audited, the y-axis is the audit performance score.

**Fig. 5:** Audit performance on obfuscated Reddit comments.

These histograms also demonstrate that **our text-generation models are not overfitted to their training datasets** in terms of the loss value. The 20% most frequent words correspond to 86.9% of the words in all of the training data and 88.1% of words in all of the test data in Reddit, 89.5% and 90.4% in SATED, and 93.1% and 94.1% in Movie dialogs. Consequently, these frequent words dominate the training and test loss value. Not surprisingly, text-generation models typically generate words from the top 20% of the word-frequency distribution. As long as the log probabilities remain similar for
Fig. 6: Histograms of log probabilities of words generated by our text-generation models. Probabilities for the words that appear in the training sequences are labeled “Train,” in test sequences as “Unseen.” The top row are the histograms for the most frequent words (top 20%), the bottom row are the histograms for the remaining 80%.

No obfuscation: I see so many adults that could benefit from this going around having themselves a big fat sugar snack or soda pop as a treat it’s so sad

Google: I saw so many adults who can benefit from cherishing big fat sugar snacks and soda pop and going around, it is very sad

Yandex: I think a lot of adults have benefited over your big fat candy and and handling of grief

TABLE III: Examples of texts obfuscated using Google translation API and Yandex translation API.

the top 20% words in both the training and test datasets, the training and test losses of the model will be similar. Therefore, the models are not overfitted.

B. Word frequency and predicted rank

Memorization of training sequences produces a much stronger signal in the relative rank assigned by the model to the candidate words in the model’s output vocabulary. Fig. 7 shows the relationship between a word’s rank in the frequency table of the training corpus and its rank in the model’s predictions. A smaller rank number indicates that the word is ranked higher in the vocabulary, i.e., more frequent in the corpus or more likely to be predicted by the model. On all datasets, less frequent words exhibit a much bigger gap between the rank predicted by the model when the word appears in a training sequence and when it appears in a test sequence. This explains why our auditing algorithm is more successful when it queries the target model with sequences consisting of the less frequent words (see Section IV-F).

C. Ablation analysis

In Sections V-A and V-B we demonstrated that log probabilities and ranks assigned by the text-generation models exhibit a gap between the training and test sequences for the less-frequent words but not for the most-frequent words. We hypothesize that these models learn generalizable patterns for the most-frequent words while hard-memorizing the sequences consisting of the less-frequent words.

To gather preliminary evidence for this hypothesis, we carried out an experiment based on ablation analysis that was recently proposed to detect memorization in deep-learning models [27]. The latter work empirically shows that as more hidden units are ablated, accuracy on the training data degrades quicker for models that are hard-memorizing the training data.

We train target models without dropout (since dropout ablates the hidden units during training) on Reddit and SATED, keeping the other hyper-parameters the same as in Section IV-B. We randomly set a fraction of the model’s
hidden representation to zero and evaluate the accuracy of word prediction on the training data. We vary the fraction from 0.1 to 0.5 on Reddit and 0.1 to 0.9 on SATED and report the accuracy score separately for the 10% most frequent words and the remaining 90%.

Fig. 8 shows the results. When no hidden units are ablated, the accuracy is similar for the most-frequent words and the rest. As the fraction of ablated units increases, accuracy on the less-frequent words drops more significantly than on the most-frequent words. This indicates that predicting less-frequent words is more dependent on specific hidden units in the model and thus involves more memorization.

VI. LIMITATIONS

Models trained on a very large number of users. In some industrial implementation of text-generation models [24], [25], the number of users is on the scale of millions. As we have shown in Section V-F, performance of our auditor starts to drop when the number of users reaches 10,000. We expect that our black-box algorithm will not be able to audit models trained on a very large number (dozens or hundreds of thousands) of users. That said, (a) many state-of-the-art models are trained on fewer than 10,000 users [20], [26], [42], and (b)
white-box auditing techniques may be effective even against models trained on dozens of thousands of users. This is a topic for future work.

**Differentially private models.** In theory, user-level differential privacy (DP) is a direct countermeasure to user-level membership inference. We used federated learning with differential privacy as proposed by McMahan et al. \[25\] to train a next-word prediction model on the Reddit dataset. We set the number of users to 5,000, user sampling rate to 0.04 in a single round of federated learning, $L_2$ bound on a single user’s contribution to 10.0, and the other hyper-parameters the same as in \[25\]. This produced an $(\epsilon, \delta)$-DP model with $\epsilon = 4.129$ and $\delta = 1e - 4$ after 300 rounds of training. This model achieves 15% word prediction accuracy, similar to the accuracy reported in \[25\]. By contrast, the accuracy of our non-DP model is 20% when trained on only 100 users’ data, i.e., the differentially private model is significantly less accurate than the non-DP one. Our auditing algorithm fails against the differentially private model, with performance scores near 0.5 (equivalent to random guessing).

To further investigate the predictive power of the differentially private model, Fig. 9 plots the ranks of words in the vocabulary (based on their frequencies) and in the model’s predictions. The predicted rank is larger than the frequency rank for the 50% most frequent words and remains around 3,000 for the other 50%. The predicted rank is very similar for the words in the training and test sequences, which explains why auditing fails.

The plot also suggests that the differentially private model will almost always predict common words and hardly ever predict relatively rare words. While it does not appear that the model memorizes its training data, it is not clear to what extent it generalizes.

VII. RELATED WORK

**Membership inference.** As discussed in Section II-C, release of aggregate statistics about private data is generically vulnerable to membership inference attacks \[9\]. Membership inference against ML models was studied in \[13\], \[22\], \[29\], \[31\].

Shokri et al. \[31\] developed membership inference techniques against black-box models, exploiting the differences in the models’ outputs on training and non-training inputs. Their attack performs best when the target model is overfitted to the training data. Truex et al. \[37\] extend and generalize this work, including white-box and federated-learning setting.

Rahman et al. \[29\] extend the membership inference attack to differentially private ML models and show that the model may need to sacrifice its test accuracy to achieve membership privacy. Hayes et al. \[13\] studied membership inference attacks against generative models. Long et al. \[22\] showed that well-generalized models can leak membership information, but their attack requires the adversary to first identify a handful of vulnerable records in the training dataset. Yeom et al. \[40\] formalized membership inference and theoretically showed that overfitting is sufficient but not necessary.

We believe that our auditing method is the first positive application of membership inference. Furthermore, it’s the first example of user-level membership inference against ML models and the first that targets text-generation models, which are different and more complex than the models previously considered in the membership inference literature. Other innovations vs. prior methods were summarized in Sections I and II-C, such as the fact that our methods work with well-generalized methods that do not output numeric probabilities.

**Memorization in ML models.** Zhang et al. \[41\] showed that deep learning models can achieve perfect accuracy even on randomly labeled training data. Song et al. \[32\] presented malicious training algorithms that intentionally encode the training data in the parameters of the model without affecting its accuracy on the main task. By contrast, we demonstrate that popular text-generation models unintentionally memorize training data in their hidden activation units and outputs.

Carlini et al. \[7\] showed that deep learning-based generative sequence models unintentionally memorize training data, and that an adversary can extract specific numbers with black-box access to the model given some prior knowledge about the format of the input (e.g., a credit card number). For a text-generation model, numbers are essentially random data, thus this is another illustration that models memorize random data. By contrast, we show that text-generation models memorize even words and sentences that are directly related to their primary task—without a negative impact on their test accuracy—and leverage this into an effective auditing method.

**User-level differential privacy.** As discussed in Section VI, user-level differential privacy (DP) bounds the influence of any single user on the model. McMahan et al. proposed a differentially private federated learning algorithm for training
language models [25]. Geyer et al. proposed a similar algorithm and evaluated it on an image classification model [12]. Bonawitz et al. designed a secure aggregation algorithm for distributed training that hides users’ contributions and can be used in conjunction with a differentially private mechanism to provide stronger guarantees [5].

Auditing target models trained with user-level differential privacy is expected to fail. However, given the current state of the art in differentially private ML algorithms, a massive number of users (at least 10,000) is needed to create differentially private models that achieve reasonable accuracy on their tasks. How to build accurate differentially private models with fewer users remains an open research question.

**Auditing ML models.** Much recent work aims to understand the behavior of ML models with black-box access [1]. [19]. These approaches improve the interpretability of the model by showing how features or training data points influence the model’s predictions. Another line of model-auditing research focuses on detecting bias and discrimination in models or data-driven applications [2], [35], [36]. We are not aware of any prior work that aims to audit the use of specific data sources to train a model.

VIII. CONCLUSION

Deep learning-based, text-generation models for word prediction, translation, and dialog generation are the core components of many popular online services. We demonstrated that these models memorize their training data. This memorization does not appear to manifest in reduced test accuracy, which is a symptom of “conventional” overfitting, but is reflected instead in how they rank the candidate words they generate.

We developed a black-box auditing method that enables users to check if their chats, messages, or comments have been used to train someone else’s model. Our auditing method, based on a new flavor of membership inference that exploits memorization in text-generation models, is very effective. It works even if the model’s output is limited to a few dozen or hundreds of ranked words (without numeric probabilities) and if the auditor is limited to a few queries. For translation models, even a single query can be sufficient to detect with high confidence that the user’s data has been used to train the model. More powerful auditing algorithms may be possible if the auditor has access to the model’s parameters and can observe its internal representations rather than just output predictions. This is a topic for future work.

We view the results of this paper as essentially positive, demonstrating how memorization in ML models can help detect unauthorized uses of sensitive personal data and ensure compliance with GDPR and other data-protection policies and regulations.

**Acknowledgments.** This research was supported in part by the NSF grants 1611770 and 1704296 and the generosity of Eric and Wendy Schmidt by recommendation of the Schmidt Futures program.

**REFERENCES**

[1] Philip Adler, Casey Falk, Sorelle A Friedler, Tonney Nix, Gabriel Rybeck, Carlos Scheidegger, Brandon Smith, and Suresh Venkatasubramanian. Auditing black-box models for indirect influence. Knowledge and Information Systems, 54(1):95–122, 2018.
[2] Aws Albarghouthi, Loris D’Antoni, Samuel Drews, and Aditya V Nori. FairSquare: Probabilistic verification of program fairness. In OOPSLA, 2017.
[3] Michael Backes, Pascal Berrang, Mathias Humbert, and Praveen Manoharan. Membership privacy in microRNA-based studies. In CCS, 2016.
[4] BBC. Google DeepMind NHS app test broke UK privacy law. [https://www.bbc.com/news/technology-40483202], 2017.
[5] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In CCS, 2017.
[6] Michael Brennan, Sadia Afroz, and Rachel Greenstadt. Adversarial stylometry: Circumventing authorship recognition to preserve privacy and anonymity. ACM Transactions on Information and System Security (TISSEC), 15(3):12, 2012.
[7] Nicholas Carlini, Chang Liu, Jernej Kos, Ulfar Erlingsson, and Dawn Song. The Secret Sharer: Measuring unintended neural network memorization & extracting secrets. arXiv:1802.08322, 2018.
[8] Cristian Danescu-Niculescu-Mizil and Lillian Lee. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialog. In Workshop on Cognitive Modeling and Computational Linguistics, ACL, 2011.
[9] Cynthia Dwork, Adam Smith, Thomas Steinke, Jonathan Ullman, and Salil Vadhan. Robust traceability from trace amounts. In FOCS, 2015.
[10] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. JMLR, 9(Aug):1871–1874, 2008.
[11] General Data Protection Regulation. [https://en.wikipedia.org/wiki/General_Data_Protection_Regression], 2018.
[12] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. arXiv:1712.07557, 2017.
[13] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. LOGAN: Evaluating privacy leakage of generative models using generative adversarial networks. arXiv:1705.07663, 2017.
[14] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In NIPS, 2015.
[15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural Computation, 9(8):1735–1780, 1997.
[16] Nils Homer, Szabolcs Stelzinger, Margot Redman, David Duggan, Waibhav Tenbe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. PLoS genetics, 4(8):e1000167, 2008.
[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv:1412.6980, 2014.
[18] Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In MT Summit, volume 5, 2005.
[19] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In ICML, 2017.
[20] Satwik Kottur, Xiaoyu Wang, and Vitor R Carvalho. Exploring personalized neural conversational models. In IJCAI, 2017.
[21] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. Nature, 521(7553):436, 2015.
[22] Yunhui Long, Vincent Bindschaedler, Lei Wang, Diyue Bu, Xiaofeng Wang, Haixu Tang, Carl A Gunter, and Kai Chen. Understanding data distributions in machine learning. In PCI, 2017.
[23] Ryan Lowe, Nissan Pow, Julian V Serban, and Joelle Pineau. The Ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In SIGDIAL, 2015.
[24] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In AISTATS, 2017.
[25] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private language models without losing accuracy. arXiv:1710.06963, 2017.
[26] Paul Michel and Graham Neubig. Extreme adaptation for personalized neural machine translation. arXiv:1805.01817, 2018.
[27] Ari S Morcos, David GT Barrett, Neil C Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. arXiv:1803.06959, 2018.
[28] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. Knock knock, who’s there? Membership inference on aggregate location data. In NDSS, 2018.
[29] Md Atiqur Rahman, Tanzila Rahman, Robert Laganière, Noman Mohammed, and Yang Wang. Membership inference attack against differentially private deep learning model. Transactions on Data Privacy, 11(1):61–79, 2018.
[30] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In EMNLP, 2015.
[31] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In S&P, 2017.
[32] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In CCS, 2017.
[33] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. JMLR, 15:1929–1958, 2014.
[34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In NIPS, 2015.
[35] Sarah Tan, Rich Caruana, Giles Hooker, and Yin Lou. Detecting bias in black-box models using transparent model distillation. arXiv:1710.06169, 2017.
[36] Florian Tramèr, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, Jean-Pierre Hubaux, Mathias Humbert, Ari Juels, and Huang Lin. FairTest: Discovering unwarranted associations in data-driven applications. In EuroS&P, 2017.
[37] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. Towards demystifying membership inference attacks. arXiv:1807.09173, 2018.
[38] Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. arXiv:1802.05351, 2018.
[39] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google’s neural machine translation system: Bridging the gap between human and machine translation. arXiv:1609.08144, 2016.
[40] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In CSF, 2018.
[41] Chiyuan Zhang, Sany Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In ICLR, 2017.
[42] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too? In ACL, 2018.