Information Leakage in Embedding Models

Congzheng Song*
Cornell University
cs2296@cornell.edu

Ananth Raghunathan*
Facebook
ananthr@cs.stanford.edu

ABSTRACT

Embeddings are functions that map raw input data to low-dimensional vector representations, while preserving important semantic information about the inputs. Pre-training embeddings on a large amount of unlabeled data and fine-tuning them for downstream tasks is now a de facto standard in achieving state of the art learning in many domains.

We demonstrate that embeddings, in addition to encoding generic semantics, often also present a vector that leaks sensitive information about the input data. We develop three classes of attacks to systematically study information that might be leaked by embeddings. First, embedding vectors can be inverted to partially recover some of the input data. As an example, we show that our attacks on popular sentence embeddings recover between 50%–70% of the input words (F1 scores of 0.5–0.7). Second, embeddings may reveal sensitive attributes inherent in inputs and independent of the underlying semantic task at hand. Attributes such as authorship of text can be easily extracted by training an inference model on just a handful of labeled embedding vectors. Third, embedding models leak moderate amount of membership information for infrequent training data inputs. We extensively evaluate our attacks on various state-of-the-art embedding models in the text domain. We also propose and evaluate defenses that can prevent the leakage to some extent at a minor cost in utility.

CCS CONCEPTS

• **Security and privacy → Software and application security**.

KEYWORDS

machine learning, embeddings, privacy

ACM Reference Format:
Congzheng Song and Ananth Raghunathan. 2020. Information Leakage in Embedding Models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20), November 9–13, 2020, Virtual Event, USA*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3372297.3417270

1 INTRODUCTION

Machine learning (ML) has seen an explosive growth over the past decade and is now widely used across industry from image analysis [24, 34], speech recognition [22], and even in applications in the medical sciences for diagnosis [8, 56]. These advances rely on not only improved training algorithms and architectures, but also access to high-quality and often sensitive data. The wide applications of machine learning and its reliance on quality training data necessitates a better understanding of how exactly ML models work and how they interact with their training data. Naturally, there is increasing literature focused on these aspects, both in the domains of interpretability and fairness of models, and their privacy. The latter is further of timely importance given recent regulations under the European Union's General Data Protection Regulation (GDPR) umbrella requiring users to have greater control of their data.

Applying a line of research investigating whether individual genomic records can be subsequently identified [28, 62], Shokri et al. [66] developed the first *membership inference* tests investigating how ML models may leak some of their training data. This subsequently led to a large body of work exploring this space [45, 48, 60, 61, 77]. Another research direction investigating how models might memorize their training data has also shown promising results [4, 67]. Apart from training data privacy, modern deep learning models can unintentionally leak information of the sensitive input from the model's representation at inference time [9, 18, 38, 50, 68].

This growing body of work analyzing models from the standpoint of privacy aims to answer natural questions: *Besides the learning task at hand, what other information do models capture or expose about their training data?* And, *what functionalities may be captured by ML models unintentionally?* We also note that beyond the scope of this paper are other rich sets of questions around adversarial behavior with ML models—the presence of adversarial examples, data poisoning attacks, and more.

Embeddings. In most of the research investigating the privacy of ML models, one class of models is largely conspicuously absent. Embeddings are mathematical functions that map raw objects (such as words, sentences, images, user activities) to real valued vectors with the goal of capturing and preserving some important semantic meaning about the underlying objects. The most common application of embeddings is *transfer learning* where the embeddings are pre-trained on a large amount of unlabeled raw data and later fine-tuned on downstream tasks with limited labeled data. Transfer learning from embeddings have been shown to be tremendously useful for many natural language processing (NLP) tasks such as paraphrasing [14], response suggestion [30], information retrieval, text classification, and question answering [57], where labeled data is typically expensive to collect. Embeddings have also been successfully applied to other data domains including social networks [21], source code [2], YouTube watches [11], movie feedback [25], locations [12], etc. In some sense, their widespread use should not be surprising—embedding models and their success illustrate why deep learning has been successful at capturing interesting semantics from large quantities of raw data.

*Work done while at Google Brain.

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

ACM ISBN 978-1-4503-7089-9/20/11.
https://doi.org/10.1145/3372297.3417270

Embedding models are often pre-trained on raw and unlabeled data at hand, then used with labeled data to transfer learning to various downstream tasks. Our study of embeddings is motivated by their widespread application and trying to better understand how these two stages of training may capture and subsequently leak information about sensitive data. While it is attractive that these models inherently capture relations and similarities and other semantic relationships between raw objects like words or sentences, it also behooves one to consider if this is *all* the information being captured by the embeddings. Do they, perhaps, in addition to meaning of words inadvertently capture information about the authors perhaps? Would that have consequences if these embeddings are used in adversarial ways? What aspects of the seminal research into privacy of ML models through the lens of membership inference attacks and memorization propensity can we apply to better understand the privacy risks of embeddings?

These questions and more lead us to initiate a systematic study of the privacy of embedding models by considering three broad classes of attacks. We first consider *embedding inversion* attacks whose goal is to invert a given embedding back to the sensitive raw text inputs (such as private user messages). For embeddings susceptible to these attacks, it is imperative that we consider the embedding outputs containing inherently as much information with respect to risks of leakage as the underlying sensitive data itself. The fact that embeddings appear to be abstract real-numbered vectors should not be misconstrued as being safe.

Along the lines of Song and Shmatikov [68], we consider *attribute inference* attacks to test if embeddings might unintentionally reveal attributes of their input data that are sensitive; aspects that are not useful for their downstream applications. This threat model is particularly important when sensitive attributes orthogonal to downstream tasks are quickly revealed given very little *auxiliary data*, something that the adversary might be able to capture through external means. As an example, given (sensitive) demographic information (such as gender) that an adversary might retrieve for a limited set of data points, the embeddings should ideally not enable even somewhat accurate estimates of these sensitive attributes across a larger population on which they might be used for completely orthogonal downstream tasks (e.g. sentiment analysis).

Finally, we also consider classic *membership inference* attacks demonstrating the need to worry about leakage of training data membership when given access to embedding models and their outputs, as is common with many other ML models.

Our Contributions. As discussed above, we initiate a systematic and broad study of three classes of potential information leakage in embedding models. We consider word and sentence embedding models to show the viability of these attacks and demonstrate their usefulness in improving our understanding of privacy risks of embeddings. Additionally, we introduce new techniques to attacks models, in addition to drawing inspiration from existing attacks to apply them to embeddings to various degrees. Our results are demonstrated on widely-used word embeddings such as Word2Vec [46], FastText [3], GloVe [54] and different sentence embedding models including dual-encoders [39] and BERT [13, 35].

(1) We demonstrate that sentence embedding vectors encode information about the exact words in input texts rather than merely abstract semantics. Under scenarios involving black-box and white-box access to embedding models, we develop several inversion techniques that can invert the embeddings back to the words with high precision and recall values (exceeding 60% each) demonstrating that a significant fraction of the inputs may be easily recovered. We note that these techniques might be of independent interest.

(2) We discover certain embeddings training frameworks in particular favor *sensitive attributes* leakage by showing that such embedding models improve upon state-of-the-art stylometry techniques [59, 65] to infer text authorship with very little training data. Using dual encoder embeddings [39] trained with contrastive learning framework [63] and 10 to 50 labeled sentences per author, we show 1.5–3× fold improvement in classifying hundreds of authors over prior stylometry methods and embeddings trained with different learning paradigms.

(3) We show that membership inference attacks are still a viable threat for embedding models, albeit to a lesser extent. Given our results that demonstrate adversary can achieve a 30% improvement on membership information over random guessing on both word and sentence embeddings, we show that is prudent to consider these potential avenues of leakage when dealing with embeddings.

(4) Finally, we propose and evaluate adversarial training techniques to minimize the information leakage via inversion and sensitive attribute inference. We demonstrate through experiments their applicability to mitigating these attacks. We show that embedding inversion attacks and attribute inference attacks against our adversarially trained model go down by 30% and 80% respectively. These come at a minor cost to utility indicating a mitigation that cannot simply be attributed to training poorer embeddings.

Paper Organization. The rest of the paper is organized as follows. Section 2 introduces preliminaries needed for the rest of the paper. Sections 3, 4, and 5 cover attacks against embedding models—inversion attacks, sensitive attributes inference attacks, and membership inference attacks respectively. Our experimental results and proposed defenses are covered in Sections 6 and 7 respectively, followed by related work and conclusions.

2 PRELIMINARIES

2.1 Embedding Models

Embedding models are widely used machine learning models that map a raw input (usually discrete) to a low-dimensional vector. The embedding vector captures the semantic meaning of the raw input data and can be used for various downstream tasks including nearest neighbor search, retrieval, and classification. In this work, we focus on embeddings of text input data as they are widely used in many applications and have been studied extensively in research community [3, 5, 13, 33, 35, 39, 46, 54, 55, 58].

Word embeddings. Word embeddings are look-up tables that map each word w from a vocabulary $\mathcal{V}$ to a vector $\boldsymbol{v}_w \in \mathbb{R}^d$. Word embedding vectors capture the semantic meaning of words in the following manner: words with similar meanings will have small cosine distance in the embedding space, the cosine distance of $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ defined as $1 - (\boldsymbol{v}_1^\top \cdot \boldsymbol{v}_2 / \|\boldsymbol{v}_1\| \|\boldsymbol{v}_2\|)$.

Popular word embedding models including Word2Vec [46], FastText [3] and GloVe [54] are learned in an unsupervised fashion on

a large unlabeled corpus. In detail, given a sliding window of words $C = [w_b, \dots, w_0, \dots, w_e]$ from the training corpus, Word2Vec and FastText train to predict the context word w_i given the center word w_0 by maximizing the log-likelihood $\log P_{\boldsymbol{V}}(w_i|w_0)$ where

$$P_{\boldsymbol{V}}(w_i|w_0) = \frac{\exp(\boldsymbol{v}_{w_i}^\top \cdot \boldsymbol{v}_{w_0})}{\sum_{w \in \{w_i\} \cup \mathcal{V}_{\text{neg}}} \exp(\boldsymbol{v}_w^\top \cdot \boldsymbol{v}_{w_0})} \tag{1}$$

for each $w_i \in C/\{w_0\}$. To accelerate training, the above probability is calculated against $\mathcal{V}_{\text{neg}} \subset \mathcal{V}$, a sampled subset of words not in C instead of the entire vocabulary. GloVe is trained to estimate the co-occurrence count of w_i and w_j for all pairs of $w_i, w_j \in C$.

A common practice in modern deep NLP models is to use word embeddings as the first layer so that discrete inputs are mapped to a continuous space and can be used for later computation. Pre-trained word embeddings is often used to initialize the weights of the embedding layer. This is especially helpful when the downstream NLP tasks have a limited amount of labeled data as the knowledge learned by these pre-trained word embeddings from a large unlabeled corpus can be transferred to the downstream tasks.

Sentence embeddings. Sentence embeddings are functions that map a variable-length sequence of words x to a fix-sized embedding vector $\Phi(x) \in \mathbb{R}^d$ through a neural network model Φ. For a input sequence of ℓ words $x = [w_1, w_2, \dots, w_\ell]$, Φ first maps x into a sequence of word vectors $\boldsymbol{X} = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_\ell]$ with a word embedding matrix $\boldsymbol{V}$. Then Φ feeds $\boldsymbol{X}$ to a recurrent neural networks or Transformer [70] and obtain a sequential hidden representation $[\boldsymbol{h}_1, \boldsymbol{h}_2, \dots, \boldsymbol{h}_\ell]$ for each word in x. Finally Φ outputs the sentence embedding by reducing the sequential hidden representation to a single vector representation. Common reducing methods include taking the last representation where $\Phi(x) = \boldsymbol{h}_\ell$ and mean-pooling where $\Phi(x) = (1/\ell) \cdot \sum_{i=1}^{\ell} \boldsymbol{h}_i$.

Sentence embedding models are usually trained with unsupervised learning methods on a large unlabeled corpus. A popular architecture for unsupervised sentence embedding is the dual-encoder model proposed in many prior works [5, 7, 26, 39, 40, 58, 75]. The dual-encoder model trains on a pair of context sentences (x_a, x_b) where the pair could be a sentence and its next sentence in the same text or a dialog input and its response, etc. Given a randomly sampled set of negative sentences $\mathcal{X}_{\text{neg}}$ that are not in the context of x_a, x_b, the objective of training is to maximizes the log-likelihood $\log P_\Phi(x_b|x_a, \mathcal{X}_{\text{neg}})$ where

$$P_\Phi(x_b|x_a, \mathcal{X}_{\text{neg}}) = \frac{\exp(\Phi(x_b)^\top \cdot \Phi(x_a))}{\sum_{x \in \{x_b\} \cup \mathcal{X}_{\text{neg}}} \exp(\Phi(x)^\top \cdot \Phi(x_a))}. \tag{2}$$

Intuitively, the model is trained to predict the correct context x_b from the set $\{x_b\} \cup \mathcal{X}_{\text{neg}}$ when conditioned on x_a. In other words, the similarity between embeddings of context data is maximized while that between negative samples is minimized.

Sentence embeddings usually outperform word embeddings on transfer learning. For downstream tasks such as image-sentence retrieval, classification, and paraphrase detection, sentence embeddings are much more efficient for the reason that only a linear model needs to be trained using embeddings as the feature vectors.

Pre-trained language models. Language models are trained to learn contextual information by predicting next words in input text, and pre-trained language models can easily adopt to other NLP tasks by fine-tuning. The recently proposed Transformer architecture [70] enables language models to have dozen of layers with huge capacity. These large language models, including BERT [13], GPT-2 [55], and XLNet [76], are trained on enormous large corpus and have shown impressive performance gains when transferred to other downstream NLP tasks in comparison to previous state of the art methods.

These pre-trained language models can also be used to extract sentence embeddings. There are many different ways to extract sentence-level features with pre-trained language models. In this work, we follow Sentence-BERT [58] which suggests that mean-pooling on the hidden representations yields best empirical performance.

2.2 Threat Model and Attack Taxonomy

In this section, we give an overview of threat models we consider in this paper and describe a taxonomy of attacks leaking information from embedding models. Figure 1 shows an overview of the attack taxonomy.

In many NLP applications, embedding vectors are often computed on sensitive user input. Although existing frameworks propose computing embeddings locally on user's device to protect the privacy of the raw data [6, 36, 50, 71], the embeddings themselves are shared with machine learning service providers or other parties for downstream tasks (training or inference). It is often tempting to assume that sharing embeddings might be "safer" than sharing the raw data, by their nature of being "simply a vector of real numbers." However, this ignores the information that is retained in, and may be extracted from embeddings. We investigate the following questions: *What kinds of sensitive information about the inputs are encoded in the embeddings? And can an adversary with access to the embeddings extract the encoded sensitive information?*

Threat model. Our threat model comprises the following entities. (1) $\mathcal{D}_{\text{train}}$ is a training dataset that may contain sensitive information. (2) Φ, the embedding model, which might be available in a white-box or black-box fashion. White-box access to the model reveals the model architecture and all parameters. Black-box access allows anyone to compute $\Phi(x)$ for x of their choice. (3) $\mathcal{E}_{\text{target}} = \{\Phi(x_i^*)\}$, a set of embedding vectors on sensitive inputs x_i^*. (4) $\mathcal{D}_{\text{aux}}$ is an auxiliary dataset available to the adversary comprising of either limited labeled data drawn from the same distribution as $\mathcal{D}_{\text{train}}$ or unlabeled raw text data. In the text domain, unlabeled data is cheap to collect due to the enormous amount free text available on the Internet while labeling them is often much more expensive.

An adversary, given access to some of the entities described above, aims to leak some sensitive information about the input x from the embedding vector $\Phi(x)$. By looking at variants of what information an adversary possesses and their target, we arrive at three broad classes of attacks against embeddings.

With sensitive input data such as personal messages, it is natural to consider an adversary whose goal is to (partially) recover this text from $\Phi(x)$. Even if the input data is not sensitive, they may be associated with sensitive attributes, and an adversary could be tasked with learning these attributes from $\Phi(x)$. And finally, rather than recovering inputs, the adversary given some information about

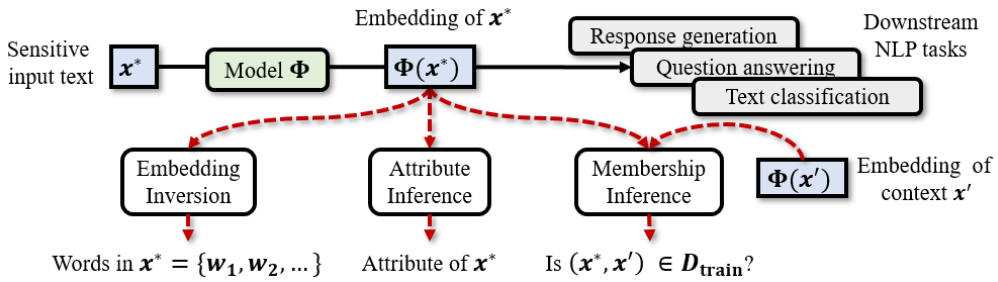

Figure 1: Taxonomy of attacks against embedding models. We assume adversary has access to the embedding $\Phi(x^*)$ of a sensitive input text x^* that will be used for downstream NLP tasks, and perform three information leakage attacks on $\Phi(x^*)$: (1) inverting the embedding back to the exact words in x^*, (2) inferring sensitive attribute of x^*, and (3) inferring the membership, i.e. whether x^* and its context x' has been used for training.

$\Phi(x)$ can aim to find if x was used to train Φ or not. These are formalized below.

Embedding inversion attacks. In this threat model, the adversary's goal is to invert a target embedding $\Phi(x^*)$ and recover words in x^*. We consider attacks that involve both black-box and white-box access to Φ. The adversary is also allowed access to an unlabeled $\mathcal{D}_{\text{aux}}$ and in both scenarios is able to evaluate $\Phi(x)$ for $x \in \mathcal{D}_{\text{aux}}$.

Sensitive attribute inference attacks. In this threat model, the adversary's goal is to infer sensitive attribute s^* of a secret input x^* from a target embedding $\Phi(x^*)$. We assume the adversary has access to Φ, and a set of labeled data of the form (x, s) for $x \in \mathcal{D}_{\text{aux}}$. We focus on discrete attributes $s^* \in \mathcal{S}$, where $\mathcal{S}$ is the set of all possible attribute classes, and an adversary performing the inference by learning a classifier f on $\mathcal{D}_{\text{aux}}$. Given sufficient labeled data , the adversary's task trivially reduces to plain supervised learning, which would rightly not be seen as adversarial. Therefore, the more interesting scenario, which is the focus of this paper, is when the adversary is given access to a very small set of labeled data (in the order of 10–50 per class) where transfer learning from $\Phi(x)$ is likely to outperform supervised learning directly from inputs x.

Membership inference attacks. Membership inference against ML models are well-studied attacks where the adversary has a target data point x^* and the goal is to figure out with good accuracy whether $x^* \in \mathcal{D}_{\text{train}}$ or not. Unlike previous attacks focused on supervised learning, some embedding models, such as word embeddings, allow you to trivially infer membership. For word embeddings, every member of a vocabulary set is necessarily part of $\mathcal{D}_{\text{train}}$. Instead, we expand the definition of training data membership to consider this data with their *contexts*, which is used in training. We assume the adversary has a target context of word $[w_1^*, \ldots, w_n^*]$ and access to $\boldsymbol{V}$ for word embedding, or a context of target sentences (x_a^*, x_b^*) and access to the model Φ for sentence embedding, and the goal is to decide the membership for the context. We also consider the target to be an aggregated level of data sentences $[x_1^*, \ldots, x_n^*]$ comprising multiple contexts for the adversary to determine if it were part of training Φ. Membership privacy for aggregation in the user level has also been explored in prior works [44, 67].

We further assume adversary has a limited $\mathcal{D}_{\text{aux}}$ labeled with membership. We propose that this is a reasonable and practical assumption as training data for text embeddings are often collected from the Internet where an adversary can easily inject data or get access to small amounts of labeled data through a more expensive labeling process. The assumption also holds for adversarial participants in collaborative training or federated learning [43, 44].

3 EMBEDDING INVERSION ATTACKS

The goal of inverting text embeddings is to recover the target input texts x^* from the embedding vectors $\Phi(x^*)$ and access to Φ. We focus on inverting embeddings of short texts and for practical reasons, it suffices to analyze the privacy of the inputs by considering attacks that recover a set of words without recovering the word ordering. We leave open the problem of recovering exact sequences and one promising direction involves language modeling [64].

A naïve approach for inversion would be enumerating all possible sequences from the vocabulary and find the recovery $\hat{x}$ such that $\Phi(\hat{x}) = \Phi(x^*)$. Although such brute-force search only requires black-box access to Φ, the search space grows exponentially with the length of x and thus inverting by enumerating is computationally infeasible.

The brute-force approach does not capture inherently what information might be leaked by Φ itself. This naturally raises the question of what attacks are possible if the adversary is given complete access to the parameters and architecture of Φ, i.e., white-box access. This also motivates a relaxation technique for optimization-based attacks in Section 3.1. We also consider the more constrained black-box access scenario in Section 3.2 where we develop learning based attacks that are much more efficient than exhaustive search by utilizing auxiliary data.

3.1 White-box Inversion

In a white-box scenario, we assume that the adversary has access to the embedding model Φ's parameters and architecture. We formulate white-box inversion as the following optimization problem:

$$\min_{\hat{x} \in \mathcal{X}(\mathcal{V})} ||\Phi(\hat{x}) - \Phi(x^*)||_2^2 \tag{3}$$

where $\mathcal{X}(\mathcal{V})$ is the set of all possible sequences enumerated from the vocabulary $\mathcal{V}$. The above optimization can be hard to solve

Algorithm 1 White-box inversion

Input: target embedding $\Phi(x^*)$, white-box embedding model Φ with lower layer representation function Ψ, temperature T, sparsity threshold τ_{sp}, auxiliary dataset $\mathcal{D}_{\text{aux}}$
Query function Φ and Ψ with $\mathcal{D}_{\text{aux}}$ and collect $\{(\Phi(x_i), \Psi(x_i))|x_i \in \mathcal{D}_{\text{aux}}\}$.
Train a linear mapping M by minimizing $||M(\Phi(x_i)) - \Psi(x_i)||_2^2$ on $\{(\Phi(x_i), \Psi(x_i))\}_i$.
if Ψ is mean-pooling on word embedding $\boldsymbol{V}$ of Φ **then**
    Initialize $\boldsymbol{z} \in \mathbb{R}^{|\mathcal{V}|}$
    while objective function of Eq 7 not converged **do**
        Update $\boldsymbol{z}$ with gradient of Eq 7.
        Project $\boldsymbol{z}$ to non-negative orthant.
    return $\hat{x} = \{w_i|\boldsymbol{z}_i \geq \tau_{\text{sp}}\}_{i=1}^{|\mathcal{V}|}$
else
    Initialize $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_\ell] \in \mathbb{R}^{\ell \times |\mathcal{V}|}$
    while objective function of Eq 6 not converged **do**
        Update $\boldsymbol{Z}$ with gradient of Eq 6.
    return $\hat{x} = \{w_i|i = \arg\max \boldsymbol{z}_j\}_{j=1}^{\ell}$

directly due to the discrete input space. Inspired by prior work on relaxing categorical variables [29], we propose a continuous relaxation of the sequential word input that allows more efficient optimization based on gradients.

The goal of the discrete optimization in Equation 3 is to select a sequence of words such that the distance between the output embeddings is minimized. We relax the word selection at each position of the sequence with a continuous variable $\boldsymbol{z}_i \in \mathbb{R}^{|\mathcal{V}|}$. As mentioned before, Φ first maps the input x of length ℓ into a sequence of word vectors $\boldsymbol{X} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_\ell]$ and then computes the text embedding based on $\boldsymbol{X}$. For optimizing $\boldsymbol{z}_i$, we represent the selected word vectors $\hat{\boldsymbol{v}}_i$ using a softmax attention mechanism:

$$\hat{\boldsymbol{v}}_i = \boldsymbol{V}^\top \cdot \text{softmax}(\boldsymbol{z}_i/T) \quad \text{for } i = 1, \ldots, \ell \tag{4}$$

where $\boldsymbol{V}$ is the word embedding matrix in Φ and T is a temperature parameter. The softmax function approximates hard argmax selection for $T < 1$. Intuitively, $\text{softmax}(\boldsymbol{z}_i/T)$ models the probabilities of selecting each word in the vocabulary at position i in the sequence and $\hat{\boldsymbol{v}}^i$ is the average of all word vectors weighted by the probabilities.

Let $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_\ell] \in \mathbb{R}^{\ell \times |\mathcal{V}|}$ and $\text{relaxed}(\boldsymbol{Z}, T) = [\hat{\boldsymbol{v}}_1, \ldots, \hat{\boldsymbol{v}}_\ell]$ be the sequence of softmax relaxed word vectors. For simplicity, we denote $\Phi(\boldsymbol{X})$ as the text embedding computed from any sequence of word vectors $\boldsymbol{X}$. Then our relaxed optimization problem is:

$$\min_{\boldsymbol{Z}} ||\Phi(\text{relaxed}(\boldsymbol{Z}, T)) - \Phi(x^*)||_2^2. \tag{5}$$

With continuous relaxation and white-box access to Φ, the optimization problem can now be solved by gradient-based methods as gradients of $\boldsymbol{Z}$ can be calculated using back-propagation. To recover, we compute $\hat{x} = \{w_i|i = \arg\max \boldsymbol{z}_j\}_{j=1}^{\ell}$.

Inverting embedding from deep models. Prior works [15, 16, 41, 69] demonstrated that inverting image features from the higher layers of deep model can be challenging as representations from the higher layers are more abstract and generic while inverting from lower-layer representations results in better reconstruction of the image. With the recent advance in Transformer models [70], text embeddings can also be computed from deep models with many layers such as BERT [13]. Directly inverting such deep embeddings using the relaxed optimization method in Equation 5 can results in inaccurate recovery as the optimization becomes highly non-convex and different sequences with similar semantics can be mapped to the same location in the high-level embedding space. In reality, however, it is more common to use the embeddings from the higher layers than from lower layers for downstream tasks and thus an adversary might not be able to observe embeddings from lower layers at all.

To resolve this issue, we propose to invert in two stages: (1) the adversary first maps the observed higher layer embedding $\Phi(x)$ to a lower layer one with a learned mapping function M, and (2) the adversary then solves the optimization problem of minimizing $||\Psi(\hat{x}) - M(\Phi(x^*))||_2^2$ where Ψ denotes the lower layer embedding function. To learn the mapping function M, adversary queries the white-box embedding model to get $(\Phi(x_i), \Psi(x_i))$ for each auxiliary data $x_i \in \mathcal{D}_{\text{aux}}$ and trains M with the set $\{(\Phi(x_i), \Psi(x_i))\}_i$. After M is trained, adversary solve the relaxed optimization:

$$\min_{\boldsymbol{Z}} ||\Psi(\text{relaxed}(\boldsymbol{Z}, T)) - M(\Phi(x^*))||_2^2 \tag{6}$$

In practice, we find that learning a linear least square model as M works reasonably well.

A special case of inverting lowest representation. The lowest embedding we can compute from the text embedding models would be the average of the word vectors, i.e., $\Psi(x) = (1/\ell) \cdot \sum_{i=1}^{\ell} \boldsymbol{v}_i$. Inverting from such embedding reduces to the problem of recovering the exact words vectors with given averaged vector. Instead of using the relaxed optimization approach in Equation 5, we use a sparse coding [49] formulation as following:

$$\min_{\boldsymbol{z} \in \mathbb{R}_{\geq 0}^{|\mathcal{V}|}} ||\boldsymbol{V}^\top \cdot \boldsymbol{z} - M(\Phi(x^*))||_2^2 + \lambda_{\text{sp}}||\boldsymbol{z}||_1 \tag{7}$$

where $\boldsymbol{V}$ is the word embedding matrix and the variable $\boldsymbol{z}$ quantifies the contribution of each word vector to the average. We constrain $\boldsymbol{z}$ to be non-negative as a word contributes either something positive if it is in the sequence x or zero if it is not in x. We further penalize $\boldsymbol{z}$ with L^1 norm to ensure its sparsity as only few words from the vocabulary contributed to the average. The above optimization can be solved efficiently with projected gradient descent, where we set the coordinate $\boldsymbol{z}_j$ to 0 if $\boldsymbol{z}_j < 0$ after each descent step. The final recovered words are those with coefficient $\boldsymbol{z}_j > \tau_{\text{sp}}$ for a sparsity threshold hyper-parameter τ_{sp}.

3.2 Black-box Inversion

In the black-box scenario, we assume that the adversary only has query access to the embedding model Φ, i.e., adversary observes the output embedding $\Phi(x)$ for a query x. Gradient-based optimization is not applicable as the adversary does not have access to the model parameters and thus the gradients.

Instead of searching for the most likely input, we directly extract the input information retained in the target embedding by formulating a learning problem. The adversary learns an inversion model Υ that takes a text embedding $\Phi(x)$ as input and outputs the

set of words in the sequence x. As mentioned before, our goal is to recover the set of words in the input independent of their word ordering. We denote $\mathcal{W}(x)$ as the set of words in the sequence x. The adversary utilizes the auxiliary dataset $\mathcal{D}_{\text{aux}}$ and queries the black-box Φ and obtain a collection of $(\Phi(x), \mathcal{W}(x))$ for each $x \in \mathcal{D}_{\text{aux}}$. The adversary then trains the inversion model Υ to maximize $\log P_\Upsilon(\mathcal{W}(x)|\Phi(x))$ on the collected set of $(\Phi(x), \mathcal{W}(x))$ values. Once Υ is trained, the adversary predicts the words in the target sequence x^* as $\Upsilon(\Phi(x^*))$ for an observed $\Phi(x^*)$.

Multi-label classification. The goal is to predict the set of words in sequence x given the embedding $\Phi(x)$. A common choice for such a goal is to build a multi-label classification (MLC) model, where the model assigns a binary label of whether a word is in the set for each word in the vocabulary. The training objective function is then:

$$\mathcal{L}_{\text{MLC}} = -\sum_{w \in \mathcal{V}} [y_w \log(\hat{y}_w) + (1 - y_w) \log(1 - \hat{y}_w)] \tag{8}$$

where $\hat{y}_w = P_\Upsilon(y_w|\Phi(x))$ is the predicted probability of word w given Υ conditioned on $\Phi(x)$ and $y_w = 1$ if word w is in x and 0 otherwise.

Multi-set prediction. One drawback in the above multi-label classification model is that the model predicts the appearance of each word independently. A more sophisticated formulation would be predicting the next word given the current predicted set of words until all words in the set are predicted. We adopt the multi-set prediction loss [72] (MSP) that is suited for the formulation. Our MSP model trains a recurrent neural network that predicts the next word in the set conditioned on the embedding $\Phi(x)$ and current predicted set of words. The training objective is as follows:

$$\mathcal{L}_{\text{MSP}} = \sum_{i=1}^{\ell} \frac{1}{|\mathcal{W}_i|} \sum_{w \in \mathcal{W}_i} -\log P_\Upsilon(w|\mathcal{W}_{<i}, \Phi(x)) \tag{9}$$

where $\mathcal{W}_i$ is the set of words left to predict at timestamp i and $\mathcal{W}_{<i}$ is the set of the predicted words before i. The MSP formulation allows Υ to learn a policy on the order of the words should be predicted instead of predicting all words independently and simultaneously. In Section 6.2, we empirically show that MSP outperforms MLC in terms of the precision-recall trade-off.

4 ATTRIBUTE INFERENCE ATTACKS

Embeddings are designed to encode rich semantic information about the input data. When the input data is user-related, e.g., a user's video watch history, the embedding vector naturally captures information about the user and is often much more informative than the raw input. Although in many applications such rich information from the embeddings is desired in order to provide personalized services to the user, the embeddings may potentially reveal sensitive information about the input that might not directly appear in or be easy to infer from the input. As a motivating example, consider an adversary that may curate a small set of public comments labeled with authors of interest and then planning to use semantically rich embeddings on unlabeled, targeted text fragments to aim to deanonymize the author of the text. This is also the typical setup in stylometry research [59, 65] that we make more realistic by considering the challenges of curating labeled data (which may be a costly process) and strictly limiting the number of labeled examples the adversary has to work with.

Algorithm 2 Black-box Inversion with multi-set prediction

Input: target embedding $\Phi(x^*)$, black-box model Φ, auxiliary data $\mathcal{D}_{\text{aux}}$
procedure MSPLoss(x, Φ, Υ)
  Initialize $\mathcal{L} \leftarrow 0, \mathcal{W}_i \leftarrow \mathcal{W}(x), \mathcal{W}_{<i} \leftarrow \emptyset$
  for $i = 1$ to ℓ **do**
    Predict a word $\hat{w} = \arg\max P_\Upsilon(w|\mathcal{W}_{<i}, \Phi(x))$.
    $\mathcal{W}_i \leftarrow \mathcal{W}_i / \{\hat{w}\}$ and $\mathcal{W}_{<i} \leftarrow \mathcal{W}_{<i} \cup \{\hat{w}\}$.
    $\mathcal{L} \leftarrow \mathcal{L} - \frac{1}{|\mathcal{W}_i|} \sum_{w \in \mathcal{W}_i} \log P_\Upsilon(w|\mathcal{W}_{<i}, \Phi(x))$.
  return $\mathcal{L}$
Initialize Υ as a recurrent neural network.
while Υ not converged **do**
  Sample a batch $\mathcal{B} \subset \mathcal{D}_{\text{aux}}$.
  Compute $\mathcal{L}_{\text{MSP}} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{x_i \in \mathcal{B}} \text{MSPLoss}(x_i, \Phi, \Upsilon)$.
  Update Υ with $\nabla \mathcal{L}_{\text{MSP}}$.
return $\hat{x} = \{\arg\max P_\Upsilon(w|\mathcal{W} < i, \Phi(x^*))\}_{i=1}^{\ell}$

Algorithm 3 Sensitive attribute inference

Input: target embedding $\Phi(x^*)$, black-box model Φ, labeled auxiliary data $\mathcal{D}_{\text{aux}}$
Query Φ with $\mathcal{D}_{\text{aux}}$ and collect $\{(\Phi(x_i), s_i)|x_i \in \mathcal{D}_{\text{aux}}\}$.
Train a classifier f that predicts s on $\{(\Phi(x_i), s_i)\}$.
return $\hat{s} = f(\Phi(x^*))$

For inferring sensitive attributes, we assume that the adversary has a limited set $\mathcal{D}_{\text{aux}} = \{(\Phi(x_i), s_i)\}_i$ of embeddings labeled with the sensitive attribute, such as text authorship. The adversary then treats the inference problem as a downstream task and trains a classifier which predicts s given $\Phi(x)$ as inputs on $\mathcal{D}_{\text{aux}}$. At inference time, adversary simply applies the classifier on observed embedding $\Phi(x^*)$ to infer the sensitive attribute of x^*. We focus on the scenario of the adversary only having limited labeled data so as to (a) closely match real scenarios where labeled sensitive data would be challenging to collect, and (b) demonstrate how easily sensitive information can be extracted from the embeddings which therefore constitutes an important vector of information leakage from embeddings.

Connections between leakage and the objective. In *supervised* learning, deep representations can reveal sensitive attributes of the input as these attributes might be used as internal features for the learning task [68]. The connection between *unsupervised* learning tasks and the leakage of sensitive attributes is less well understood. The objective functions (Equation 1 and 2) that maximize the semantic similarity of data in context for training unsupervised dual-encoder embedding models fall into *contrastive learning* framework [63]. This framework theoretically explains how unsupervised embedding training helps with the downstream tasks with a utility perspective. We explore how it might also favor the inference of some sensitive attributes from the perspective ofprivacy.

In this framework, training data of the embedding models are associated with latent classes (e.g., authors of the texts). When

Algorithm 4 MIA on word embeddings

Input: target window of words $C = [w_b, \dots, w_0, \dots, w_e]$, word embedding matrix $\boldsymbol{V}$, similarity function δ
Map words in C with $\boldsymbol{V}$ and get $[\boldsymbol{v}_{w_b}, \dots, \boldsymbol{v}_{w_0}, \dots, \boldsymbol{v}_{w_e}]$.
$\Delta \leftarrow \{\delta_i | \delta_i = \delta(\boldsymbol{v}_{w_0}, \boldsymbol{v}_{w_i}), \forall w_i \in C/\{w_0\}\}$.
return "member" if $\frac{1}{|\Delta|}\sum_{\delta_i \in \Delta} \delta_i \geq \tau_m$ else "non-member"

training with contrastive loss, the embeddings are learned so as to be similar for data in the same context and to be dissimilar for data coming from negative samples. Our approach takes advantage of the fact that data sharing the same latent class will often appear in the same context. Therefore, embedding similarity will be closer for inputs from the same class, and consequently with the same sensitive attribute when there is a correlation. We further note that unsupervised embeddings are especially helpful for attribute inference under the limited data constraints as the embeddings are trained on much larger unlabeled data which allows them to learn semantic similarity over latent classes that might not be captured only given limited labeled data.

5 MEMBERSHIP INFERENCE ATTACKS

Both inverting embeddings and inferring sensitive attributes concern inference-time input privacy, i.e., information leaked about the input x from the embedding vector. Another important aspect of privacy of ML models is *training data privacy*, namely, what information about the training data (which might be potentially sensitive) is leaked by a model during the training process? We focus on membership inference attacks [66] as a measurement of training data leakage in the embedding models.

The goal of membership inference is to infer whether a data point is in the training set of a given machine learning model. Classic membership inference attacks mainly target supervised machine learning, where a data point consists of a input feature vector and a class label. For unsupervised embedding models trained on units of data in context, we thus wish to infer the membership of a context of data (e.g., a sliding window of words or a pair of sentences).

5.1 Word Embeddings

Prior works [60, 77] on membership inference suggest that simple thresholding attacks based on loss values can be theoretically optimal under certain assumptions and practically competitive to more sophisticated attacks [66]. In embedding models, the loss is approximated based on sampling during training as described in Section 2 and computing exact loss is inefficient. We thus develop simple and efficient thresholding attacks based on similarity scores instead of loss values.

Word embeddings are trained on a sliding window of words in the training corpus. To decide the membership for a window of words $C = [w_b, \dots, w_0, \dots, w_e]$, the adversary first converts each word into its embedding vectors $[\boldsymbol{v}_{w_b}, \dots, \boldsymbol{v}_{w_0}, \dots, \boldsymbol{v}_{w_e}]$. Then the adversary computes a set of similarity scores $\Delta = \{\delta(\boldsymbol{v}_{w_0}, \boldsymbol{v}_{w_i}) | \forall w_i \in C/\{w_0\}\}$, where δ is a vector similarity measure function (e.g. cosine similarity). Finally, the adversary uses the averaged score in Δ to decide membership: if the averaged score is above some threshold then C is a member of the training data and not a member otherwise.

Algorithm 5 Aggregated-level MIA on sentence embeddings

Input: target sentences in context $\mathcal{X} = [x_1, \dots, x_n]$, sentence embedding model Φ, similarity function δ, auxiliary data $\mathcal{D}_{\text{aux}}$ with membership labels
Map sentences in $\mathcal{X}$ with Φ and get $[\Phi(x_1), \dots, \Phi(x_n)]$.
if learning similarity function **then**
    Initialize similarity function δ' with projection W_m.
    while δ' not converged **do**
        Sample a batch $\mathcal{B} \subset \mathcal{D}_{\text{aux}}$.
        Compute loss $\mathcal{L}_{\text{MIA}}$ for $(x_a, x_b) \in \mathcal{B}$ with Eq 10.
        Update W_m with $\nabla \mathcal{L}_{\text{MIA}}$.
    Replace similarity function $\delta \leftarrow \delta'$.
$\Delta \leftarrow \{\delta_i | \delta_i = \delta(\Phi(x_i), \Phi(x_{i+1}))\}_{i=1}^{n-1}$.
return "member" if $\frac{1}{|\Delta|}\sum_{\delta_i \in \Delta} \delta_i \geq \tau_m$ else "non-member"

5.2 Sentence Embeddings

In sentence embeddings, we wish to decide membership of a pair of sentence in context (x_a, x_b). We simply use the similarity score $\delta(\Phi(x_a), \Phi(x_b))$ as the decision score for membership inference as sentences in context used for training will be more similar to each other than sentences which were not used for training.

Aggregate-level membership inference. Sometimes, deciding membership of a pair of sentences might not be enough to cause a real privacy threat. In many user-centric applications, models are trained on aggregation of data from users; e.g., a keyboard prediction model is trained on users' input logs on their phone [44]. In this scenario, the adversary infers membership on aggregate data from a particular user to learn whether this user participated training or not.

To perform MIA on aggregate text of n sentences $\mathcal{X} = [x_1, x_2, \dots, x_n]$, the adversary first gets each sentence embeddings $[\Phi(x_1), \Phi(x_2), \dots, \Phi(x_n)]$. Then the adversary collects the set of similarity score $\Delta = \{\delta(x_i, x_{i+1})\}_{i=1}^{n-1}$. Finally, similar to MIA against word embeddings, we use the average score in Δ as the decision score for membership inference.

Learned similarity metric function. Using a pre-defined similarity measure may not achieve best membership inference results. With auxiliary data labeled with membership information, an adversary can learn a similarity metric customized for inference membership. More specifically, they learn a projection matrix W_m and computes the learned similarity as $\delta'(x_a, x_b) = \delta(W_m^\top \cdot \Phi(x_a), W_m^\top \cdot \Phi(x_b))$. The attack optimizes the binary cross-entropy loss with membership labels as following:

$$\mathcal{L}_{\text{MIA}} = -[y_m \log(\delta'_{a,b}) + (1 - y_m) \log(1 - \delta'_{a,b})] \qquad (10)$$

where $\delta'_{a,b} = \delta'(x_a, x_b)$ and $y_m = 1$ if (x_a, x_b) where in the training set and 0 otherwise.

6 EXPERIMENTAL EVALUATION

6.1 Embedding Models and Datasets

As each of the attacks assess different perspectives of privacy, we evaluate them on different text embedding models that are either trained locally on consumer hardware or are trained elsewhere and public available. Here, we describe text embedding models (and their corresponding datasets) we trained locally and evaluated against attacks described in previous sections. Other public models and datasets are detailed in subsequent subsections.

Word Embeddings on Wikipedia. We collected nearly 150,000 Wikipedia articles [42] for evaluating word embeddings. We locally trained Word2Vec [46], FastText [3], and GloVe [54] embedding models using half of the articles and use the other half for evaluating membership inference.

For all word embeddings, we set the number of dimension in embedding vector d to be 100. For Word2Vec and FastText, we set the number of sampled negative words $|\mathcal{V}_{\text{neg}}|$ to be 25, learning rate to be 0.05, sliding window size to be 5 and number of training epochs to be 5. For GloVe, we set the number of training iterations to be 50 as suggested in the original paper [54].

Sentence Embeddings on BookCorpus. Following prior works on training unsupervised sentence embeddings [33, 39], we collected sentences from BookCorpus [79] consists of 14,000 books. We sample 40 millions sentences from half of the books as training data and use the rest as held-out data.

We locally train sentence embedding models with dual-encoder architecture described in Section 2. We considered two different neural network architecture for the embedding models: a recurrent neural network (LSTM [27]) and a three-layer Transformer [70]. For the LSTM, we set the size of embedding dimension d to be 1,200, number of training epochs to be 1 and learning rate to be 0.0005 following previous implementation [39]. For the Transformer, we set d to be 600, number of training epochs to be 5 with a warm-up scheduled learning rate following [70]. For both architectures, we train the model with Adam optimizer [32] and set the negative samples $\mathcal{X}_{\text{neg}}$ as the other sentences in the same training batch at each step and $|\mathcal{X}_{\text{neg}}| = 800$.

6.2 Embedding Inversion

Target and auxiliary data. We randomly sample 100,000 sentences from 800 authors in the held-out BookCorpus data as the target data $\mathcal{D}_{\text{target}} = \{x_i^*\}$ to be recovered and perform inversion on the set of embeddings $\mathcal{E}_{\text{target}} = \{\Phi(x_i^*)\}$. We consider two types of auxiliary data $\mathcal{D}_{\text{aux}}$: same-domain and cross-domain data. For same-domain $\mathcal{D}_{\text{aux}}$, we use a set of 200,000 randomly sampled sentences from BookCorpus that is disjoint to $\mathcal{D}_{\text{target}}$. For cross-domain $\mathcal{D}_{\text{aux}}$, we use a set of 800,000 randomly sampled sentences from Wikipedia articles. We set the number of cross-domain data points to be more than the same-domain data to match real-world constraints where cross-domain data is typically public and cheap to collect.

Additional embedding models. In addition to the two dual-encoder embedding models with LSTM and Transformer, we further experiment with popular pre-trained language models for sentence embedding. We consider the original BERT [13] and the state-of-the-art ALBERT [35]. We use mean pooling of the hidden token representations as the sentence embedding as described in Section 2.1.

Table 1: White-box inversion results on sentence embeddings. Pre denotes precision and Rec denotes recall. We leave the cross-domain results for Equation 5 as blank as no learning on auxiliary data is needed. The best results are in bold.

	Same domain			Cross domain		
Equation 5	Pre	Rec	F1	Pre	Rec	F1
LSTM	56.93	56.54	56.74	-	-	-
Transformer	35.74	35.44	35.59	-	-	-
BERT	0.84	0.89	0.87	-	-	-
ALBERT	3.36	2.95	3.14	-	-	-
Equation 7	Pre	Rec	F1	Pre	Rec	F1
LSTM	63.68	56.69	**59.98**	57.98	48.05	52.55
Transformer	65.32	60.39	**62.76**	59.97	54.45	57.08
BERT	50.28	49.17	**49.72**	46.44	43.73	45.05
ALBERT	70.91	55.49	**62.26**	68.45	53.18	59.86

Evaluation metrics. As the goal of inversion is to recover the set of words in the sensitive inputs, we evaluate our inversion methods based on precision (the percentage of recovered words in the target inputs), recall (the percentage of words in the target inputs are predicted) and F1 score which is the harmonic mean between precision and recall.

White-box inversion setup. We evaluate the white-box inversion with Equation 5 and Equation 7. For inversion with Equation 5, we set the temperature T to be 0.05. For inversion with Equation 7, we set the L^1 penalty coefficient λ_{sp} to be 0.1 and the sparsity threshold τ_{sp} to be 0.01. For both methods, we use Adam optimizer [32] for gradient descent with learning rate set to 0.001. The hyperparameters are tuned on a subset of the adversary's auxiliary data.

White-box inversion results. Table 1 summarizes the results. Note that there is no cross domain results for Equation 5 since no learning is needed. For inversion with Equation 5, an adversary can extract more than a half and a third of the target input from LSTM and Transformer embedding models respectively as indicate by the F1 score. This method performs poorly on BERT and ALBERT models. One plausible reason is that with many more layers in BERT where higher layer embeddings are more abstract than lower layers, directly optimizing for the highest layer could lead to recovering synonyms or semantically-related words rather than the targets.

For inversions with Equation 7, all performance scores increase from Equation 5 on all models. We can recover more than half of the input texts on nearly all models. We also notice that there is only a little loss in performance when using cross-domain data for training the mapping M.

We further investigate the performance of inverting embeddings from different layers in BERT and ALBERT as shown in Figure 2. There are in total 12 Transformer layers in BERT and ALBERT models and we choose embeddings from 2, 4, 6, 8, 10, 12 layer for inversion. We also compare with inverting from layer 0, i.e. mean-pooling

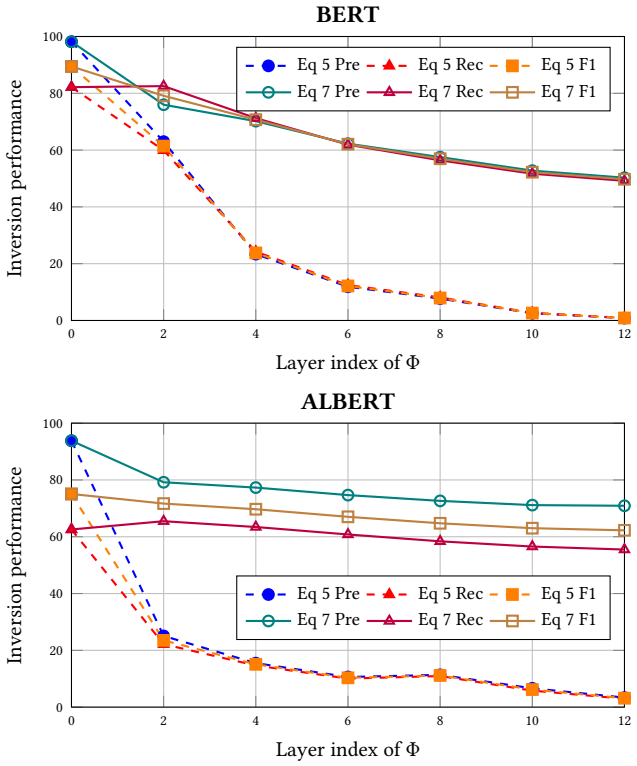

Figure 2: Performance of embedding inversion on sentence embedding from different layers of BERT and ALBERT. Pre denotes precision and Rec denotes recall. The x-axis is the layer index denoting which layer the embeddings are computed. Index 0 is the lowest (bottom) layer and 12 is the highest (top) layer.

Table 2: Black-box inversion results on sentence embeddings. Pre denotes precision and Rec denotes recall. The best results are in bold.

	Same domain			Cross domain		
$\mathcal{L}_{\text{MLC}}$	Pre	Rec	F1	Pre	Rec	F1
LSTM	90.53	39.35	54.86	87.71	32.91	47.86
Transformer	81.18	26.07	39.47	77.34	21.82	34.04
BERT	89.70	36.80	52.19	84.05	30.28	44.52
ALBERT	95.92	48.71	64.61	92.51	44.30	59.91
$\mathcal{L}_{\text{MSP}}$	Pre	Rec	F1	Pre	Rec	F1
LSTM	61.69	64.40	**63.02**	59.52	62.20	60.83
Transformer	53.59	55.72	**54.63**	51.37	52.78	52.07
BERT	60.21	59.31	**59.76**	55.18	55.44	55.31
ALBERT	76.77	72.05	**74.33**	74.07	70.66	72.32

of word embedding in BERT models. The performance drops drastically when the layer goes high when inverting with Equation 5. When training a mapping M and inverting with Equation 7, the drop in the performance is much less significant for higher layers.

Black-box setup. We evaluate the black-box inversion with $\mathcal{L}_{\text{MLC}}$ (Equation 8) and $\mathcal{L}_{\text{MSP}}$ (Equation 9). For $\mathcal{L}_{\text{MLC}}$, we train $|\mathcal{V}|$ binary classifiers as Υ for each $w \in \mathcal{V}$. For $\mathcal{L}_{\text{MSP}}$, we train a one-layer LSTM as Υ with number of hidden units set to 300. We train both models for 30 epochs with Adam optimizer and set learning rate to 0.001, batch size to 256.

Black-box results. Table 2 summarizes the results. Inversion with $\mathcal{L}_{\text{MLC}}$ can achieve high precision with low recall. This might be due to the inversion model Υ being biased towards the auxiliary data and thus confident in predicting some words while not others. Inversion with $\mathcal{L}_{\text{MSP}}$ yields better balance precision and recall and thus higher F1 scores.

6.3 Sensitive Attribute Inference

Target and auxiliary data. We consider authorship of sentence to be the sensitive attribute and target data to be a collection of sentences of randomly sampled author set $\mathcal{S}$ from the held-out dataset of BookCorpus, with 250 sentences per author. The goal is to classify authorship s of sentences amongst $\mathcal{S}$ given sentence embeddings. For auxiliary data, we consider 10, 20, 30, 40 and 50 labeled sentences (disjoint from those in the target dataset) per author. We also vary the size of author set $|\mathcal{S}| = 100, 200, 400$ and 800 where the inference task becomes harder as $|\mathcal{S}|$ increases.

Baseline model. To demonstrate sensitive attribute leakage from the embedding vector, we compare the attack performance between embedding models and a baseline model that is trained from raw sentences without access to the embeddings. We train a TextCNN model [31] as the baseline, which is efficient to train and has been shown to achieve accurate authorship attribution in previous works [59, 65].

Additional embedding models. As discussed in Section 4, the embedding models trained with dual-encoder and contrastive learning that is a focus of this paper might, in particular, favor attribute inference. We compare the dual-encoder embedding models with two other embedding models trained with different objective functions. The first is the Skip-thought embedding model [33] which is trained to generate the context given a sentence. We also evaluate on InferSent embeddings [10] that is trained with supervised natural language inference tasks.

Setup. For the baseline TextCNN model, we set the number of filters in convolutional layer to 128. For all other embedding models, we train a linear classifier for authorship inference. We train all inference models for 30 epochs with Adam optimizer and set learning rate to 0.001 and batch size to 128. We repeat each experiment 5 times with sampled $\mathcal{S}$ using different random seed and report the averaged top-5 accuracy.

Results. Figure 3 demonstrates the results of authorship inference with different number of labeled data and different number of authors. The baseline TextCNN models trained from scratch with limited labeled data have the worst performance across all settings. Skip-thought and InferSent embeddings can outperform TextCNN but the gap between the performance decreases as the number of labeled examples increase. The LSTM and Transformer dual-encoder models achieve best inference results and are better than the baseline by a significant margin in all scenarios. This demonstrates that

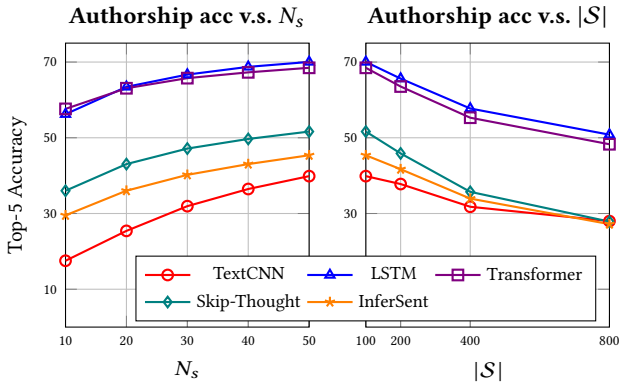

Figure 3: Performance of sensitive attribute (author) inference with different models. For the left figure, the x-axis is the number of labeled data per author N_s and the y-axis is the top-5 accuracy of classifying 100 authors. For the right figure, the x-axis is the number of author classes $|\mathcal{S}|$ and the y-axis is the top-5 accuracy with 50 labeled data per author.

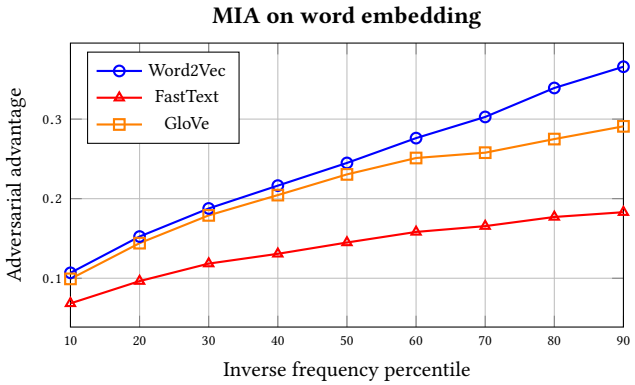

Figure 4: Performance of membership inference attack on Word2Vec, FastText and GloVe. The x-axis is the inverse frequency percentile range (the smaller the more frequent) and the y-axis is the adversarial advantage.

embeddings from dual-encoder models trained with contrastive learning framework [63] aid attribute inference attacks the most comparing to other pre-trained embeddings.

6.4 Membership Inference

Evaluation metrics. We consider membership inference as a binary classification task of distinguishing members and non-members of training data. We evaluate the performance of membership inference attacks with adversarial advantage [77], defined as the difference between the true and false positive rate. Random guesses offer an advantage of 0.

Word embedding setup. We evaluate membership inference attacks on sliding window of 5 words from Wikipedia articles. We also perform the attack separately for windows with central words having different frequencies, following the intuition that rare words are prone to more memorization [67]. Specifically, we evaluate the attack for windows with frequency of central words in decile (10th, … , 90th percentile) ranges. We use cosine similarity for π.

Word embedding results. Figure 4 demonstrates the results. For frequent (10th percentile) central words, there is almost no memorization (advantage < 0.1). As this frequency decreases, the advantage increases correspondingly to roughly 0.3. FastText is most resistant to these attacks possibly due to its operating on sub-word units rather than exact words.

Sentence embedding setup. We evaluate membership inference attacks on context- and aggregate-level data from the BookCorpus dataset. We consider a pair of sentences for context-level data, and a collection of sentences from the same book for aggregate-level data with a goal of inferring if the book is part of the training corpus. As with word embeddings, we evaluate the attack on different frequencies (averaged across words in a sentence). We use dot-product similarity for π, and for learning-based similarity, we learn a projection matrix with 10% of training and hold-out data.

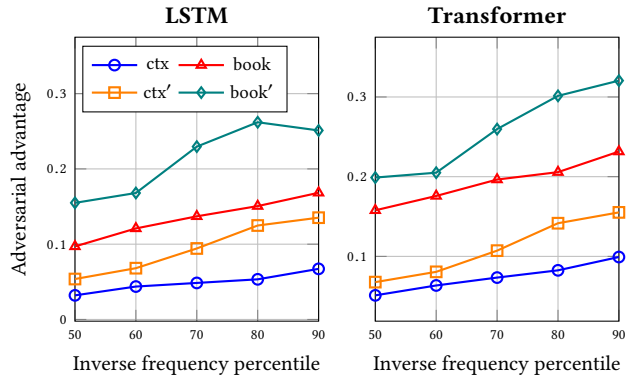

Figure 5: Performance of context-level (ctx) and book-level membership inference attack on sentence embedding trained with LSTM and Transformer. The x-axis is the inverse frequency percentile range (the smaller the more frequent) and the y-axis is the adversarial advantage. The ctx′ and book′ denote results with learned similarity

We optimize $\mathcal{L}_{\text{MIA}}$ with the Adam optimizer for 10 epochs with learning rate set to 0.001.

Sentence embedding results. Figure 5 shows the results of MIA on embeddings from LSTM and Transformer dual-encoder models. For both models, context-level MIA advantage scores are below 0.1 for all frequency ranges, indicating that adversarial does not gain much information about context-level membership from the embeddings. Learning based similarity can improve context-level MIA slightly. For aggregated book-level inference, adversaries achieve a greater advantage than context-level inference and learning-based similarity scores can boost the advantage to 0.3 for books with infrequent sentences.

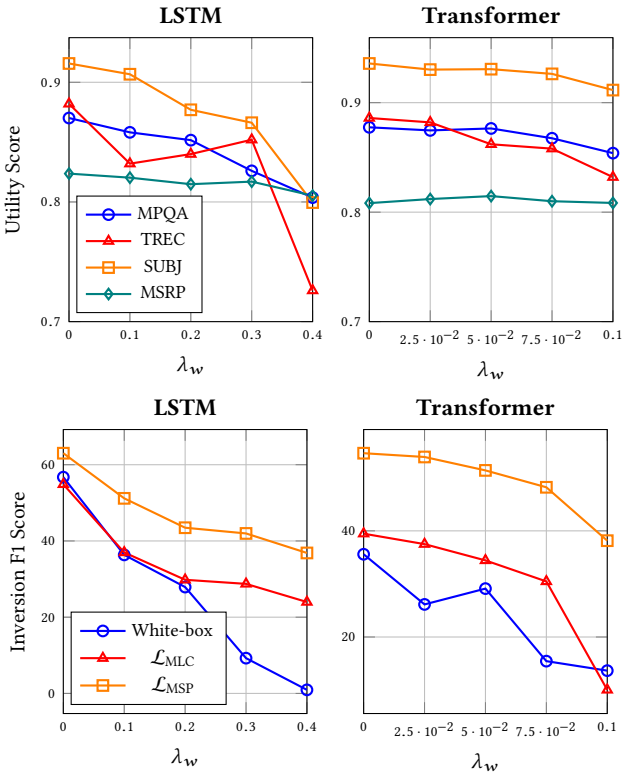

Figure 6: Effects of adversarial training against embedding inversion on the utility (top row) and the inversion F1 score (bottom row) for sentence embeddings trained with LSTM and Transformer.

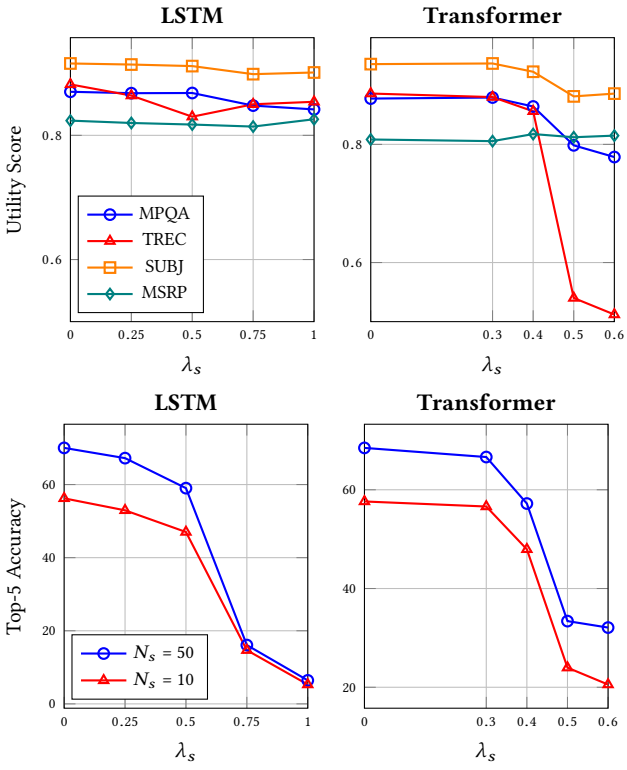

Figure 7: Effects of adversarial training against sensitive attribute inference on the utility (top row) and the author classification top-5 accuracy (bottom row) for sentence embeddings trained with LSTM and Transformer. N_s denotes number of labeled data per author.

7 DEFENSES

Adversarial training. Attacks involving embedding inversions and sensitive attributes are both inference-time attacks that wish to infer information about the sensitive inputs given the output of the embedding. A common defence mechanism for such inference-time attacks is adversarial training [9, 17, 18, 38, 74]. In this framework, a simulated adversary $\mathcal{A}$ is trained to infer any sensitive information jointly with a main model Φ while Φ is trained to maximize the adversary's loss and minimize the primary learning objective. The embeddings trained with this minimax optimization protects sensitive information from an inference-time adversary to an extent while maintaining their utility for downstream tasks.

To defend against embedding inversion attacks, $\mathcal{A}$ is trained to predict the words in x given $\Phi(x)$. For a pair of sentences in context (x_a, x_b) and a set of negative example $\mathcal{X}_{\text{neg}}$, the training objective for Φ is:

$$\min_{\Phi} \max_{\mathcal{A}} \lambda_w \log P_{\mathcal{A}}(\mathcal{W}(x_b)|\Phi(x_b)) - \log P_{\Phi}(x_b|x_a, \mathcal{X}_{\text{neg}}),$$

where $\mathcal{W}(x)$ is the set of words in x and the coefficient λ_w controls the balance between the two terms. A natural choice for $\log P_{\mathcal{A}}$ is to use the multi-label classification loss $\mathcal{L}_{\text{MLC}}$ in Equation 8. To defend against sensitive attribute attacks, $\mathcal{A}$ is trained to predict the sensitive attribute s in x from the embedding $\Phi(x)$. As above, the training objective for Φ is:

$$\min_{\Phi} \max_{\mathcal{A}} \lambda_s \log P_{\mathcal{A}}(s|\Phi(x_b)) - \log P_{\Phi}(x_b|x_a, \mathcal{X}_{\text{neg}}),$$

where the coefficient λ_s controls the balance between the two terms. Both minimax training objectives can be efficiently optimized with the gradient reversal trick [20].

Results with adversarial training. We evaluate this adversarial training approach on dual-encoder models with LSTM and Transformer. We keep all the training hyper-parameters the same as in Section 6.1. We train multiple models under different λ_w and λ_s and evaluate their effects on attack performance as well as utility scores for downstream tasks. For utility measurement, we evaluate the adversarially trained embeddings on four sentence analysis benchmarks: multi-perspective question answering (MPQA) [73], text retrieval (TREC) [37], subjectivity analysis (SUBJ) [52] and Microsoft Research paraphrase corpus (MSRP) [14]. We treat all benchmarks as classification problem and train a logistic regression model for each task following previous works [33, 39].

Figure 6 shows the results for adversarial training against inversion. As λ_w increases, the adversary performance drops for all models. White-box inversion attacks drop most significantly. LSTM embedding models needs a larger λ_w than Transformer models to achieve similar mitigation, possible due to the fact that Transformer

models have more capacity and are more capable of learning two tasks. Utility scores on the benchmarks drop more drastically on embedding models using LSTM than Transformer due to the larger value of λ_w.

Figure 7 shows the results for adversarial training against authorship inference. As λ_s increases, the adversary performance on inferring authorship drops significantly for both inference attack models trained with 10 and 50 labeled data per author. Nearly all utility scores on the four benchmarks remain rather stable for different λ_s's. This also demonstrates that different adversary tasks can have different impact on the utility of embeddings. In our case, removing input word information from embeddings is a harder task than removing authorship and thus will have a larger impact on the utility.

8 RELATED WORK

Privacy in deep representations. Prior works demonstrate that representations from supervised learning models leak sensitive attributes about input data that are statistically uncorrelated with the learning task [68], and gradient updates in collaborative training (which depend on hidden representations of training inputs) also leak sensitive attributes [45]. In contrast, this work focuses on leakage in unsupervised text embedding models and considers leakage of both sensitive attributes and raw input text. In addition, we consider a more realistic scenario where the labeled data is limited for measuring sensitive attributes leakage which is not evaluated in prior works.

Recently and concurrently with this work, Pan et al. [51] also considered the privacy risks of general purpose language models and analyzed model inversion attacks on these models. Our work presents and develops a taxonomy of attacks, which is broader in scope than their work. Our work on inversion also assumes no structures or patterns in input text, which is the main focus of their work, and our work shows that we still recover substantial portion of input data. Additional structural assumptions will lead to a higher recovery rate as is shown in their work.

There is a large body of research on learning privacy-preserving deep representations in supervised models. One popular approach is through adversarial training [17, 74] as detailed in Section 7. The same approach as been applied in NLP models to remove sensitive attributes from the representation for privacy or fairness concern [9, 18, 38]. Another approach for removing sensitive attributes is through directly minimizing the mutual information between the sensitive attributes and the deep representations [47, 50]. We adopt the adversarial training approach during training embedding models as defense against embedding inversion and sensitive attribute inference attacks.

Inverting deep representations. In computer vision community, inverting deep image representation has been studied as a way for understanding and visualizing traditional image feature extractor and deep convolutional neural networks. Both optimization-based approach [41, 69] and learning-based approach [15, 16] as been proposed for inverting image representations. In contrast, we focus on text domain data which is drastically different than image domain data. Text data are discrete and sparse in nature while images are often considered as continuous and dense. Our proposed inversion methods are tailored for unsupervised text embedding models trained with recurrent neural networks and Transformers.

Model inversion attacks [19] use gradient based method to reconstruct the input data given a classifier model and a class label. The reconstruction is often a class representatives from the training data, e.g. averaged face images of female for a gender classifier [45]. Embedding inversion, on the other hand, takes the representation vector as input and reconstruct the exact raw input text.

Membership inference and memorization. Membership inference attacks (MIA) have first been studied against black-box supervised classification models [66], and later on generative models and language models [23, 67]. MIA is closely connected to generalization where overfitted models are prone to the attacks [77]. In this work, we extended the study of MIA to unsupervised word and sentence embedding models without a clear notion of generalization for such models.

It has been shown that deep learning models have a tendency to memorize [78]. Later work showed that adversaries can extract formatted training text from the output of text generation models [4], indicating a real privacy threat caused by memorization. In this work, we focus on embedding models where the output is an embedding vector without the possibility of extract training data directly from the output as in text generation models. Instead, we demonstrated how to measure the memorization in embeddings through MIA.

Differential Privacy. Differentially-private (DP) training of ML models [1, 44] involves clipping and adding noise to instance-level gradients and is designed to train a model to prevent it from memorizing training data or being susceptible to MIA. DP, which limits how sensitive the model is to a training example does not provide a defense against attacks that aim to infer sensitive attributes (which is an aggregate property of training data). The noisy training techniques are challenging for models with large parameters on large amounts of data and sensitive to the hyper-parameters [53]. Our embeddings, with over 10 million parameters in the word embedding matrices $\boldsymbol{V}$ make DP training and hyper-parameter tuning computationally infeasible. Therefore, we leave it to future work to explore how to efficiently train embeddings with DP.

9 CONCLUSIONS

In this paper, we proposed several attacks against embedding models exploring different aspects of their privacy. We showed that embedding vectors of sentences can be inverted back to the words in the sentences with high precision and recall, and can also reveal the authorship of the sentences with a few labeled examples. Embedding models can also leak moderate amount of membership information for infrequent data by using similarity scores from embedding vectors in context. We finally proposed defenses against the information leakage using adversarial training and partially mitigated the attacks at the cost of minor decrease in utility.

Given their enormous popularity and success, our results strongly motivate the need for caution and further research. Embeddings not only encode useful semantics of unlabeled data but often sensitive information about input data that might be exfiltrated in various ways. When the inputs are sensitive, embeddings should not be treated as simply "vectors of real numbers."

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 308–318.

[2] Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. 2019. code2vec: Learning distributed representations of code. *POPL* (2019).

[3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *TACL* (2017).

[4] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security*.

[5] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175* (2018).

[6] Jianfeng Chi, Emmanuel Owusu, Xuwang Yin, Tong Yu, William Chan, Patrick Tague, and Yuan Tian. 2018. Privacy Partitioning: Protecting User Data During the Deep Learning Inference Phase. *arXiv preprint* (2018).

[7] Muthuraman Chidambaram, Yinfei Yang, Daniel Cer, Steve Yuan, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Learning cross-lingual sentence representations via a multi-task dual-encoder model. *arXiv preprint* (2018).

[8] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. 2016. Doctor ai: Predicting clinical events via recurrent neural networks. In *MLHC*.

[9] Maximin Coavoux, Shashi Narayan, and Shay B Cohen. 2018. Privacy-preserving Neural Representations of Text. In *EMNLP*.

[10] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *EMNLP*.

[11] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *RecSys*.

[12] Amine Dadoun, Raphaël Troncy, Olivier Ratier, and Riccardo Petitti. 2019. Location Embeddings for Next Trip Recommendation. In *WWW*.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.

[14] William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *International Workshop on Paraphrasing*.

[15] Alexey Dosovitskiy and Thomas Brox. 2016. Generating images with perceptual similarity metrics based on deep networks. In *CVPR*.

[16] Alexey Dosovitskiy and Thomas Brox. 2016. Inverting visual representations with convolutional networks. In *CVPR*.

[17] Harrison Edwards and Amos Storkey. 2015. Censoring representations with an adversary. In *ICLR*.

[18] Yanai Elazar and Yoav Goldberg. 2018. Adversarial removal of demographic attributes from text data. In *EMNLP*.

[19] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *CCS*.

[20] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *ICML*.

[21] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*.

[22] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint* (2014).

[23] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. 2019. LOGAN: Membership inference attacks against generative models. *PETS* (2019).

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

[25] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*.

[26] Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *arXiv preprint* (2017).

[27] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* (1997).

[28] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. 2008. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLoS genetics* (2008).

[29] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *ICLR*.

[30] Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Laszlo Lukacs, Marina Ganea, Peter Young, et al. 2016. Smart reply: Automated response suggestion for email. In *KDD*.

[31] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*.

[32] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint* (2014).

[33] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NeurIPS*.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NeurIPS*.

[35] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint* (2019).

[36] Meng Li, Liangzhen Lai, Naveen Suda, Vikas Chandra, and David Z Pan. 2017. PrivyNet: A Flexible Framework for Privacy-Preserving Deep Neural Network Training. *arXiv preprint* (2017).

[37] Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING*.

[38] Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018. Towards Robust and Privacy-preserving Text Representations. In *ACL*.

[39] Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *ICLR*.

[40] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. In *SIGDIAL*.

[41] Aravindh Mahendran and Andrea Vedaldi. 2015. Understanding deep image representations by inverting them. In *CVPR*.

[42] Matt Mahoney. 2009. Large text compression benchmark. https://cs.fit.edu/~mmahoney/compression/text.html

[43] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*.

[44] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017. Learning differentially private recurrent language models. In *ICLR*.

[45] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *Symposium on Security and Privacy (S&P)*.

[46] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*.

[47] Daniel Moyer, Shuyang Gao, Rob Brekelmans, Aram Galstyan, and Greg Ver Steeg. 2018. Invariant representations without adversarial training. In *NeurIPS*.

[48] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2018. Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks. *arXiv preprint* (2018).

[49] Bruno A Olshausen and David J Field. 1997. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision research* (1997).

[50] Seyed Ali Osia, Ali Taheri, Ali Shahin Shamsabadi, Minos Katevas, Hamed Haddadi, and Hamid RR Rabiee. 2018. Deep private-feature extraction. *TKDE* (2018).

[51] Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. 2020. Privacy Risks of General-Purpose Language Models. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1314–1331.

[52] Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*.

[53] Nicolas Papernot, Steve Chien, Shuang Song, Abhradeep Thakurta, and Ulfar Erlingsson. 2020. Making the Shoe Fit: Architectures, Initializations, and Tuning for Learning with Privacy. https://openreview.net/forum?id=rJg851rYwH

[54] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

[55] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *arXiv preprint* (2019).

[56] Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M Dai, Nissan Hajaj, Michaela Hardt, Peter J Liu, Xiaobing Liu, Jake Marcus, Mimi Sun, et al. 2018. Scalable and accurate deep learning with electronic health records. *NPJ Digital Medicine* (2018).

[57] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint* (2016).

[58] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP*.

[59] Sebastian Ruder, Parsa Ghaffari, and John G Breslin. 2016. Character-level and multi-channel convolutional neural networks for large-scale authorship attribution. *arXiv preprint* (2016).

[60] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Herve Jegou. 2019. White-box vs Black-box: Bayes Optimal Strategies for Membership Inference. In *ICML*.

[61] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2018. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint* (2018).

[62] Sriram Sankararaman, Guillaume Obozinski, Michael I Jordan, and Eran Halperin. 2009. Genomic privacy and limits of individual detection in a pool. *Nature genetics*

(2009).
[63] Nikunj Saunshi, Orestis Plevrakis, Sanjeev Arora, Mikhail Khodak, and Hrishikesh Khandeparkar. 2019. A Theoretical Analysis of Contrastive Unsupervised Representation Learning. In *ICML*.
[64] Allen Schmaltz, Alexander M Rush, and Stuart M Shieber. 2016. Word Ordering Without Syntax. In *EMNLP*.
[65] Rakshith Shetty, Bernt Schiele, and Mario Fritz. 2018. A4NT: author attribute anonymity by adversarial training of neural machine translation. In *USENIX Security*.
[66] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *Symposium on Security and Privacy (S&P)*.
[67] Congzheng Song and Vitaly Shmatikov. 2019. Auditing Data Provenance in Text-Generation Models. In *KDD*.
[68] Congzheng Song and Vitaly Shmatikov. 2019. Overlearning Reveals Sensitive Attributes. *arXiv preprint* (2019).
[69] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2018. Deep image prior. In *CVPR*.
[70] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.
[71] Ji Wang, Jianguo Zhang, Weidong Bao, Xiaomin Zhu, Bokai Cao, and Philip S Yu. 2018. Not just privacy: Improving performance of private deep learning in mobile cloud. In *KDD*.
[72] Sean Welleck, Zixin Yao, Yu Gai, Jialin Mao, Zheng Zhang, and Kyunghyun Cho. 2018. Loss functions for multiset prediction. In *NeurIPS*.
[73] Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation* (2005).
[74] Qizhe Xie, Zihang Dai, Yulun Du, Eduard Hovy, and Graham Neubig. 2017. Controllable invariance through adversarial feature learning. In *NeurIPS*.
[75] Yinfei Yang, Steve Yuan, Daniel Cer, Sheng-Yi Kong, Noah Constant, Petr Pilar, Heming Ge, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Learning semantic textual similarity from conversations. *arXiv preprint* (2018).
[76] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint* (2019).
[77] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *CSF*.
[78] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. Understanding deep learning requires rethinking generalization. In *ICLR*.
[79] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*.