Privacy Risks of General-Purpose Language Models

Xudong Pan*, Mi Zhang*, Shouling Ji†‡ and Min Yang*
*Fudan University, †Zhejiang University, ‡Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies
Emails: xdpan18@fudan.edu.cn, mi_zhang@fudan.edu.cn, sji@zju.edu.cn, m_yang@fudan.edu.cn

Abstract—**Recently, a new paradigm of building general-purpose language models (e.g., Google's Bert and OpenAI's GPT-2) in Natural Language Processing (NLP) for text feature extraction, a standard procedure in NLP systems that converts texts to vectors (i.e., *embeddings*) for downstream modeling, has arisen and starts to find its application in various downstream NLP tasks and real world systems (e.g., Google's search engine [6]). To obtain general-purpose text embeddings, these language models have highly complicated architectures with millions of learnable parameters and are usually pretrained on billions of sentences before being utilized. As is widely recognized, such a practice indeed improves the state-of-the-art performance of many downstream NLP tasks.**

However, the improved utility is not for free. We find the text embeddings from general-purpose language models would capture much sensitive information from the plain text. Once being accessed by the adversary, the embeddings can be reverse-engineered to disclose sensitive information of the victims for further harassment. Although such a privacy risk can impose a real threat to the future leverage of these promising NLP tools, there are neither published attacks nor systematic evaluations by far for the mainstream industry-level language models.

To bridge this gap, we present the first systematic study on the privacy risks of 8 state-of-the-art language models with 4 diverse case studies. By constructing 2 novel attack classes, our study demonstrates the aforementioned privacy risks do exist and can impose practical threats to the application of general-purpose language models on sensitive data covering identity, genome, healthcare and location. For example, we show the adversary with nearly no prior knowledge can achieve about 75% accuracy when inferring the precise disease site from Bert embeddings of patients' medical descriptions. As possible countermeasures, we propose 4 different defenses (via rounding, differential privacy, adversarial training and subspace projection) to obfuscate the unprotected embeddings for mitigation purpose. With extensive evaluations, we also provide a preliminary analysis on the utility-privacy trade-off brought by each defense, which we hope may foster future mitigation researches.

I. Introduction

With the advances of deep learning techniques in Natural Language Processing (NLP), the last year has witnessed many breakthroughs in building *general-purpose language models* by industry leaders like Google, OpenAI and Facebook [16], [17], [41], [44], [54], [55], [67], [76], which have been widely used in various downstream NLP tasks such as text classification and question answering [15] and start to find its application in real-world systems such as Google's search engine [6], which is said to represent "the biggest leap forward in the past five years, and one of the biggest leaps forward in the history of Search" [6].

Unlike traditional statistical models or shallow neural network models, *general-purpose language models* typically refer to *the family of Transformer-based pretrained giant language models* including Google's Bert and OpenAI's GPT-2, which are composed of layers of Transformer blocks [72] with millions of learnable parameters, and are usually pretrained on billions of sentences before being released. According to the official tutorials [2], users can apply these pretrained models as text feature extractors for encoding sentences into dense vectors, or called *sentence embeddings*, which can be further used for various downstream tasks (e.g., text classification). With the release of Bert, Google AI envisions the future of general-purpose language models as, "anyone in the world can train their own state-of-the-art question answering system (or a variety of other models) in about 30 minutes on a single Cloud TPU, or in a few hours using a single GPU" [4].

Despite the bright envision, for the first time, we observe *these general-purpose language models tend to capture much sensitive information in the sentence embeddings, which leaves the adversary a window for privacy breach*. For example, in a typical use case of these language models in intelligent healthcare, a third-party organization issues a cooperation with a hospital for developing a *patient guide system*, which automatically assigns the patients to a proper department based on the symptom descriptions. Due to the generality of Google's Bert [17], the organization only needs to request the hospital to provide the embeddings of the patients' symptom descriptions as the essential information for building a high-utility system. Due to the lack of understanding of the privacy properties of the general-purpose language models, the hospital may expect sharing the vector-form features would be much less private than sharing the plain text, especially when they are told the encoding rule itself is based on highly complicated neural networks that are near to black-boxes. In fact, we do observe with experiments in Appendix F that, even with a standard decoder module in NLP, it is difficult to recover any useful information from the embeddings. However, on eight state-of-the-art language models including Bert and GPT-2, we devise a lightweight yet effective attack pipeline and strikingly find that *given the unprotected sentence embeddings, even an adversary with nearly zero domain knowledge can infer domain-related sensitive information in the unknown plain text with high accuracy*. In the above medical example, our observation strongly implies that the honest-but-curious service provider as the adversary can easily infer the identity, gender, birth date, disease type or even the precise disease site regarding a particular victim, only if the target piece of information appears in his/her original description.

Our Work. In this paper, we provide the first systematic study on the potential privacy risks of general-purpose language models. Specifically, we want to answer the main research question: *is it possible for the adversary to infer user's private information in the unknown plain text, when the adversary only has access to his/her submitted embeddings?* If the answer is affirmative, then the future applications of state-of-the-art NLP tools in compelling learning paradigms like collaborative or federated learning [36], [37], [46] can be largely threatened and restricted, especially on privacy-critical domains including healthcare, genomics and finance. Besides the novelty in our major research object, our research question also has its own specialness compared with most existing works including membership inference attacks [63], property inference attacks [23] and model inversion attacks [21], in terms of the information source and the attack objective. A more detailed comparison between our study and related attacks can be found in Section II.

Although previous works in computer vision have shown the possibility of reconstructing original images from their pretrained embeddings via autoencoders [18] or generative models [61], *no similar attacks were reported in NLP before.* From our perspective, the discreteness of tokens and the invisibility of the vocabulary are two major technical challenges that prevent the success of reconstruction attacks on text embeddings. On one hand, the discreteness of tokens makes the search over the space of all possible sentences highly inefficient, mainly because the learning objective is no longer differentiable as in the visual cases and therefore gradient-based methods can hardly work [79]. On the other hand, as language models are accessed as black boxes, the adversary has no knowledge of the ground-truth vocabulary, without which the adversary cannot convert the recovered word index sequences into plain text [38]. Even if the adversary may prepare one's own vocabulary, it can be either too small to contain some sensitive words in the unknown plain text or so large that bring high computational complexity.

To address these aforementioned challenges, we propose to reconstruct sensitive information from text embeddings via inference. Taking inspirations from the observation that the privacy-related information in text usually appears in small segments, or is related with the occurrence of certain keywords [62], we construct two different attack classes, namely *pattern reconstruction attacks* and *keyword inference attacks*, to demonstrate how sensitive information can be extracted from the text embeddings. In pattern reconstruction attacks, the raw text has fixed patterns (e.g., genome sequences) and the adversary attempts to recover a specific segment of the original sequences that contains sensitive information (e.g., disease-related gene expressions). In keyword inference attack, the adversary wants to probe whether the unknown plain text (e.g., medical descriptions) contains certain sensitive keyword (e.g., disease site). Focusing on a small segment, the adversary only needs to infer from a limited number of possibilities for reconstruction purposes, which alleviates the optimization difficulty caused by the discreteness of tokens. Meanwhile, the adversary has no need to know the whole vocabulary if the adversary only cares about the word he/she is interested in.

Extensive experiments on 8 state-of-the-art general-purpose language models with 4 (identity-, genome-, medical-, location-related) case studies showed, the adversary can precisely infer various levels of sensitive information of a target user from his/her leaked embeddings. For pattern reconstruction, our attack achieves optimal and average accuracy respectively of 98.2% and 62.4% when inferring the exact nucleotide type at any specified positions from the GPT-2 embeddings of 20-length genome sequences, without any auxiliary information. For keyword inference, our attack achieves average accuracy of 99.8% and 74.8% respectively, when inferring the occurrence of 10 body-related keywords from the Bert embeddings of medical descriptions with and without a shadow corpus. *These results highly demonstrate that the aforementioned privacy risks do exist and can impose real threats to the application of general-purpose language models on sensitive data.* Noticeably, all our attacks only need to access the language model as a cloud service (i.e. ML-as-a-service) and can be conducted with one PC device. With additional ablation studies, we further discuss some architecture-related and data-related factors which may influence the privacy risk level of language models. Furthermore, we also propose and evaluate four possible countermeasures against the observed threats, via quantization, differential privacy [20], adversarial training [57] and subspace projection [14]. We hope our preliminary mitigation study will shed light on future defense researches and contribute to the design of secure general-purpose language models.

In summary, we make the following contributions:

- We discover the potential privacy risks in general-purpose language models by showing, a nearly-zero-knowledge adversary with access to the text embeddings can disclose much sensitive information in the unknown text.
- We design a general attack pipeline for exploiting user privacy in text embeddings and implement two practical attacks with advanced deep learning techniques to demonstrate the privacy risks.
- We present the first systematic evaluations on 8 state-of-the-art general-purpose language models with 4 diverse case studies to demonstrate the hidden privacy risks, with an in-depth analysis on the factors that influence the privacy.
- We also provide preliminary studies on four possible countermeasures and their utility-privacy trade-off, which we hope may foster future defense studies.

II. Related Works

Privacy Attacks against ML. Model inversion attack was first proposed by Fredrikson et al. on statistical models [22] and later generalized to deep learning systems [21]. In terms of the attack objective, Fredrikson et al's attack on image classifiers aims at recovering the prototypical image that represents a specific class, while our attacks aim at recovering partially or fully the plain text behind the embedding. In terms of the information source, model inversion attack mainly relies on the parameters of the model itself, while for our attacks, the information source is the sentence embedding produced from the general-purpose language models.

Meanwhile, Fredrikson et al. [21], [22] also discussed the model inversion attack in the sense that the attack inverts sensitive information about the input from the model's output. To the best of our knowledge, their original attack was mainly implemented for the decision tree model and is not directly applicable to deep learning models. Later, some very recent works have proposed finer-grained attacks which attempt to recover the exact training images or texts from the predictions [58], [77] or the gradients [47], [82] in an unknown mini-batch during the training phase. However, two of them that target on recovering text from gradients [47], [82] utilize the explicit representation of word composition in bag-of-words and cannot be applied to our adversarial setting which reconstructs texts from the dense sentence embeddings from general-purpose language models.

As a complement to model inversion attack, Shokri et al. devised the first membership inference attack against machine learning models [63], which aroused wide research interests [50], [59], [66], [78] in the past few years. In terms of the attack objective, membership inference attempts to disclose the *is-in* relation between the sample and the real private training set. In terms of the information source, the membership inference attack relies on the probability vector associated with the input sample. Different from membership inference, another branch of works called property inference aims at infering whether the training set has certain global property, which was first studied by [10] on shallow models and later extended by [23] to deep models.

Aside from privacy attacks on the datasets, some other threats against the model privacy have also been studied, e.g., by demonstrating the possibility of stealing model parameters [70], architectures [19], and hyper-parameters [73]. In a wider picture of adversarial machine learning, there still remains many open problems including adversarial example [27], data poisoning [13], Byzantine workers [48] and fairness [29], which are calling for future research efforts on building more robust and reliable machine learning systems.

Privacy Attacks using ML. Besides, there are also plenty of prior works using ML approaches to evaluating user privacy risks regarding, e.g., his/her biomedical and geological profiles. On biomedical privacy, for example, Humbert et al. [30], [31] leveraged graphical models to infer the genome of an individual from parental relationships and expert knowledge, which was recently extended to other types of biomedical data by [12]. On location privacy, for example, Shokri et al. [64] used Markov chain modeling to reconstruct the actual traces of users from obfuscated location information, while some recent works exploit side channels from social media like hashtags for location inference using clustering [8] or random forests [80].

III. Preliminaries

A. Sentence Embedding

Given a vocabulary $\mathcal{V}$ that consists of $|\mathcal{V}|$ tokens, we call a sequence $x := (w_1, \ldots, w_n)$ is a *sentence* of length n if each token (or *word*) w_i is in the vocabulary $\mathcal{V}$. Following the nomenclature of representation learning [11], we call a mapping f from sentences to a real vector space $\mathbb{R}^d$ as a feature extractor. For the sentence x, the vector $z := f(x)$ is called its *embedding*.

Prior to the proposal of general-purpose language models, word embedding and sentence embedding as two traditional NLP tasks have been widely studied, for which several mature algorithms exist. For word embedding, algorithms like *word2vec* [49] encode the word to its vector representation that can noticeably preserve the relative semantics between words, e.g., the difference of the embeddings of the words *queen* and *woman* was observed to be almost identical to that of *king* and *man* [49]. For sentence embeddings, word-frequency-based algorithms like *TF-IDF* [60] directly counts word statistics of a sentence and thus the produced sentence embeddings are explicit in word composition, which are not suitable for privacy-critical scenarios [46]. Other learning-based sentence embedding methods like *doc2vec* [42] borrow the idea of *word2vec* and encode sentences to vectors that preserve the relative semantics between the sentence and its composite words in the training corpus. As a result, the produced sentence embeddings from doc2vec are usually corpus-specific and are mainly used for sentence clustering or paraphrase detection on a given corpus [40], [42].

Recently, the boom of general-purpose language models has largely reformed how we understand and use embeddings in the following aspects. On one hand, the boundary between word embedding and sentence embedding are no longer clear due to *contextualized word embeddings* [52], a fundamental concept behind these general-purpose language models. Intuitively, contextualized word embeddings suggest the embedding of the same word can vary according to the sentence where it occurs. For example, the contextualized embedding of the word *apple* should be different in *"I like apple"* and *"I like Apple macbooks"*. Consequently, most general-purpose language models list sentence embedding as one major use case instead of word embedding [2], [74]. On the other hand, sentence embeddings from pretrained general-purpose language models have better generality and can be directly used as input to train downstream learning models. For instance, with a simple linear layer for output, embeddings from a pretrained Bert model can achieve state-of-the-art performance on eleven NLP tasks [17].

B. General-Purpose Language Models for Sentence Embedding

Roughly speaking, existing general-purpose language models are mainly variants of stacked recurrent Transformers, which consist of millions of learnable parameters. Before coming into use, general-purpose language models first need to be pretrained on extremely large corpus such as the English Wikipedia. Typical pretraining tasks include masked language modeling and next sentence prediction [17].

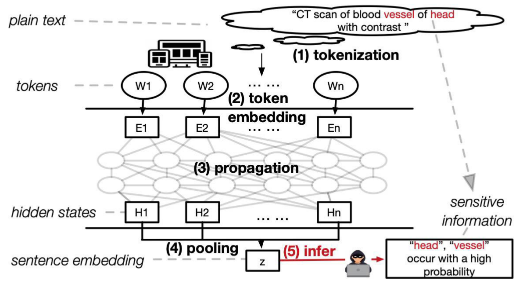

Fig. 1. General-purpose language models for sentence embedding and the potential privacy risks. The red directed line illustrates the discovered privacy risks: the adversary could reconstruct some sensitive information in the unknown plain texts even when he/she only sees the embeddings from the general-purpose language model.

To obtain the embedding of a given sentence x, the following procedures are required [17]: (1) tokenization according to a prepared vocabulary; (2) token embedding (i.e., the token index is mapped to a corresponding vector with the aid of a learnable look-up table); (3) propagation through the Transformers along two dimensions. At the last layer, the sentence is transformed to an n-length sequence of vectors in $\mathbb{R}^d$ (i.e., *hidden states*); and (4) finally, a pooling operation is performed on the hidden states to get the sentence embedding. The pooling operation for general-purpose language models is to take the last hidden state at the final layer as the embedding of sentence x, because most general-purpose language models by default add a special token (i.e, $\langle$CLS$\rangle$, which intuitively means *to classify*) to the end of the input sentence during the pretraining phase. As a result, to use the last hidden state as the sentence embedding usually brings better utility [17], [44]. Fig. 1 provides a schematic view on the aforementioned procedures. Although intuitions on the described workflow suggests that a certain level of context information should be preserved in the last hidden state, there is little known to our community that, to what granularity the original sentence is preserved in the encoding, whether and how the resided sensitive information can be decoded by potential attacks.

C. General-Purpose Language Models in the Wild

TABLE I
BASIC INFORMATION OF MAINSTREAM PRETRAINED LANGUAGE MODELS. (* IMPLIES THE STATISTICS IS ESTIMATED ACCORDING TO THE ORIGINAL PAPER.)

Name	Proposed by	Dimension d	Pretraining Data Size
Bert [17]	Google	1024	13GB
Transformer-XL [16]	Google	1024	517MB*
XLNet [76]	Google	768	76GB
GPT [54]	OpenAI	768	4GB*
GPT-2 [55]	OpenAI	768	40GB
RoBERTa [44]	Facebook	768	160GB
XLM [41]	Facebook	1024	10GB*
Ernie 2.0 [67] (abbr. ERNIE)	Baidu	768	33GB*

As is discussed, training a general-purpose language model from scratch can be highly expensive. As an alternative, most of the state-of-the-art models have a pretrained version published online for free access. In this paper, we study 8 mainstream language models developed by industry leaders including Google, OpenAI, Facebook and Baidu. Table I lists the basic information of these target models.

IV. General Attack Pipeline

Although the state-of-the-art language models provide a direct and effective way for obtaining general-purpose sentence embeddings for various downstream tasks, we find their improved utility is accompanied with hidden privacy risks. By constructing two novel attack classes, we show an adversary is able to reverse-engineer various levels of sensitive information in the unknown plain text from the embeddings. In this section, we first present some general statements of our attacks.

A. Attack Definition

Generally speaking, in both attacks the adversary wants to infer some sensitive information of the sentence from the accessed embeddings. Formally, we formulate the attack model as $\mathcal{A} : z \rightarrow s$, where z is the embedding of a target sentence x and s denotes certain type of sensitive information that can be obtained from the plaintext with a publicly-known algorithm $\mathcal{P} : x \rightarrow s$. For example, from the treatment description "CT scan of blood vessel of head with contrast", we can tell the patient probably has sickness at his/her *head*. In practice, the sensitive information s can be of various types, from a small segment that contains sensitive information (i.e., $\mathcal{P}$ is an operation that takes out a specified part of the whole sequence) to a predicate on the plain text x. For example, in the above *head* case, $\mathcal{P}$ maps any sentence x to $\{0, 1\}$: if the sentence x has word *head*, then $\mathcal{P}(x) = 1$; otherwise $\mathcal{P}(x) = 0$. This notion will be used in formulating our attack pipeline.

B. Threat Model

In general, we focus on the following threat model.

- **Assumption 0.** The adversary has access to a set of embeddings of plain text, which may contain the sensitive information the adversary is interested in.
- **Assumption 1.** For simplicity only, we assume the adversary knows which type of pretrained language models the embeddings come from. *Later in Section VIII, we show this assumption can be easily removed with a proposed learning-based fingerprinting algorithm.*
- **Assumption 2.** The adversary has access to the pretrained language model as an oracle, which takes a sentence as input and outputs the corresponding embedding.

For each attack, we also impose different assumptions on the adversary's prior knowledge of the unknown plain text, which are detailed in the corresponding parts.

C. Attack Pipeline

Our general attack pipeline is divided into four stages. At the first stage, the adversary prepares an external corpus

$\mathcal{D}_{\text{ext}} := \{x_i\}_{i=1}^N$ and uses the algorithm $\mathcal{P}$ to extract the $\{\mathcal{P}(x_i)\}_{i=1}^N$ as labels. It is worth to notice, as the external corpus is basically generated with algorithms or crawled from open domains like Yelp restaurant reviews, the extracted labels usually contain no truly sensitive information. At the second stage, the adversary queries the pretrained language model with each sentence $x_i \in \mathcal{D}_{\text{ext}}$ and receives their embeddings $\{z_i\}_{i=1}^N$. At the third stage, the adversary combines the embeddings with the extracted labels to train an attack model $\mathcal{A}$. At the final stage, the adversary uses the well-trained attack model to infer sensitive information s from the target embedding z. Fig. 2 provides an overview of our attack pipeline. In the next parts, we provide a general introduction of each stage in the pipeline.

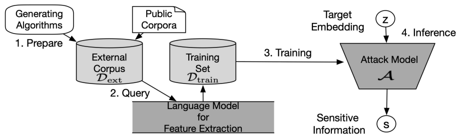
Generating Algorithms
Public Corpora
1. Prepare
External Corpus $\mathcal{D}_{\text{ext}}$
Training Set $\mathcal{D}_{\text{train}}$
2. Query
Language Model for Feature Extraction
3. Training
Target Embedding
z
4. Inference
Attack Model $\mathcal{A}$
Sensitive Information
s

Fig. 2. General attack pipeline.

Stage 1: Prepare External Corpus. The preparation of the training set is accomplished in the first two stages. First, as the attack model infers sensitive information in the unknown plain text, a proper external corpus $\mathcal{D}_{\text{ext}} := \{x_i\}_{i=1}^N$ is therefore essential to play the role of a probing set for successful attacks. Based on different knowledge levels on the plain text, we suggest the adversary can create the external corpus (1) by generating algorithms or (2) from public corpora in open domains. The details are provided in the corresponding sections. After the external corpus is prepared, we apply the algorithm $\mathcal{P}$ on each $x_i \in \mathcal{D}_{\text{ext}}$ to obtain the label $\mathcal{P}(x_i)$, which concludes the first stage.

Stage 2: Query the Language Model. The second stage for training set preparation is to convert the sentences in $\mathcal{D}_{\text{ext}}$ to the corresponding embeddings. Ideally, it is quite straightforward as the adversary only needs to query the language model with each sentence. In practice, according to the knowledge of which model is used, the adversary can deploy the corresponding pretrained model on his/her devices for local query. The adversary may also save some budget by utilizing online language model services [74]. Without loss of generality, our evaluations are conducted in the former setting. More details can be found in Appendix G. At the end of this stage, we have the training set $\mathcal{D}_{\text{train}}$ of the form $\{(z_i, \mathcal{P}(x_i))\}_{i=1}^N$, where z_i is the embedding corresponding to the sentence x_i.

Stage 3: Train the Attack Model. With the training set $\mathcal{D}_{\text{train}}$ at hand, the adversary can now train an attack model g for inference usage. In general, the model is designed as a classifier, which takes the embedding z_i as its input and outputs a probabilistic vector $g(z_i)$ over all possible values of the sensitive information. To train the attack model with the prepared dataset, the adversary needs to solve the following optimization problem with gradient-based algorithms such as Stochastic Gradient Descent (SGD [56]) or Adam [39], $\min_g \frac{1}{N}\sum_{i=1}^N \ell(g(z_i), \mathcal{P}(x_i))$, where ℓ is a loss function that measures the difference between the predicted probabilities and the ground-truth label. Throughout this paper, ℓ is always implemented as the cross-entropy loss.

As a final remark, depending on the knowledge level of the adversary, the architecture of g varies in different settings. For example, knowledgeable attackers will find off-the-shelf classifiers such as logistic regression or linear SVM work surprisingly well, while attackers with no prior knowledge can leverage advanced transfer learning techniques for successful attacks.

Stage 4: Inference. After the training phase, given the target embedding z, the adversary infers the sensitive information based on the following equation $s := \mathcal{A}(z) = \arg\max_{i\in\{1,2,\ldots,K\}}[g(z)]_i$, where $[g(z)]_i$ is the value of $g(z)$ at its i-th dimension and K denotes the total number of possible values for s. In other words, the adversary considers the value with the highest probability as the most possible value of the sensitive information in the unknown sentence x.

V. Pattern Reconstruction Attack

In this section, we focus on the situation when the adversary has knowledge of the generating rule of the unknown plain text, which usually happens when the format of the plain text is common sense (e.g., identity code). We provide this section as a starting point to understand how much sensitive information is encoded in the embeddings from the general-purposed language models.

A. Attack Definition

Intuitively speaking, the pattern reconstruction attack aims at recovering a specific segment of the plain text which has a fixed format. The target segment may contain sensitive information such as birth date, gender or even gene expression. Formally, we construct the pattern reconstruction attack under the following assumption.

- **Assumption 3a.** The format of the plain text is fixed and the adversary knows the generating rules of the plain text.

Following the general statements in Section IV-A, we formally define the routine $\mathcal{P}$ for extracting the sensitive information s from the sentence $x := (w_1, \ldots, w_n)$ as $\mathcal{P}_{\text{pattern}} : (w_1, \ldots, w_n) \to (w_{\text{b}}, \ldots, w_{\text{e}})$, where b and e are the starting and the termination index of the target segment. As $\mathcal{P}$ is assumed to be publicly known, it is also known by the adversary. Therefore, the pattern reconstruction attack w.r.t. $\mathcal{P}_{\text{pattern}}$ can be defined as $\mathcal{A}_{\text{pattern}} : z \to (w_{\text{b}}, \ldots, w_{\text{e}})$.

To be concrete, we provide the following two illustrative examples.

Case Study - Citizen ID (abbr. *Citizen*). Structured information such as identity code or zip code commonly appears in our daily conversations, and these conversations are proved to be useful for training chatbots with the aid of general-purpose language models [55]. However, we find if the messages are not properly cleaned, the adversary, given the sentence embeddings, is capable to recover the structured information with high accuracy and thus conduct further harassment. For example, in many countries, citizen ID is a typical sensitive information for its owner. Once being leaked to the adversary, the identity code can be used to access the victim's other sensitive information or allow the adversary to impersonate the victim to participate in illegal activities [3]. To demonstrate, we consider the case of citizen ID in China, which consists of the 18 characters (from the vocabulary $\{0, \ldots, 9\}$), i.e. 6 for the residence code (3000 possibilities), 8 for the birth date (more than $100 \times 12 \times 30$ possibilities) and 4 for extra code (10^4 possibilities). Consider the adversary wants to recover the exact birth date of the victim via the leaked embedding of his/her citizen ID, we define the mapping P as

$$\mathcal{P}_{\text{citizen}} : |\text{residence}|\text{birthday}|\text{extra}| \rightarrow |\text{birthday}| \qquad (1)$$

Case Study - Genome Sequence (abbr. *Genome*). Roughly, a genome is a sequence of nucleotide which has four different types, namely *A, C, G, T*, as its vocabulary. With increasingly many NLP techniques being applied in computational genetics and pharmacogenomics [43], [45], [81], general-purpose language models are also used in genomics-related tasks. To demonstrate this point, we implement eight benchmark systems by incorporating different general-purpose language models for *splice site prediction* [45], a classical binary classification problem in computational genetics. Basically, our systems exhibit a high utility performance. For example, the splice site prediction system with Google's Bert achieves over 75% classification accuracy. We report the utility of our systems in Fig. Fig. 8(a) of the Appendix and more details in Appendix A.

However, genetic data is highly sensitive in a personalized way – even the nucleotide type at a specific position i in a genome sequence can be related with certain type of genetic decease or characterizes racial information [65] – and thus the adversary is very likely to be interested in recovering the exact nucleotide at a target position. From the disclosed nucleotide, the adversary can further know the gender, race or other privacy-critical information of the victim. For demonstration, we define the mapping P as $\mathcal{P}_{\text{genome},i} : (w_1, w_2, \ldots, w_n) \rightarrow w_i$. In other words, the nucleotide at position i is assumed to be sensitive.

B. Methodology

To realize the attack $\mathcal{A}_{\text{pattern}}$, we present the implementation details on preparation of the external corpus and the architecture of the attack model. In the following parts, we denote the set of all possible values for sequence s as $V(s)$.

1) Generate External Corpus: Knowing the generating rule of the target plain text, the adversary can prepare the external corpus via generating algorithms. A basic generating algorithm generates batches of training samples by randomly sampling from the possible values in $V(x)$, i.e. the set of all possible sentences.

2) Attack Model's Architecture: Naively, the attack model g can be designed as a fully-connected neural network that has input dimension d and output dimension $|V(w_b \ldots w_e)|$, i.e. the number of possible values of the sensitive segment. However, $|V(w_b \ldots w_e)|$ can be very large. For example, in the Citizen case, the number of possible birth dates is near $40,000$. As a result, the free parameters in the attack model will be of large number, which further makes both the batch generation and model training difficult. To tackle this, we follow the divide-and-conquer idea to decompose the attack $\mathcal{A}_{\text{pattern}}$ into small sub-attacks, according to the adversary's knowledge of the format. Again on Citizen, we can decompose the attack model g_{birth} into three *sub-attacks*, namely *year attack* g_{year}, *month attack* g_{month} and *day attack* g_{day}. Each sub-attack model can be independently implemented with fully-connected neural networks of much smaller size and the total parameter number is largely truncated from $O(|V(w_b)| \times \ldots \times |V(w_e)|)$ to $O(|V(w_b)| + \ldots + |V(w_e)|)$. Besides, the generating algorithm can also be decomposed to subroutines for each attack model, so that the training of each sub-module can be conducted in parallel.

C. Experimental Setup

Benchmark Systems.

- Citizen: We randomly generate 1000 citizen IDs according to the generating rule in Eq. 1 as the ground-truth plain text. Then we query the target language model with these citizen IDs to get the corresponding embeddings as the victims.
- Genome: We implement eight genome classification systems for splice site prediction based on a public genome dataset called HS3D (Homo Sapiens Splice Sites Dataset [53]). All the genome sequences are of length 20. We assume the embeddings of genome sequences in the test set, which contains respectively 1000 samples with or without the splice site, are leaked to the adversary.

Attack Implementation.

- Citizen: Following the discussion in Section V-B, we implement the year, month and date sub-attacks as three-layer MLPs which respectively contain 400, 25, 200 hidden units with sigmoid activation. The training batch size is set as 128 for each sub-attack.
- Genome: In practice, we augment the training pair (z, w_i) by concatenating the embedding z of the generated sample with the *positional embedding* p_i for the target position i. We discuss the motivation in Appendix B. Technically, we use the sinusoidal positional embedding as in [72], which has the same dimension as z. Corresponding to this modification, we implement one single attack model for inferring the nucleotide type at any specified position. Different from the Citizen case, this modification will not increase the parameter number as the class number is still 4. The attack model is implemented as a four-layer MLP which takes input $z \oplus p_i$ of dimension $2d$ and has $400, 100$ hidden units with sigmoid activation and intermediate batch normalization layers [32] for faster convergence. For training, we generate mini-batches of size 128 that consist of tuples (z, p_i, w_i), where the positional embedding i is randomly sampled from the interval of possible positions (i.e., $1, \ldots, 20$). For inference, the attacker inputs the victim's embedding and the target

TABLE II
ACCURACY OF SEGMENT RECONSTRUCTION ATTACKS ON CITIZEN.

	Year		Month		Date		Whole	
	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
Bert	0.661	0.926	0.616	0.950	0.539	0.885	0.219	0.384
Transformer-XL	0.725	0.927	**0.802**	**0.992**	**0.839**	**0.992**	**0.488**	**0.624**
XLNet	0.506	0.748	0.484	0.877	0.457	0.797	0.112	0.186
GPT	**0.735**	**0.978**	0.601	0.987	0.630	0.960	0.281	0.434
GPT-2	0.626	0.882	0.664	0.968	0.624	0.927	0.259	0.384
RoBERTa	0.454	0.774	0.441	0.889	0.307	0.703	0.061	0.108
XLM	0.572	0.847	0.509	0.911	0.642	0.908	0.187	0.263
Ernie 2.0	0.584	0.892	0.559	0.924	0.465	0.843	0.152	0.257
Baseline	0.01	0.05	0.083	0.417	0.033	0.167	0.0001	0.0005

position and the model outputs the predicted nucleotide type. More implementation details can be found in Appendix B.

D. Results & Analysis

Table II reports the Top-1 and Top-5 accuracy of the sub-attacks and of inferring the whole birth date with the ensemble attack after $100,000$ iterations of training, where the baseline denotes the performance of a random guesser. Fig. 3 reports the average and per-nucleotide Top-1 accuracy of the attacks on Genome after $100,000$ iterations of training, where we report the proportion of the most frequently appeared nucleotide type as the baseline.

1) Effectiveness & Efficiency: From Table II & Fig. 3, considering the performance of baseline, we can see that our attacks are effective in recovering sensitive segments from their embeddings. For example, when given Transformer-XL (abbr. XL in later sections) embeddings of citizen ID, our attack is able to recover the exact month and date of the victim's birthday with over 80% Top-1 accuracy and recover the whole birth date with over 62% Top-5 accuracy. When given GPT embeddings of genome sequences, our attack achieves near-100% accuracy of inferring the victim's nucleotide type at both ends and over 62% accuracy on average. These results highly demonstrate the effectiveness of our attacks and thus the common existence of privacy risks in the popular industry-level language models.

Moreover, our attack is also efficient in terms of the throughput, which are reported in Table VI of the Appendix. In both cases the attack can learn from over 100 batches in one second. To achieve the reported accuracy, the training takes less than 30 minutes on a medium-end PC. More details of our experimental environment is in Appendix H.

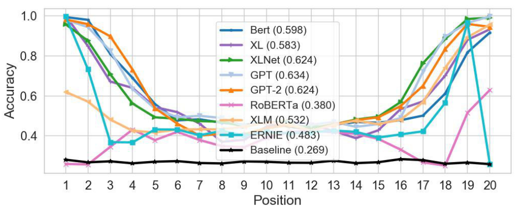

Fig. 3. Accuracy of segment reconstruction attacks on Genome per nucleotide position. The average accuracy is reported in the legend.

2) Comparison among Language Models: First, we notice Facebook's RoBERTa shows stronger robustness than other language models in both cases. By investigating its design, we find RoBERTa is a re-implementation of Google's Bert but uses a different byte-level tokenization scheme (i.e., tokenize sentences in the unit of bytes instead of characters or words) [44]. As RoBERTa shows about 50% lower privacy risks than Bert when facing the same attacks, we conjecture the reason is that the byte-level tokenization scheme may make the embeddings less explicit in character-level sensitive information and thus more robust against our attacks. Similar phenomenon is also observed in the next section. However, RoBERTa suffers a clear utility degradation as a trade-off between utility and privacy. As we can see from Fig. 6(c), the system with Bert achieves an about 33% higher utility performance than that with RoBERTa on Genome. Also, we notice OpenAI's GPT and GPT-2, which share the same architecture but are pretrained on 4GB and 40GB texts, show similar security properties against our attacks and comparable utility performance. Combined with other results, no immediate relatedness is observed between the pretraining data size and the privacy risk level.

3) Other Interesting Findings: From Fig. 3, we can see a majority of the accuracy curves present a valley-like shape, which implies that most language models capture more information of the tokens around the ends than those in the middle, which is probably due to the information at ends usually propagates along the longest path in the recurrent architecture. In other words, the sensitive information which lies at the sentence boundary is more prone to malicious disclosure.

VI. KEYWORD INFERENCE ATTACK

In this section, we study a more general scenario where *the plain text can be arbitrary natural sentences and the knowledge-level of the adversary is much lower*. As a result, successful attacks in this case can impose stronger threats to real-world systems.

A. Attack Definition

The adversary in keyword inference attack is curious about the following predicate, whether certain keyword k is contained in the unknown sentence x. The keyword k can be highly sensitive, which contains indicators for the adversary to further determine e.g., location, residence or illness history of the victim [62].

Before introducing two illustrative examples, we formulate the mapping $\mathcal{P}_{\text{keyword},k}$ for defining the sensitive information related with keyword k from a sentence x as $\mathcal{P}_{\text{keyword},k} : x \to (\exists w \in x, w == k)$, where the right side denotes a predicate that yields *True* if a word w in the sentence x is the target keyword k and otherwise *False*. As the keyword k is specified by the adversary, the routine $\mathcal{P}_{\text{keyword},k}$ is obviously known by him/her. Correspondingly, the keyword inference attack regarding $\mathcal{P}_{\text{keyword},k}$ is defined as $\mathcal{A}_{\text{keyword},k} : z \to (\exists w \in x, w == k)$. Different from pattern reconstruction attacks, keyword inference attacks probes the occurrence of certain keywords instead of exact reconstuction of the whole sequence.

Case Study - Airline Reviews (abbr. *Airline*). Sometimes airline companies survey their customers in order to e.g., improve their customer service. With the aid of advanced NLP techniques, large amounts of airline reviews in text form can be automatically processed for understanding customers' opinion (i.e., opinion mining [69]). As is widely recognized [16], [17], [41], utilizing the pre-trained language models for feature extraction can further improve the utility of many existing opinion mining systems.

However, once accessing the embeddings, the adversary can infer various location-related sensitive information about the victim, including his/her departure, residence, itinerary, etc. As a preliminary step for further attacks, we show the adversary can accurately estimate the probability of whether certain city name is contained in the review.

Case Study - Medical Descriptions (abbr. *Medical*). With the booming of intelligent healthcare, some hospitals tend to build an automatic pre-diagnosis system for more effective service flow [28]. The system is expected to take the patient's description of the illness to predict which department he/she ought to consult. To form a benchmark system, we concatenate the pretrained language models with an additional linear layer for guiding the patients to 10 different departments. Through evaluations, we show the systems can achieve over 90% accuracy on real-world datasets in Fig. 8(b) of the Appendix. More details can be found in Appendix A.

However, when the adversary gets access to the embeddings only, he/she can indeed infer more sensitive and personalized information about the patient as a victim. Besides the department the patient ought to consult, the adversary can further determine other fine-grained information like the disease type or even the precise disease site. To demonstrate, we suppose an adversary wants to pinpoint *the precise disease site* of the victim by inferring the occurrence probability of body-related words in his/her descriptions.

B. Methodology

In this part, we detail our implementations for keyword inference attacks. According to the different levels of the adversary's knowledge on the plain text, the methodology part is divided into *white-box* and *black-box* settings, which respectively require the following two assumptions.

- **Assumption 3b.** The adversary has access to a *shadow corpus*, which consists of sentences that are sampled from the same distribution of the target plain text (which we refer to as *white-box*).
- **Assumption 3c.** The adversary has no information on the target plain text (which we refer to as *black-box*).

Noteworthily, the adversary under Assumption 3c has almost no prior knowledge except that he/she (e.g., any attacker who captures the embeddings) has access to the embeddings, which therefore poses a rather practical threat to the general-purpose language models, while Assumption 3b is also possible to happen in real-world situations when, if we continue the above medical example, some hospital publishes an anonymised dataset of medical descriptions for research purposes [1] or the service provider is honest-but-curious.

Attack in White-Box Settings. Basically, as the adversary has a shadow corpus $\mathcal{D}_{\text{shadow}} := \{(x_i^{'})\}_{i=1}^{N}$ which is sampled from the same distribution as the unknown plain text, he/she can directly use $\mathcal{D}_{\text{shadow}}$ as the external corpus $\mathcal{D}_{\text{ext}}$ and extract the binary label $y_i^{'} = \mathcal{P}_{\text{keyword},k}(x_i^{'})$. Next, the adversary trains a binary classifier with the dataset to conduct $\mathcal{A}_{\text{keyword},k}$. However, we notice in practice the adversary may confront with several pitfalls.

First, the label set $\{y_i^{'}\}_{i=1}^{N}$ can be highly imbalanced. In other words, the sentences with the keyword k (i.e., *the positive samples*) may be in an absolute minority compared to those without k (i.e., *the negative samples*). According to previous researches, imbalance in label will let the attack model prone to overfitting and thus hinder the attack's performance [33]. To alleviate, we propose to randomly replace certain word in the negative samples with the keyword, and we replace the keyword in the positive samples with other random word in the vocabulary (referred to as the *word substitution trick*). After this operation, the original shadow corpus will be twice enlarged and the samples are balanced in both classes.

Next, the shadow corpus after word substitution can still be limited in size, i.e., N is small. In this case, we suggest the adversary should implement their attack model with a Support Vector Machine (SVM), which is especially effective for small sample learning [71]. When M is larger than certain threshold (empirically over 10^3 samples), the adversary can switch to a fully-connected neural network as the attack model, which brings higher attack accuracy.

Attack in Black-Box Settings. The adversary under Assumption 3c faces the most challenging situations, as he/she has merely no prior knowledge of the plain text. In turn, successful attacks in this general scenario will raise a huge threat on the privacy of general-purpose language models.

To implement the keyword inference attack with no prior knowledge, we propose to first crawl sentences from the Internet to form the external corpus and then transfer the adversarial knowledge of an attack model on the external corpus to the target corpus dynamically. Details are as follows.

1) Create the External Corpus from Public Corpora: With the aid of the Internet, it is relatively convenient for the adversary to obtain an external corpus from other public corpora. Next, the adversary can generate positive and negative samples via the same word substitution trick we mentioned in the previous part.

2) Transfer Adversarial Knowledge: During our preliminary attempts, we find if we directly train an off-the-shelf classifier (e.g., linear SVM or MLP) on the external corpus and use it to conduct keyword inference attacks on the target embeddings, the attack's accuracy can sometimes be poor. We speculate it is the *domain misalignment* that causes this phenomenon. To validate, we first train a 3-layer MLP classifier on an external corpus w.r.t. the keyword *head*, which is prepared from the Yelp-Food dataset (i.e., a dataset that consists of restaurant reviews). Next, we plot the decision boundary of the classifier on the external corpus in Fig. 4(a). We also plot the expected decision boundary of the classifier on the target medical dataset that contains 1000 sentences in Fig. 4(b), where the scattered points plot the intermediate representations of the XLNet embeddings at the hidden layer after dimension reduction with Principle Component Analysis (PCA) and the two colors imply whether the plain text contains *head* or not [1]. As we can see, the two decision boundaries are almost orthogonal to each other. As a result, even though the attack model on the public domain (i.e., on restaurant reviews) achieves a near 100% accuracy, its performance is no better than random guess when applied on the private domain (i.e., on medical descriptions).

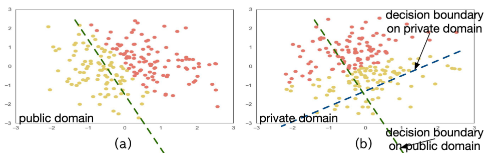

Fig. 4. Domain misalignment between (a) the external corpus and (b) the target corpus, through the lens of the (expected) decision boundary of a MLP classifier trained on the external corpus.

In general, the key challenge here is how to transfer the adversarial knowledge learned by the attack model from the public domain (e.g., Yelp-Food dataset) to the private one (e.g., medical dataset). First, we introduce some essential notations. We denote the public domain and the private domain respectively as $\mathcal{X}_0, \mathcal{X}_1$. Given a training set $\mathcal{D}_{\text{public}} := \{(z_i^{'}, y_i^{'})\}_{i=1}^N$ from $\mathcal{X}$ and some target embeddings $\mathcal{D}_{\text{private}} := \{z_i\}_{i=1}^{n_1}$ from $\mathcal{Y}$, the adversary wants to train an attack model $\mathcal{A}_{\text{keyword},k}$ that performs well on $\mathcal{D}_{\text{private}}$. When the $\mathcal{D}_{\text{private}}$ and $\mathcal{D}_{\text{public}}$ distribute divergently, the straightforward approach works poorly and thus the phenomenon in Fig. 4 occurs.

Therefore, we propose to learn a unified domain-invariant hidden representations for embeddings from $\mathcal{D}_{\text{private}}$ and $\mathcal{D}_{\text{public}}$. To realise this, we are inspired from the idea of Domain-Adversarial Neural Network (DANN) [9] and propose the architecture of our attack model in Fig. 5.

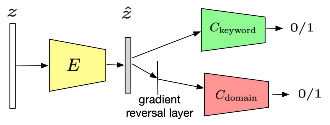

Fig. 5. Architecture of the attack model in the black-box setting.

The model consists of four sub-modules. First, the module E is an encoder which takes the sentence embedding as input and is expected to output a *domain-invariant* representation $\hat{z}$. The hidden representation is followed by two binary classifiers, i.e. C_{keyword} and C_{domain}. The keyword classifier C_{keyword} takes $\hat{z}$ as input and predicts whether the sentence x contains the keyword k, while the domain classifier C_{domain} outputs whether the embedding comes from $\mathcal{X}_0$ or $\mathcal{X}_1$. In practice, we implement E as a nonlinear layer with sigmoid activation and implement C_{keyword} and C_{domain} as two linear layers followed with a softmax layer. For both classifiers, the loss is calculated as the cross-entropy between the output and the ground-truth. Moreover the loss of C_{keyword} is calculated on $\mathcal{D}_{\text{public}}$, while the loss of C_{domain} is calculated on $\{(z_i^{'}, 0)\}_{i=1}^N \cup \{(z_i, 1)\}_{i=1}^{n_1}$.

In our implementations, an additional module called *gradient reversal layer [9]* is fundamental to learn domain-invariant representations and therefore help transfer the adversarial knowledge. The gradient reversal layer is intermediate to the domain classifier and the hidden representation, which works as an identity layer during the forwarding phase and reverses the gradient by putting a minus sign to each coordinate during the back-propagation phase. Intuitively, the gradient reversal layer regularizes the hidden representation $\hat{z}$ by amplifying the keyword-related features and eliminating the domain-related information. Algorithm 1 in Appendix I details a typical iteration in the learning process of our DANN-based attack model. For inference, we take $C_{\text{keyword}} \circ E$ as the attack model g.

C. Experimental Setup

We evaluate the proposed keyword inference attack with two case studies on Airline and Medical in both white-box and black-box settings.

Benchmark Systems.

- Airline: We collect the airline review dataset from Skytrax [5] and preserve the reviews that contain one of the 10 specified city names (e.g., *Bangkok, Frankfurt, etc.*) to form our benchmark dataset. The preprocessed dataset contains 4685 airline reviews (average length 15), and we randomly split the dataset into 10 : 1 to get the test set and the shadow dataset, which is used to simulate the white-box setting. We choose the shadow set to be the much smaller partition to better simulate the real-world situations. We then query the target language models with the reviews in the test set and obtain the embeddings as the victims. In the black-box setting, the adversary only accesses the embeddings of the test set for adversarial knowledge transfer. We suppose the adversary's keyword set as the 10 appeared city names.
- Medical: We implement eight pre-diagnosis systems based on the CMS public healthcare records [1]. These systems are designed to guide patients to the proper department according to the textual description of their disease. We report the utility of the benchmark systems and more implementation details in Appendix A. The preprocessed dataset contains 120,000 disease descriptions of average length 10. We randomly split the dataset into 10 : 1, to form the test set and the shadow dataset. We query the target language models with the descriptions in the test set to form the target set. We suppose the adversary's keyword set contains 10 body-related words (e.g., *head, foot, etc.*) that appear in the dataset.

Metrics. For evaluations, we prepare *balanced* test sets for each target keyword. In detail, we preserve the embeddings of all sentences that contain the keyword from the test set as

[1]More details regarding the external corpus and the medical dataset can be found in the next section.

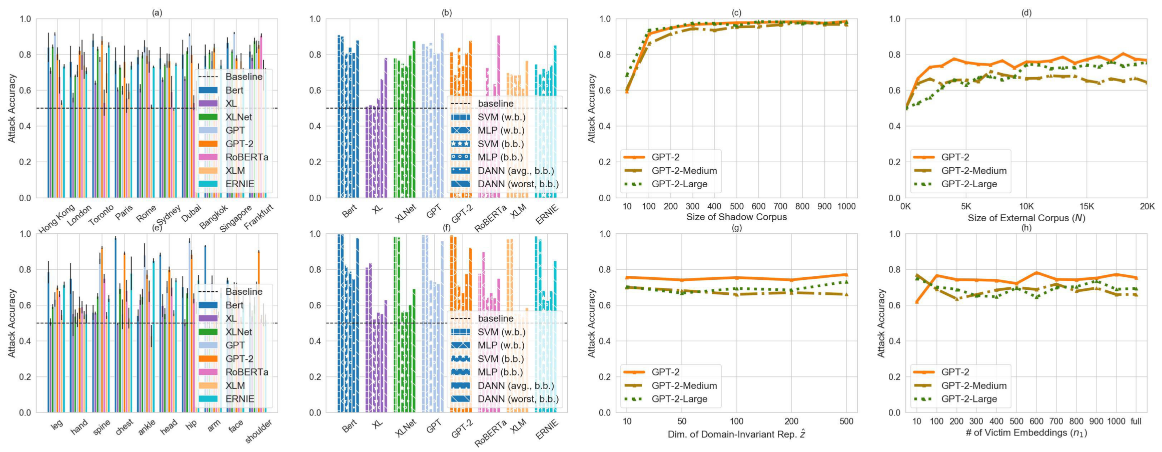

Fig. 6. **(a), (e):** Accuracy of DANN-based attack per keyword on (a) Airline and (e) Medical. **(b), (f):** Accuracy of keyword inference attack on (b) Airline and (f) Medical, averaged on 10 specific city names as keywords. **(c):** Accuracy of MLP-based white-box attack on Medical with varied size of the shadow corpus. **(d), (g), (h):** Accuracy of DANN-based attack on Medical with (d) different size of the external corpus, (f) varied dimension of the domain-invariant representation and (h) varied number of victim embeddings.

the positive samples, and randomly sample the same number of embeddings from the rest of the test set as the negative samples. The statistics of each test set is in Appendix H. We measure the attack's effectiveness on each keyword with the classification accuracy on the prepared test sets. To ensure the attack's effectiveness is not caused by the adversary's knowledge of which keyword to infer, we also conduct black-box attacks with DANN on Airline and Medical to infer 5 random keywords which are not contained in the target corpus.

External Corpus in Black-Box Setting. Following the procedures in Section VI-B, we create the external corpus for the black-box setting from the Yelp-Food dataset, which contains customers' reviews for local restaurants, has less than 20% common words with Medical or Airline and can be replaced with other public corpus. Specifically, we choose 2000 sentences that contain the word *salad* and prepare 2000 positive and negative samples respectively for each keyword inference attack with our proposed word substitution trick.

Attack Implementation.

- White-box setting: We study two implementations of the attack model in the white-box setting, namely the linear SVM and the 3-layer MLP with 80 hidden units with sigmoid activations. The batch size is set as 64.
- Black-box setting: We study three implementations in the black-box setting, namely linear SVM, 3-layer MLP and the DANN-based attacks. The DANN model has 25-dimensional domain-invariant representations and the coefficient λ in Algorithm 1 is set as 1. We use the Adam optimizer with learning rate 0.001 for both the MLP and the DANN-based attacks. The batch size is set as 64.

D. Results & Analysis

Fig. 6(b) & (f) report the performance of keyword inference attacks in white-box and black-box settings with different attack models, respectively on Airline and Medical. The results are averaged on 10 keywords. We also provide the DANN-based attacks' accuracy on each keyword in Fig. 6(a) & (e). Due to the game-theoretical essence of DANN, we notice the accuracy of DANN-based attacks dynamically change over time. We report both the average and the optimal accuracy of the DANN-based attack in 50 epochs to reflect the average and worst-case privacy risk. For the baseline methods SVM & MLP, we report their accuracy after their learning processes converge.

1) Effectiveness & Efficiency: These experimental results highly demonstrate the effectiveness of our attacks in both white-box and black-box settings. For example, from Fig. 6(f), we can see our white-box attack, given Bert's embeddings of the victims' medical descriptions, achieves over 99% accuracy when inferring the occurrence of certain body part, while our black-box attack with no prior knowledge can still achieve over 75% accuracy on average. Similarly, when given the GPT and GPT-2 embeddings of the victim's airline review, our white-box and black-box attacks respectively achieve over 95% and 75% accuracy on average for inferring the occurrence of certain city names. We also report the DANN-based attacks' accuracy on Airline and Medical in Table III, when inferring on 5 random keywords which are not contained in the target sentences. As we can see, our black-box attack achieves over 90% average accuracy in most cases. Moreover, we report the throughput of training attack models in Table VI, which indicates the attacks can be efficiently constructed in less than 2 minutes. Combined with the success of the pattern reconstruction attacks in the previous section, these observations further support our main finding that much sensitive information is encoded in the embeddings from these 8 target language models and can be practically reverse-engineered for malicious purposes.

2) Comparison among Language Models: From Fig. 6(b), we notice Google's XL and Facebook's RoBERTa show much stronger robustness than other language models when facing our white-box attacks on Airline. For these two models, our white-box attacks only outperform the random guesser with a slight margin, while on Medical, white-box attacks aiming at

TABLE III
AVERAGE AND WORST-CASE RISK WHEN DANN-BASED ATTACKS INFER 5 RANDOM KEYWORDS (# OF TEST SAMPLE SIZE = 1000).

Name	Medical (worst/average)	Airline (worst/average)
Bert	1.000/0.860	0.999/0.891
Transformer-XL	0.667/0.555	0.907/0.807
XLNet	0.974/0.951	0.994/0.974
GPT	0.996/0.974	0.998/0.987
GPT-2	0.998/0.973	0.995/0.981
RoBERTa	0.996/0.992	0.998/0.996
XLM	0.948/0.907	0.959/0.925
Ernie 2.0	0.965/0.862	0.964/0.922

these two models achieve around 80% accuracy when attacks on other models are uniformly over 95%. This phenomenon implies the sensitive information in XL's and RoBERTa's embeddings is much harder to be reverse-engineered, which we speculate the major causes as XL's relative small pretraining data size (and thus smaller vocabulary [16]), and RoBERTa's byte-level tokenization scheme. As a result, linear SVM has no sufficient learning capacity [71] to exploit the sensitive information from their embeddings, while the MLP-based attacks suffer from underfitting caused by the limited sample size on Airline (only about 400 on Airline c.f. 2000 on Medical). However, combined with Fig. 6(g), slight utility-privacy trade-off is observed on Medical for XL, which shows about 7% lower utility than the best utility performance achieved by Bert, while no utility-privacy trade-off is observed for RoBERTa, which is probably because the utility performance of most of our systems is high (over 90%) and hence the utility difference is not very clear.

3) Comparison among Attack Implementations: First, comparing the left two columns and the right three columns in each bar group of Fig. 6(b) & (f), we can see the white-box attacks in general are more effective than the black-box counterparts. For example, we notice white-box attacks on Medical show an average 25% margin over the black-box attacks, while exceptions are observed for RoBERTa and XL, which we have discussed above. Next, among the black-box attacks, MLP and DANN attacks show similar effectiveness in many configurations. However, for Facebook's XLNet on Medical, the accuracy of the MLP attack is only 0.560, which corresponds to the domain misalignment in Fig. 4, while the DANN approach improves the attack's accuracy to 0.601 on average and 0.691 in the worst case. Finally, we also investigate the performance of DANN attacks on each keyword. From Fig. 6(a) & (e), we can see different language models have their especially vulnerable keyword. For example, DANN-based attack achieves over 95% accuracy when inferring the word *chest* from Google's Bert embeddings, and over 80% when inferring $ankle$ from Baidu's ERNIE embeddings. We would like to invesigate the fundamental cause of this interesting phenomenon in a future work.

4) Ablation Study: We also conduct an overall ablation study by investigating the keyword inference attack in a wide range of configurations on three variants of OpenAI's GPT-2 (namely GPT-2, GPT-2-Medium, GPT-2-Large), which are pretrained on the same corpus, but increments in parameter numbers and embedding dimension [55]. Detailed statistics can be found in Appendix H.

First, we study the the impact of external corpus size on the attack effectiveness: (1) For the white-box setting, we vary the size of the shadow corpus, which represents the knowledge level of the adversary in the white-box setting, in $\{10, 100, \ldots, 1000\}$ and conduct the MLP attack on Medical. The results are provided in Fig. 6(c). As we can see, the attack accuracy remains over 90% for each language model when the shadow corpus size is larger than 100. Moreover, we interestingly observe, a larger language model (GPT-2-Large) is less robust than a smaller one when the adversary's knowledge is limited. When the shadow corpus size is only 10, the attack accuracy is 68.5%, 61.0% and 59.4% against GPT-2-Large, GPT-2-Medium and GPT-2 respectively. This observation is also consistent with the conclusion in [34]: complicated models tend to enlarge the attack surface. (2) Similarly, for the black-box setting, we compare the DANN attack's performance by varying the size of the external corpus from 100% to 5% of the original size 2000, with results reported in Fig. 6(d). As we can see, a larger external corpus helps our attack achieve higher accuracy, which demonstrates the effectiveness of our proposed adversarial knowledge transfer procedure.

Also, we study the robustness of DANN attack w.r.t. its hyperparameters. We respectively control the size of the victim embeddings, which corresponds to the knowledge level of the adversary in the black-box setting, and the dimension of the domain-invariant representation in DANN, which is the only architectural factor of DANN, and conduct the DANN attack on Medical. For these two settings, we report the attack accuracy respectively in Fig. 6(g) & (h). As is shown, the accuracy of DANN attack remains high with different hyperparameter choices, which highly reflects the robustness and effectiveness of our proposed attack model.

VII. POSSIBLE DEFENSES

As the sentence embedding is the direct source of potential information leakage, a general principle for mitigation is to obfuscate the embeddings. For this purpose, we empirically evaluate four possible technical choices, where the first three are general against both attacks and the last one is specially designed for keyword inference attack. Although the ideal situation is that the sensitive information can be totally eliminated while the required information for other normal tasks can be highly preserved, in practice such a utility-privacy trade-off seems unavoidable, at least according to our reported trade-off results below. Here, the *utility* denotes the classification accuracy of the underlying benchmark systems. We hope our preliminary study will foster future mitigation studies. The omitted technical details and experimental setups can be found in Appendix C.

(1) Rounding. For the first defense, we apply floating-point rounding on each coordinate of the sentence embeddings for obfuscation. Formally, we write the rounding defense as $\hat{z} =$

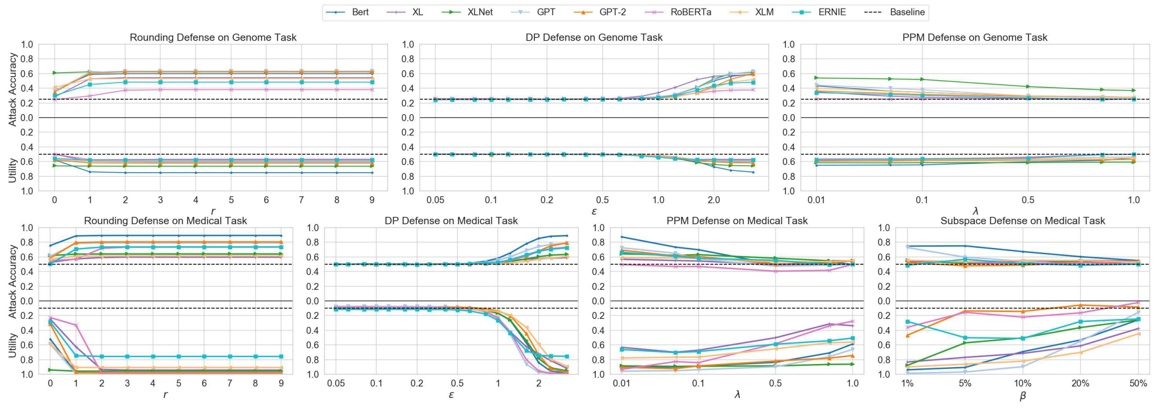

Fig. 7. The utility and attack accuracy curves along on Genome and Medical when four possible defenses with different parameters are applied for mitigation. For DP & PPM defenses, the x-axes of utility and attack accuracy curves are in log scale.

rounding(z, r), where the non-negative integer r denotes the number of decimals preserved after rounding.

(2) Laplace Mechanism. For the second defense, we leverage a differential privacy approach, the *Laplace mechanism* [20]. Roughly speaking, we perturb the embedding coordinate-wise with samples from a Laplace distribution whose parameters are determined by the ℓ_1-*sensitivity* of the language model f (denoted as Δf, which we estimate with numeric algorithms). Formally, the defense works as $\hat{z} = z + (Y_1, \ldots, Y_d)$, where Y_i are i.i.d. random samples drawn from $Lap(\Delta f/\epsilon)$, the Laplace distribution with location 0 and scale $\Delta f/\epsilon$.

(3) Privacy Preserving Mapping. The third defense is based on adversarial training. We borrow the notion of a Privacy Preserving Mapping (PPM) from [57] to denote a mapping $D_\theta : \mathbb{R}^d \to \mathbb{R}^d$ parameterized by θ, which is trained to minimize the effectiveness of an imagined adversary $\mathcal{A}_\psi$. Meanwhile, the PPM is required to follow the utility constraint by distorting the embeddings only in a limited radius around the original embedding, which is implemented as a regularization term. Formally, we propose to learn the privacy preserving mapping D_θ by solving the following minimax game $\min_\theta \max_\psi \frac{1}{n} \sum_{i=1}^{n} \mathcal{A}_\psi(D_\theta(z_i), s_i) + \lambda \|D_\theta(z_i) - z_i\|_2$ where λ controls the privacy level, the higher the lower privacy level.

(4) Subspace Projection. The last defense is especially designed as a countermeasure to keyword inference attacks, inspired by [14] on debiasing word embeddings from gender bias. The general idea of this defense is to project out the unwanted subspace (i.e., *privacy subspace*) that encodes the occurrence of the keyword from the universal sentence embedding space. Technical details on how to identify the privacy subspace and how to do the projection can be found in Appendix C. In our evaluations, we consider the ratio β between the dimension of the privacy subspace and that of the universal embedding space as the parameter of this defense. Intuitively, a higher β is expected to bring a stricter privacy mechanism.

Evaluations. We evaluate the first three defenses against the pattern reconstruction attack on Genome and all four defenses against the DANN-based keyword inference attack on Medical with a wide range of settings. The configurations and results of the first three defenses are presented in Fig. 7.

As we can see from Fig. 7, although each defense can attenuate the attacker to a total random guesser under certain privacy budgets, they simultaneously compromise the utility of downstream tasks by causing an unacceptable degradation. For example, the Laplace mechanism degrades the utility to a total random guesser as well when achieving the optimal defense performance. For PPM, despite a slighter trade-off is observed, the utility for RoBERTa and Transformer-XL still degrades from over 90% to around 25% when the optimal defense is achieved. Among these defenses, we notice the subspace projection defense could provide a more desirable defense quality than the three possible defenses. For example, for most of the target language models, the defense can degrade the DANN attack to a random guesser by projecting out only the 1% keyword-related subspace. However, the utility of the embeddings on the downstream task still decreases by about 15% compared with the 95% accuracy with unprotected embeddings, which implies the keywords that we want to hide are also critical to provide essential semantics for downstream tasks. Moreover, the quality of subspace defense in practice would be less desirable due to its blindness to the target keyword that the adversary is interested in.

According to our preliminary results above, how to balance the elimination of token-level sensitive information from the embeddings and the preservation of the essential information for normal tasks is still an open problem that awaits more in-depth researches. In consideration of the practical threats imposed by our attacks on the applications of general-purpose language models, we highly suggest the exploration of effective defense mechanisms as a future work.

VIII. Discussions

On Threat Model. For Assumption 0, the adversary can get the sentence embeddings of victims if general-purpose language models are deployed in collaborative or federated learning systems, especially when a) the service provider itself wants to snoop user's sensitive information or b) the embeddings are shared accidentally or abusively with some malicious attackers. In some recent protocols, the feature may be encrypted with homomorphic encryption schemes for privacy protection [24], which therefore requires an adversary to first encrypt the embeddings of the external corpus with the public key and train the attack models on the encrypted external corpus. This is an interesting scenario that deserves dedicated research and we leave this as a future work. Nevertheless, there are also many scenarios which are not suitable for homeomorphic encryption schemes due to efficiency issues, such as real-time long passage translation [75] or search engines with language models [6]. In these scenarios, our attacks remain huge threats.

For Assumption 1, we devise a learning-based fingerprinting algorithm by first determining the candidate model types based on the dimension size and then pinpointing the exact model type with a pretrained classifier. Strikingly, we find the classification accuracy can achieve 100%. More technical details and analysis can be found in Appendix D.

For Assumption 2, the adversary can easily satisfy the assumption by deploying the language model on local devices or accessing the corresponding online services. In the current work, we adopt this assumption considering the generality of our attack. Nevertheless, the adversary could further exploit the specific language model architecture and pretrained parameters for better attack effectiveness, which is an interesting and meaningful direction to pursue in the future work.

Downstream Attacks. As we have mentioned in Sections V & VI, our proposed attacks can be further used for downstream attacks that can cause more severe consequences. For example, if the attacker for some reason gets the embedding of the treatment description "CT scan of blood vessel of head with contrast", he/she can utilize the proposed keyword inference attack to infer the occurrence probability of each word in a customized vocabulary (e.g., the vocabulary contains *head* and *vessel* because they are frequent words in medical descriptions), sort the occurrence probability in the decreasing order (e.g., the two words *head* and *vessel* both have occurrence probability higher than 90%) and thus inverts out the meaning of the sentence (e.g., *the patient may have something abnormal in the blood vessel of his head*). Appendix E provides a demonstrative experiment that implement the above guideline. We find the adversary can indeed reassemble the basic semantic meaning of the original text with the above procedure, even if some words may not be in the customized vocabulary.

Utility vs. Privacy in Deploying Sentence Embeddings. Our current work indicates the improved utility of sentence embeddings from general-purpose language models is at odds with the privacy. In principle, to balance the utility-privacy trade-off requires the sentence embedding to preserve the information that is desired for the downstream task and to discard the remainder, while the following dilemma happens for general-purpose language models: these models are especially designed to provide embeddings that can be used for a wide range of downstream tasks [17], which consequently enforces the embeddings to preserve much token-level information, which is critical in forming semantics in many cases and hence leaves the adversary a window for privacy breach. Based on our systematic evaluations of eight state-of-the-art language models, we find the byte-level tokenization scheme may indeed provide additional privacy protection by design. In the meantime, obfuscating sentence embeddings via adversarial training or subspace projection may be a promising direction for future studies as they can achieve more desirable utility-privacy trade-off.

Limitations & Future Works. Although we have observed some interesting differences in the security property of different language models, we are still not very clear about how many other design choices, including the network depth, learning algorithms and hyper-parameters, influence the corresponding language model's privacy level. We are interesting to investigate these issues in a future work. Moreover, although we provided preliminary study on four possible defenses, we find none of them could achieve an optimal balance between privacy and utility on downstream tasks. Also, due to the hardware constraints, we have not evaluated the defense quality of differentially private training techniques (e.g., DPSGD [7]). We hope our work will draw more attentions from researchers to conduct more in-depth study on the privacy properties of this new NLP paradigm and the corresponding mitigation approaches.

IX. Conclusion

In this paper, we design two novel attack classes, i.e., pattern reconstruction attacks and keyword inference attacks, to demonstrate the possibility of stealing sensitive information from the sentence embeddings. We conduct extensive evaluations on eight industry-level language models to empirically validate the existence of these privacy threats. To shed light on future mitigation studies, we also provide a preliminary study on four defense approaches by obfuscating the sentence embeddings to attenuate the sensitive information. To the best of our knowledge, our work presents the first systematic study on the privacy risks of general-purpose language models, along with the possible countermeasures. For the future leverage of the cutting-edge NLP techniques in real world settings, we hope our study can arouse more research interests and efforts on the security and privacy of general-purpose language models.

Acknowledgement

We sincerely appreciate the shepherding from Piotr Mardziel. We would also like to thank the anonymous reviewers for their constructive comments and input to improve our paper. This work was supported in part by the National Natural Science Foundation of China (61972099, U1636204, U1836213, U1836210, U1736208, 61772466, U1936215, and U1836202), the National Key Research and Development Program of China (2018YFB0804102), the Natural Science Foundation of Shanghai (19ZR1404800), the Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under No. LR19F020003, the Ant Financial Research Funding, and the Alibaba-ZJU Joint Research Institute of Frontier Technologies. Min Yang is the corresponding author, and a faculty of Shanghai Institute of Intelligent Electronics & Systems, Shanghai Institute for Advanced Communication and Data Science, and Engineering Research Center of CyberSecurity Auditing and Monitoring, Ministry of Education, China.

References

[1] "Cms public healthcare dataset," https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Physician-and-Other-Supplier2016.html, accessed: 2019-09-10.

[2] "Google-research/bert," https://github.com/google-research/bert, accessed: 2020-1-4.

[3] "Identity theft continuing problem in the state, nation," https://www.enidnews.com/news/identity-theft-continuing-problem-in-the-state-nation/article_6c1cf034-f40e-11e9-8401-07cb9f8a7875.html, accessed: 2019-10-24.

[4] "Open sourcing bert: State-of-the-art pre-training for natural language processing," https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html, accessed: 2020-1-4 Published 2018-11-02.

[5] "Skytrax dataset," https://github.com/quankiquanki/skytrax-reviews-dataset, accessed: 2019-09-10.

[6] "Understanding searches better than ever before," https://blog.google/products/search/search-language-understanding-bert, accessed: 2019-09-10.

[7] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," *ArXiv*, vol. abs/1607.00133, 2016.

[8] B. Agir, K. Huguenin, U. Hengartner, and J.-P. Hubaux, "On the privacy implications of location semantics," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, pp. 165 – 183, 2016.

[9] H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, and M. Marchand, "Domain-adversarial neural networks," *ArXiv*, vol. abs/1412.4446, 2014.

[10] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *IJSN*, vol. 10, pp. 137–150, 2013.

[11] Y. Bengio, A. C. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *TPAMI*, vol. 35, pp. 1798–1828, 2012.

[12] P. Berrang, M. Humbert, Y. Zhang, I. Lehmann, R. Eils, and M. Backes, "Dissecting privacy risks in biomedical data," *Euro Security & Privacy*, pp. 62–76, 2018.

[13] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *ICML*, 2012.

[14] T. Bolukbasi, K.-W. Chang, J. Y. Zou, V. Saligrama, and A. T. Kalai, "Man is to computer programmer as woman is to homemaker? debiasing word embeddings," in *NIPS*, 2016.

[15] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, and I. Androutsopoulos, "Large-scale multi-label text classification on eu legislation," in *ACL*, 2019.

[16] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," in *ACL*, 2019.

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2018.

[18] A. Dosovitskiy and T. Brox, "Inverting visual representations with convolutional networks," *CVPR*, pp. 4829–4837, 2016.

[19] V. Duddu, D. Samanta, D. V. Rao, and V. E. Balas, "Stealing neural networks via timing side channels," *ArXiv*, vol. abs/1812.11720, 2018.

[20] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, pp. 211–407, 2014.

[21] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *CCS*, 2015.

[22] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," vol. 2014, pp. 17–32, 2014.

[23] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property inference attacks on fully connected neural networks using permutation invariant representations," in *CCS*, 2018.

[24] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. E. Lauter, M. Naehrig, and J. R. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *ICML*, 2016.

[25] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[26] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014.

[27] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *ArXiv*, vol. abs/1412.6572, 2014.

[28] W. Guo, J. Shao, R. Lu, Y. Liu, and A. A. Ghorbani, "A privacy-preserving online medical prediagnosis scheme for cloud environment," *IEEE Access*, vol. 6, pp. 48 946–48 957, 2018.

[29] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," in *NIPS*, 2016.

[30] M. Humbert, E. Ayday, J.-P. Hubaux, and A. Telenti, "Addressing the concerns of the lacks family: quantification of kin genomic privacy," in *CCS*, 2013.

[31] ——, "Quantifying interdependent risks in genomic privacy," *ACM Trans. Priv. Secur.*, vol. 20, pp. 3:1–3:31, 2017.

[32] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *ArXiv*, vol. abs/1502.03167, 2015.

[33] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intell. Data Anal.*, vol. 6, pp. 429–449, 2002.

[34] Y. Ji, X. Zhang, S. Ji, X. Luo, and T. Wang, "Model-reuse attacks on deep learning systems," in *CCS*, 2018.

[35] J. Jia and N. Z. Gong, "Attriguard: A practical defense against attribute inference attacks via adversarial machine learning," *ArXiv*, vol. abs/1805.04810, 2018.

[36] A. Jochems, T. Deist, I. E. Naqa, M. L. Kessler, C. Mayo, J. Reeves, S. Jolly, M. Matuszak, R. T. Haken, J. van Soest, C. J. G. Oberije, C. Faivre-Finn, G. J. Price, D. K. M. D. Ruysscher, P. Lambin, and A. Dekker, "Developing and validating a survival prediction model for nsclc patients through distributed learning across 3 countries," in *International journal of radiation oncology, biology, physics*, 2017.

[37] A. Jochems, T. Deist, J. van Soest, M. J. Eble, P. Bulens, P. A. Coucke, W. J. F. Dries, P. Lambin, and A. Dekker, "Distributed learning: Developing a predictive model based on data from multiple hospitals without data leaving the hospital - a real life proof of concept." *Radiotherapy and oncology : journal of the European Society for Therapeutic Radiology and Oncology*, vol. 121 3, pp. 459–467, 2016.

[38] D. Jurafsky, "Speech and language processing," 2006.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ArXiv*, vol. abs/1412.6980, 2014.

[40] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *NIPS*, 2015.

[41] G. Lample and A. Conneau, "Cross-lingual language model pretraining," *ArXiv*, vol. abs/1901.07291, 2019.

[42] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," *ArXiv*, vol. abs/1405.4053, 2014.

[43] B. Lee, T. Lee, B. Na, and S. Yoon, "Dna-level splice junction prediction using deep recurrent neural networks," *ArXiv*, vol. abs/1512.05135, 2015.

[44] Y. Liu, M. Ott, N. Goyal, J. Du, M. S. Joshi, D. Chen, O. Levy, M. Lewis, L. S. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *ArXiv*, vol. abs/1907.11692, 2019.

[45] F. MASTEROPPGAVE and Ø. Johansen, "Gene splice site prediction using artificial neural networks," 2008.

[46] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2016.

[47] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Security & Privacy*, 2019.

[48] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in byzantium," in *ICML*, 2018.

[49] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *NIPS*, 2013.
[50] M. Nasr, R. Shokri, and A. Houmansadr, “Machine learning with membership privacy using adversarial regularization,” in *CCS*, 2018.
[51] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” 2017.
[52] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. S. Zettlemoyer, “Deep contextualized word representations,” *ArXiv*, vol. abs/1802.05365, 2018.
[53] P. Pollastro and S. Rampone, “Hs3d: Homo sapiens splice site data set,” 2002.
[54] A. Radford, “Improving language understanding by generative pretraining,” 2018.
[55] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” 2019.
[56] H. E. Robbins, “A stochastic approximation method,” 2007.
[57] S. Salamatian, A. Zhang, F. du Pin Calmon, S. Bhamidipati, N. Fawaz, B. Kveton, P. Oliveira, and N. Taft, “Managing your private and public data: Bringing down inference attacks against your privacy,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, pp. 1240–1255, 2015.
[58] A. Salem, A. Bhattacharyya, M. Backes, M. Fritz, and Y. Zhang, “Updates-leak: Data set inference and reconstruction attacks in online learning,” *ArXiv*, vol. abs/1904.01067, 2019.
[59] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, “Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models,” *ArXiv*, vol. abs/1806.01246, 2018.
[60] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” *Inf. Process. Manage.*, vol. 24, pp. 513–523, 1988.
[61] G. Shen, K. Dwivedi, K. Majima, T. Horikawa, and Y. Kamitani, “End-to-end deep image reconstruction from human brain activity,” in *Front. Comput. Neurosci.*, 2019.
[62] R. Shetty, B. Schiele, and M. Fritz, “A4nt: Author attribute anonymity by adversarial training of neural machine translation,” in *USENIX Security Symposium*, 2017.
[63] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” *Security & Privacy*, pp. 3–18, 2017.
[64] R. Shokri, G. Theodorakopoulos, J.-Y. L. Boudec, and J.-P. Hubaux, “Quantifying location privacy,” *Security & Privacy*, pp. 247–262, 2011.
[65] S. S. Shringarpure and C. D. Bustamante, “Privacy risks from genomic data-sharing beacons.” *American journal of human genetics*, vol. 97 5, pp. 631–46, 2015.
[66] L. Song, R. Shokri, and P. Mittal, “Privacy risks of securing machine learning models against adversarial examples,” *ArXiv*, vol. abs/1905.10291, 2019.
[67] Y. Sun, S. Wang, Y. Li, S. Feng, H. Tian, H. Wu, and H. Wang, “Ernie 2.0: A continual pre-training framework for language understanding,” *ArXiv*, vol. abs/1907.12412, 2019.
[68] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *NIPS*, 2014.
[69] D. Tang, Y. Zhao, and T. Liu, “Document modeling with gated recurrent neural network for sentiment classification,” in *EMNLP*, 2015.
[70] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, “Stealing machine learning models via prediction apis,” *ArXiv*, vol. abs/1609.02943, 2016.
[71] V. N. Vapnik, “The nature of statistical learning theory,” in *Statistics for Engineering and Information Science*, 1995.
[72] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” pp. 5998–6008, 2017.
[73] B. Wang and N. Z. Gong, “Stealing hyperparameters in machine learning,” *Security & Privacy*, pp. 36–52, 2018.
[74] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, “Huggingface’s transformers: State-of-the-art natural language processing,” *ArXiv*, vol. abs/1910.03771, 2019.
[75] Y. Wu and M. S. et al., “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *ArXiv*, vol. abs/1609.08144, 2016.
[76] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, “Xlnet: Generalized autoregressive pretraining for language understanding,” *ArXiv*, vol. abs/1906.08237, 2019.
[77] Z. Yang, J. Zhang, E.-C. Chang, and Z. Liang, “Neural network inversion in adversarial setting via background knowledge alignment,” in *CCS*, 2019.
[78] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, “Privacy risk in machine learning: Analyzing the connection to overfitting,” *CSF*, pp. 268–282, 2017.
[79] L. Yu, W. Zhang, J. Wang, and Y. Yu, “Seqgan: Sequence generative adversarial nets with policy gradient,” *ArXiv*, vol. abs/1609.05473, 2016.
[80] Y. Zhang, M. Humbert, T. A. Rahman, C.-T. Li, J. Pang, and M. Backes, “Tagvisor: A privacy advisor for sharing hashtags,” *WWW*, 2018.
[81] Y. Zhang, X. Liu, J. N. MacLeod, and J. Liu, “Discerning novel splice junctions derived from rna-seq alignment: a deep learning approach,” in *BMC Genomics*, 2018.
[82] L. Zhu, Z. Liu, and S. Han, “Deep leakage from gradients,” in *NeurIPS*, 2019.

APPENDIX

A. *Implementation of Utility Models*

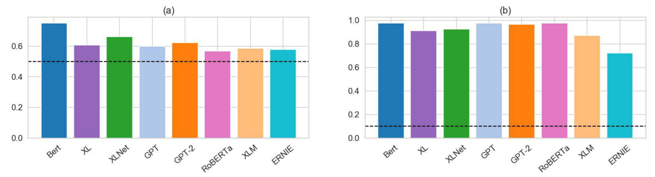

Fig. 8. Utility of benchmark systems on (a) Genome and (b) Medical.

1) Genome: We implemented eight genome classification systems for splice site prediction based on a public genome dataset called HS3D (Homo Sapiens Splice Sites Dataset) [2]. Roughly speaking, splice site prediction is a binary classification task in computational genetics which determines whether the target nucleotide sequence contains certain functional unit. To build the benchmark systems, we first prepared a dataset from HS3D that consists 28800 (2880) negative (positive) samples for training and 1000 (1000) samples for testing. All the genome sequences are of length 20. Each system is composed with a pretrained language model for feature extraction and a three-layer MLP of 200 hidden units with sigmoid activation for classification. Fig. 8(a) reports the utility (in accuracy) of our benchmark system when incorporating different language models, where the non-trivial margin over the random guess demonstrates their effectiveness.

2) Medical: We implemented eight pre-diagnosis systems based on the CMS public healthcare records [3]. These systems are designed to guide patients to the proper department according to the textual description of their decease. The preprocessed dataset contains $120,000$ decease descriptions from 10 medical departments (e.g., Orthopedic Surgery, Anesthesiology, Dermatology and so on). We model the pre-diagnosis task as 10-class classification, and implemented the systems as a combination of the pretrained language models for feature extraction and a three-layer MLP of 200 hidden units with sigmoid activation for classification. First, the classification

[2]http://www.sci.unisannio.it/docenti/rampone/

[3]https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Physician-and-Other-Supplier2016.html

accuracy in Fig. 8(b) highly demonstrates the utility of our benchmark systems.

B. Implementation Details of Attack Model on Genome

In practice, we find training an attack model targeted on the i-th nucleotide is ineffective if we only use the pairs of embeddings and the corresponding nucleotide types. We speculate it is probably because the training sample itself contains insufficient information. Consider the example of sequence $ACGTAACT$ and the attacker targeted at the fourth nucleotide. If we only supply the learning model with its embedding and the type T, the attack model has actually no idea of whether the learning objective is to infer the T at the target position, or the T at the tail. To solve this problem, we propose to augment the training pair (z, w_i) with auxiliary information regarding the target position, in the form of *positional* for the target position i.

Basically, we concatenate the embedding z of the generated sample with the positional embedding p_i for position i. Formally, we use the sinusoidal positional embedding as in [72], which is defined by $p_{i,2k} = \sin{(i/10000^{2k/d_{\text{pos}}})}, p_{i,2k+1} = \cos{(i/10000^{(2k+1)/d_{\text{pos}}})}$, where $p_{i,2k}$ denotes the $2k$-th coordinate of p_i and d_{pos} is its dimension. In our implementation, we set d_{pos} equal to the dimension of z. Corresponding to this modification, we implement one single attack model for inferring the nucleotide type at any specified position. Different from the Citizen case, this modification will not increase the parameter number as the class number is still 4. The attack model is implemented as a three-layer MLP which takes input $z \oplus p_i$ of dimension $2d$ and has 200 hidden units with sigmoid activation and intermediate batch normalization layers [32] for faster convergence. For training, we generate mini-batches of size 128 that consist of tuples (z, p_i, w_i), where the positional embedding i is randomly sampled from the interval of possible positions (i.e. $1, \ldots, 20$). For inference, the attacker inputs the victim's embedding and the targeted position and the model outputs the predicted nucleotide type.

C. Omitted Details on Defenses

Laplace Mechanism. For the second defense, we leverage a differential privacy approach for mitigation. Specifically, we apply the *Laplace mechanism* introduced in [20] to protect the original embedding from privacy breaching. Roughly speaking, the Laplace mechanism perturbs the embedding coordinate-wise with samples from a Laplace distribution whose parameters are determined by the ℓ_1-*sensitivity* of the language models (denoted as f). Formally, we propose the DP-based defense as $\hat{z} = D_{\text{lap},f,\epsilon}(z) \doteq z + (Y_1, \ldots, Y_d)$, where Y_i are i.i.d. random samples drawn from *Lap*$(\Delta f/\epsilon)$, the Laplace distribution with location 0 and scale $\Delta f/\epsilon$. Here Δf denotes the ℓ_1-sensitivity of the language model, which is defined as $\Delta f = \max_{x,x' \in \mathbb{N}^{|\mathcal{V}|}, \|x-x'\|_1 = 1} \|f(x) - f(x^{'})\|_1$ where $x^{'}, x$ are sentences that are different only in one position. Intuitively, the Laplace mechanism makes it harder for the adversary to distinguish from the embeddings the difference caused by alteration of one word, which thus defends both the pattern reconstruction attack and keyword inference attack in a straightforward way. In theory, it can be proved that the language model with protection, i.e. $D_{\text{lap},f,\epsilon} \circ f$ is $(\epsilon, 0)$-differential private. In practice, we estimate the ℓ_1-sensitivity of each language model by generating $10,000$ pairs of $(x, x^{'})$ by word substitution, querying the language models and calculating Δf according to the definition. The numeric value is provided in Table IV. However, there still exists many theoretical challenges in bounding the errors of the estimated L1 sensitivity, which will be a meaningful direction to pursue in the future.

TABLE IV
ESTIMATED ℓ_1-SENSITIVITY OF EACH LANGUAGE MODEL & TIME COST FOR QUERYING ONE TRAINING BATCH

Name	Estimated Δf	Query Time (sec.)
Bert	81.82	0.577
Transformer-XL	17.09	3.691
XLNet	601.5	0.248
GPT	73.19	0.231
GPT-2	110.2	0.206
RoBERTa	4.15	0.184
XLM	219.4	0.223
Ernie 2.0	28.20	1.777

Privacy Preserving Mapping. We borrow the notion of a Privacy Preserving Mapping (PPM) from [57] to denote a mapping $D_\theta : \mathbb{R}^d \to \mathbb{R}^d$ parameterized by θ, which is trained to minimize the effectiveness of an imagined adversary $\mathcal{A}_\psi$. Meanwhile, the PPM is required to follow the utility constraint: it can only distort the embeddings in a limited radius around the original embedding in order to maintain the utility, or otherwise a trivial yet perfect defense only needs to map all the embeddings to a constant vector. Formally, we propose to learn the privacy preserving mapping D_θ by solving the following minimax game $\min_\theta \max_\psi \frac{1}{n} \sum_{i=1}^n \mathcal{A}_\psi(D_\theta(z_i), s_i)$, s.t. $\|D_\theta(z) - z\|_2 \leq \epsilon$. In other words, the active defense accesses the plaintexts $\{x_i\}_{i=1}^n$, derives the training set $\{(z_i, s_i)\}_{i=1}^n$ for an imagined white-box adversary, and simulates the minimax game as we describe above. As the defense at the user side usually has no access to the intelligent service at the cloud, the utility constraint is formulated as the upper bound on the 2-norm distance between the protected and original embeddings, which is similar to that in [57]. In practice, D_θ is implemented as an encoder-decoder neural network and $\mathcal{A}_\psi$ is implemented as an MLP.

However, we notice an additional challenge brought by the language model setting. Although previous PPM-based defenses have studied several efficient approaches to solve the minimax game when z takes discrete values and D_θ is a combinatorial function [35], [57], our PPM is required to work on the real-valued embeddings with D_θ implemented as neural networks. By the best of our knowledge, there is no effective algorithm to exactly solve the constrained optimization problem above. As an alternative, we propose to reformulate the L2 constraint as a regularization term. In detail, it writes $\min_\theta \max_\psi 1/n \sum_{i=1}^n \mathcal{A}_\psi(D_\theta(z_i), s_i) + \lambda \|D_\theta(z_i) - z_i\|_2$, where the positive coefficient λ is expected to control the privacy level of this active defense. Intuitively, a larger λ corresponds to a stricter utility constraint and thus, a lower privacy level. To solve the unconstrained minimax game, we use the simultaneous Gradient Descent algorithm, which takes alternative gradient descent (ascent) steps on θ and ψ [26].

Subspace Projection. Similar to the methodology in [14], our defense first calculates the *privacy subspace* w.r.t. the semantic meaning we do not want the adversary to distinguish. Specifically in the setting of keyword inference attack, we expect the adversary is unable to distinguish whether a sentence contains certain keyword or not. Therefore, we first collect two sets of embeddings D_1 and D_0 that respectively correspond to sentences with and without the target keyword. Then we compute the privacy subspace as the linear space spanned by the first k orthonormal eigenvectors $u_1, \ldots, u_k$ of the following matrix $C = \sum_{i\in\{0,1\}} \sum_{z\in D_i} (z-\mu_i)(z-\mu_i)^T/|D_i|$ where $\mu_i = \sum_{z\in D_i} z/|D_i|$, i.e. the average sentence embedding in D_i. In our evaluations, we consider the ratio β between the dimension of the privacy subspace k and the full dimension d of the sentence embedding as the parameter of the *subspace defense*, i.e., $\beta = k/d$.

Next, to remove the unwanted semantics from the embedding (denoted as z), we simply project the embedding to the subspace orthogonal to the *privacy subspace*, with the following formulas: $\hat{z} \leftarrow \sum_{i=1}^{k} (I - v_i v_i^T) z, \hat{z} \leftarrow \hat{z}/\|\hat{z}\|$.

Evaluation Details. For PPM defense, we implement the virtual adversary $\mathcal{A}_\psi$ as a 3-layer MLP with 200 sigmoid hidden units and the PPM D_θ as an MLP of the architecture $(d - 200 - d)$ with ReLU activation. We train the virtual adversary and the PPM alternatively for 1 and 5 iterations, where the batch size is set as 64.

D. Fingerprinting Algorithm

We propose the following fingerprinting algorithm to relax Assumption 1, with 100% accuracy on determining the specific model type. First, the adversary determines the candidate model types according to the embedding dimension. For example, if $d = 768$, the candidate set includes GPT, GPT-2 and other three models. Next, the advervsary prepares an arbitrary corpus and queries each language model for the embedddings. Then, the adversary trains an off-the-shelf classifier with the embedding as input and the model type as label. Finally, when the adversary gets a set of embeddings as victims, he/she first uses the language model classifier at hand to determine the model type and conducts the downstream attacks as introduced in previous sections.

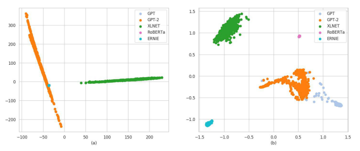

Fig. 9. Clustering phenomenon observed on (a) embeddings from 5 784-dim. language models and (b) their MLP hidden representations of 1000 randomly-sampled sentences on Medical.

TABLE V
INFERRED TOP-10 POSSIBLE KEYWORDS IN EACH SAMPLE FOR REASSEMBLING THE SEMANTICS.

Sample #1. *Destruction of malignant growth (1.1 to 2.0 centimeters) of trunk, arms, or legs*	**arm**, **trunk**, **malignant**, repair, venous, skin, veins, lower, **removal**, artery
Sample #2. *Application of ultraviolet light to skin*	venous, veins, centimeters, **radiation**, older, guidance, arm, removal, arterial, injection
Sample #3. *Removal of malignant growth (1.1 to 2.0 centimeters) of the trunk, arms, or legs*	**arm**, **trunk**, **malignant**, repair, venous, veins, artery, **removal**, tissue, insertion
Sample #4. *Removal of up to and including 15 skin tags*	veins, centimeters, tissue, **removal**, insertion, arms, spinal, arterial, **skin**, legs
Sample #5. *Biopsy of each additional growth of skin and/or tissue*	lower, venous, veins, **biopsy**, **tissue**, insertion, centimeters, artery, endoscope, ultrasound

To evaluate our fingerprinting algorithm, we implement the language model classifier as a (784-200-5) MLP, use the Yelp-Food corpus of 2000 sentences for training and a subset of the Medical corpus that consists of 1000 sentences for testing. Strikingly, we find the classifier achieve 100% accuracy. To better understand the phenomenon, we plot the original embeddings in the test set and their hidden representations at the last layer of the MLP in Fig. 9, where different colors implies different language models. As we can see, the embeddings from different language models distribute in rather divergent ways. After MLP's nonlinear mapping, the embeddings directly collapse to separated clusters according to their corresponding model type. To the best of our knowledge, we are the first to discover and report this interesting phenomenon.

E. Semantic Reassembling with Keyword Inference Attack

Experimental Settings. Following the description in Section VIII, we first select 50 medical-related words to form the candidate keyword set and train an DANN attack model for each keyword with the same configurations in our original work. The DANN attack accuracy is about 77% after being averaged on the 50 words. Then, we randomly select 5 samples from the test set of the medical case and use each DANN model to output the probability of the occurrence of the corresponding keyword. We list the inferred Top-10 keywords of each sentence in Table V.

Results & Analysis. As we can see from Table V, the attacker can actually reassemble the basic semantic meaning of the original text with such a procedure. For example, in Sample #3, when the adversary knows *arm, trunk, malignant* as the Top-3 most possible keywords, then he/she can probably infer the semantic meaning of the original description is related with malignant growth at the arms or trunk. Compared with the plain text of Sample #3, the inferred meaning is quite consistent with the original meaning, despite some minor details left out. Interestingly, we also find, although DANN may predict the occurrence of certain keywords with error, the erroneous prediction may also contribute to the semantic reconstruction. For example, in Sample #2, the adversary fails to predict the occurrence of *ultraviolet* since this word is not in the adversary's candidate set. However, due to the semantic similarity between *ultraviolet* and *radiation* [4], the DANN attack model for *radiation* predicts the high probability

[4]0.942 in cosine similarity of Bert word embeddings.

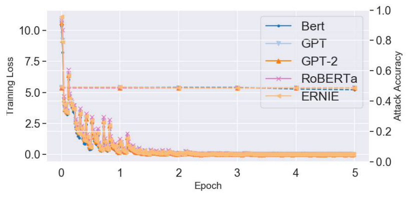

Fig. 10. The training loss of LSTM decoder and attack accuracy of decoder-based attack in the first 5 epochs.

TABLE VI
THROUGHPUT FOR TRAINING ATTACKS IN EACH CASE (ITER/SEC.)

Citizen			Genome	Airline/Medical		
Year	Month	Date		SVM	MLP	DANN
187.9	228.2	213.7	311.2	7209.6	352.3	316.8

of the occurrence of the word radiation, which, despite the inexactness, helps the adversary successfully guess that the hidden semantic is about the radiation-related procedure, i.e., the application of ultraviolet light.

F. *Keyword Inference Attack with Standard Decoder Section*

Experimental Settings. We use the standard decoder module [68], a one-layer bidirectional LSTM to implement the decoder. The vocabulary of the LSTM is initialized the same as the target language model. For most of them (excluding XLNet and ERNIE), the vocabulary is publicly accessible. We therefore implement the decoder-based attack on the rest 6 targets.

For training, we input the decoder both the embedding and the corresponding sentence: the embedding is input as the initial hidden state of the LSTM forwarding phase, while the sentence supervises the generated tokens in a teacher-forcing way [25]. During the evaluation phase, we input the victim embedding as the initial hidden state to the LSTM decoder and decode the tokens greedily. For background on training and evaluating such an LSTM module for generating sentence conditionally, please refer to e.g., [68]. To conduct the keyword inference attack, we suppose the adversary directly tests whether the keyword is contained in the generated sentence. We conduct the decoder-based keyword inference attack on Medical in the white-box setting, with exactly the same configurations of the dataset. Fig. 10 reports the training loss and the keyword inference attack accuracy in the first 5 epochs. We omit the results for Transformer-XL because the decoder cannot be trained on a 11G GPU due to the large vocabulary containing over $220,000$ tokens.

Results & Analysis. As we can see from Fig. 10, for the LSTM decoder on each language model we experiment with, the training loss decreases to almost 0 in the first several epochs. In the console logs, we correspondingly observe that, when the loss is close to 0, the decoded sentences in the training set is almost identical to the original ones. However, when applied to decode from the embedding without teacher-forcing, the decoder is observed to fail to decode any meaningful sentences, always giving a sequence of repetition of certain random word. As a result, none of the decoder-based attacks work out better than a random guesser.

G. *Experimental Environments*

All the experiments were implemented with PyTorch [51], which is an open-source software framework for numeric computation and deep learning. We used the pretrained language models implemented by PaddlePaddle[5] (for Ernie 2.0) and by HuggingFace [74] (for other seven models). We deployed the language models on a Linux server running Ubuntu 16.04, one AMD Ryzen Threadripper 2990WX 32-core processor and 2 NVIDIA GTX RTX2080 GPUs. We conducted our attacks and defenses on a Linux PC running Ubuntu 16.04, one Intel Core i7 processor and 1 NVIDIA GTX 1070 GPU, querying the server via local network. We report the time for quering one mini-batch of training data in Table IV.

H. *Other Omitted Statistics*

TABLE VII
BASIC INFORMATION OF THE THREE VARIANTS OF GPT-2 ARCHITECTURE WE HAVE USED FOR ABLATION STUDIES.

Name	Dimension	# of Parameters
GPT-2	768	1.2×10^8
GPT-2-Medium	1024	3.5×10^8
GPT-2-Large	1280	7.7×10^8

TABLE VIII
STATISTICS OF TEST SAMPLES FOR EACH KEYWORD ON AIRLINE & MEDICAL

Airline	Hong Kong	London	Toronto	Paris	Rome
	808	2656	1320	948	736
	Sydney	Dubai	Bangkok	Singapore	Frankfurt
	1434	802	1260	1264	586
Medical	leg	hand	spine	chest	ankle
	19804	3700	6222	3172	1252
	head	hip	arm	face	shoulder
	4988	2612	18600	3938	2592

I. *Learning Algorithm for DANN Attack*

Algorithm 1 A typical iteration in the learning process of our attack model

```
1: procedure ITERATION($\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}}, \lambda$)
2:     Sample a random mini-batch $B_{\text{k}}$ from $\mathcal{D}_{\text{public}}$
3:     Sample a random mini-batch $B_{\text{d}}$ from $\{(z_i^{'}, 0)\}_{i=1}^{N} \cup \{(z_i, 1)\}_{i=1}^{n_1}$    ▷ $B_{\text{k}}$, $B_{\text{d}}$ are of size $S$.
4:     $\ell_{\text{k}} \leftarrow \frac{1}{S}\sum_{(z,y)\in B_{\text{k}}} \delta(C_{\text{k}.}(E(z)), y)$
5:     $\ell_{\text{d}} \leftarrow \frac{1}{S}\sum_{(z,y^{'})\in B_{\text{d}}} \delta(C_{\text{d}.}(E(z)), y^{'})$    ▷ $\delta$ denotes the cross-entropy loss.
6:     Back-Propagation($\ell_{\text{k}}, \hat{z}$)
7:     Back-Propagation($\ell_{\text{d}}, \hat{z}$)
8:     $\Delta\hat{z} \leftarrow \nabla_{\hat{z}}\ell_{\text{k}} - \lambda\nabla_{\hat{z}}\ell_{\text{k}}$
9:     Back-Propagation($\Delta\hat{z}, z$)
10: end procedure
```

[5]https://github.com/PaddlePaddle/ERNIE