Towards Sentence Level Inference Attack Against Pre-trained Language Models

Kang Gu
Dartmouth College
Hanover, New Hampshire, USA
Kang.Gu.GR@dartmouth.edu

Ehsanul Kabir
Penn State University
University Park, Pennsylvania, USA
ejk5818@psu.edu

Neha Ramsurrun
Dartmouth College
Hanover, New Hampshire, USA
Neha.Ramsurrun.23@dartmouth.edu

Soroush Vosoughi
Dartmouth College
Hanover, New Hampshire, USA
Soroush.Vosougi@dartmouth.edu

Shagufta Mehnaz
Penn State University
University Park, Pennsylvania, USA
smehnaz@psu.edu

ABSTRACT

In recent years, pre-trained language models (e.g., BERT and GPT) have shown the superior capability of textual representation learning, benefiting from their large architectures and massive training corpora. The industry has also quickly embraced language models to develop various downstream NLP applications. For example, Google has already used BERT to improve its search system. The utility of the language embeddings also brings about potential privacy risks. Prior works have revealed that an adversary can either identify whether a keyword exists or gather a set of possible candidates for each word in a sentence embedding. However, these attacks cannot recover coherent sentences which leak high-level semantic information from the original text. To demonstrate that the adversary can go beyond the word-level attack, we present a novel decoder-based attack, which can reconstruct meaningful text from private embeddings after being pre-trained on a public dataset of the same domain. This attack is more challenging than a word-level attack due to the complexity of sentence structures. We comprehensively evaluate our attack in two domains and with different settings to show its superiority over the baseline attacks. Quantitative experimental results show that our attack can identify up to 3.5X of the number of keywords identified by the baseline attacks. Although our method reconstructs high-quality sentences in many cases, it often produces lower-quality sentences as well. We discuss these cases and the limitations of our method in detail.

KEYWORDS

Pre-trained Language Models, Inference Attack, Text Reconstruction

1 INTRODUCTION

Recent years have witnessed many breakthroughs in Natural Language Processing (NLP) domain. After the release of Transformer [58], which is the backbone of pre-trained language models, thousands of language models have been released. So far, the Hugging Face API[1] supports more than 40000 language models, the most significant models of which are BERT [12], GPT [5, 42, 43], XLNet [61], T5 [44], etc. More recently, transformers have also achieved success in computer vision tasks [8, 31]. The modern language models usually rely on very large architectures with millions of parameters and pre-training on massive datasets. Unlike traditional shallow/small neural networks, pre-trained language models are highly generalized and can be employed as a feature extractor for various downstream tasks. For example, BERT obtained state-of-the-art results on eleven NLP tasks at the time of its release [12].

Pre-trained language models have attracted wide attention in many industries [38, 54, 59]. For example, since neural machine translation models are extremely data-hungry, an e-commerce company can rely on the pre-trained mBART [54] to build the multilingual translation system for its customers. Furthermore, medical records are usually stored in relational databases and require specific queries to retrieve information of interest. However, completing such queries quickly can be challenging even for medical experts due to the barriers among subdomains of medicine. BERT can efficiently extract a syntax tree from an electric medical record, which can then be converted to SQL query directly [38]. Although the pre-trained language models have the potentials to be adapted to various downstream NLP applications, their privacy leakage risks are concerning, e.g., Carlini *et al.* [6] showed that generative language models (e.g., GPT-2) can memorize samples in the private training set.

Furthermore, the sentence embeddings of pre-trained language models can capture personal information, which may be inferred by an adversary. Pan *et al.* [37] first proposed a keyword inference attack against private embeddings. Specifically, they trained a binary classifier for each keyword to detect the keyword's existence in the embeddings. However, for this attack to be successful, the adversary must know the distribution of keywords in the dataset so that they can train the classifiers. In addition, the cost of training binary classifiers increases linearly with the number of keywords. More recently, Song *et. al* [52] proposed gradient-based embedding inversion attack, which requires no knowledge of the target dataset. Their attack consists of two steps: 1) first, the observed deep embeddings (e.g., BERT) are mapped to a lower space (input space) by a learned mapping function, and 2) the adversary solves the word distribution at each position of the sentence using gradient

[1] https://huggingface.co/models

Proceedings on Privacy Enhancing Technologies 2023(3), 62–78

https://doi.org/10.56553/popets-2023-0070

descent. However, this method can only recover a set of words without word order, which plays a crucial role in the formation of the sentence [20]. This attack remains at the word level and the problem of reconstructing at the sentence level is still left open.

Our work. To further investigate if the adversary can go beyond the previous attacks, we design a novel attack method with an adversarial decoder, which takes in embeddings as input and attempts to generate the original sentences. In practice, an organization, e.g., a hospital or an e-commerce company, first needs to convert its private text datasets into embeddings via a pre-trained language model to perform various downstream NLP tasks. A third-party service provider with expertise in the downstream NLP tasks will then have access to these embeddings. We aim to investigate whether this third-party access to these embeddings can pose privacy threats to the original private text dataset.

In our attack setting, the adversary (the third-party service provider itself or another entity working with this third-party service provider) leverages an adversarial decoder to reconstruct the texts of the private embeddings and infer the sensitive information. This adversarial decoder is trained by the adversary on a publicly available dataset from the same domain as the private dataset and the embeddings are produced by the same language model. *Note that, in our setting, the adversary does not need to know anything about the target private dataset.* The decoder learns the rules of generating text on public datasets and transfers the rules to the private dataset. Although previous work [37] reported that a regular RNN-based decoder cannot recover useful information from text embeddings, we demonstrate through our experiments that a transformer decoder can reconstruct high-quality sentences in some cases.

Contributions. We first perform a *keyword inference attack* to compare our decoder-based attack with previous attacks [37, 52]. In the original classifier-based attack [37], the adversary aims to exploit sensitive keywords (e.g., disease) in the target dataset. However, unlike this work that constrains the number of keywords to 1010, we take into consideration hundreds of keywords existing in the dataset of interest to simulate real-world scenarios. Next, we propose a novel attack, namely, *sentence inference attack*. In this attack, instead of focusing only on keywords, the adversary aims to reconstruct the entire sentence. In the cases where the adversary does not know the exact keywords in the dataset (this happens when they do not have domain expertise), they can still infer the meaning of the original text by the semantics of the reconstructed text. The contributions of this paper are briefly summarized as follows.

(1) We design a novel inference attack against sentence embeddings that reconstructs the original input text with minimal adversarial knowledge.
(2) We show that a transformer decoder can accurately reconstruct the text from its sentence embeddings in some cases.
(3) We extensively evaluate our decoder-based attack and also compare it with existing inference attacks [37, 52]. The codebase and datasets of this work will be released to enable reproducibility[2].
(4) We also discuss the limitations of our method, including the lower quality of reconstructions in many cases. Please refer to Section 11.2 for details.

[2]https://github.com/KangGu96/Adv_decoder

2 RELATED WORK

2.1 Privacy Attacks against ML

Various privacy attacks are conducted against machine learning applications [30, 46]. Among various forms of attacks, membership inference attack [50, 56, 57] discloses the least information. Shokri et al. [50] introduced the first membership inference attack against machine learning models: *given a trained model and a data record, the adversary can determine if the data record was in the model's training set.* More recently, Shejwalkar *et al.* [49] studied the susceptibility of text classifiers to membership inference, which introduced user-level membership inference that outperformed the existing attacks on both transformer-based and RNN-based models. Besides, attacks on generative models [21, 32] have also been explored.

The model inversion attack was first proposed by Fredrikson *et al.* in statistical models [16] and then generalized to deep neural networks [15]. Unlike membership inference, model inversion attacks aim to *reconstruct partially or fully the private training data that the target model is trained on.* Fredrikson et al. [15] proposed *two* formulations of model inversion attacks. In the first one, the adversary aims to learn a sensitive attribute of an individual whose data are used to train the target model, and whose other attributes are known to the adversary [33].

In the second formulation, the adversary is given access to a classification model and a particular class, and aims to come up with a typical instance for that class [63]. For example, the adversary, when given access to a model that recognizes different individuals' faces, tries to reconstruct an image that is similar to a target individual's actual facial image. Besides, the additional knowledge has been proven to increase the risk of inversion attacks. Zhao *et al.* developed inversion models that can take in model explanations, outperforming the attack methods that use model prediction only [64]. Chen *et al.* presented a novel GAN model that can better distill knowledge, which is useful for performing attacks on private models, from public data [7].

Furthermore, recent studies demonstrated that model inversion attacks could recover texts from the training dataset [6, 37]. In our adversarial setting, the reconstruction of texts relies only on the sentence embeddings generated by the pre-trained language models. We also explore the impact of additional information (pre-training) on inversion attacks.

2.2 Privacy Attacks against Pre-trained Language Models

Pre-trained language models have become a popular component of the current NLP pipeline [41]. However, there are several concerns about their privacy issues. For example, Bguelin *et al.* studied a practical scenario in which users need to continuously update the weights of the language model with modified data [4]. Their results implicated that an adversary can infer specific sentences or fragments of discourse from the difference between the data used to train the model. Furthermore, Nakamura *et al.* showed that an

adversary with some prior knowledge of the patient could employ a pre-trained masked BERT model to predict the masked personal information in the input clinical data [36].

Carlini *et al.* extended model inversion attacks to training data extraction attacks which *aim to reconstruct not just trivial uninformative examples but the verbatim training examples* [6]. They demonstrated that GPT-2 (trained on Internet text) could memorize and generate hundreds of verbatim text sequences in training data given several starting words. The most obviously-sensitive sample contained the full name, physical address, email address, phone number, and fax number of an individual. However, this attack only targeted generative language models but excluded masked language models such as BERT.

Meanwhile, Pan *et al.* first showed keyword inference attack on the embeddings of pre-trained language models [37]. Specifically, an adversary with prior knowledge of the confidential dataset (e.g., clinical note) could infer the sensitive keywords within the sentence embeddings. However, the attack relied on training a binary classifier for each keyword and was tested only in the setting of 10 keywords. Song *et al.* designed a gradient-based inversion attack to predict a set of candidates for each token from the sentence embeddings, without recovering word order [52]. Therefore, the method cannot recover the sentence structure or reveal the semantic information about the sentence.

Although our work also focuses on inference attacks against sentence embeddings, it is different from [52] in two major aspects:

(1) Our method can generate a coherent text sequence that is close to the original sentence, thus revealing more semantic information than a set of unordered words.
(2) Our method is naturally more efficient when there exist a large number of private embeddings. Specifically, the decoder generates text by querying without any gradients involved. In contrast, [52] relies on gradient descent to compute the word distribution for each token in the embeddings.

3 BACKGROUND

3.1 Transformer

Transformer [58] was originally proposed for machine translation task, which later became the backbone of recent language models. Unlike traditional sequential models, transformer adopted self-attention and multi-head attention mechanisms to capture complex sequential dependencies.

Encoder and decoder are the two components of transformer, where encoder consists of six encoding layers and decoder consists of the same number of decoding layers. In the original machine translation task, encoder maps the input in language 'A' to a hidden feature space. Then the decoder projects the hidden states to language 'B'. Our reconstruction attack is similar to the decoding process. In this paper, we employ the capacity of the transformer decoder to reconstruct the original sentence from the sentence embedding, including sensitive keywords.

3.2 Pre-trained Language Models

Language models, which are usually built on transformer architecture, are pre-trained on massive corpus to model the complex text structure. For example, BERT, one of the most popular language models, was trained using BooksCorpus [3] and English Wikipedia [51] with the objective of predicting the masked words and/or the next sentence in a text. Additionally, another significant language model, GPT-2, was trained on 40GB Internet text with the objective of predicting the next word in the text.

Our paper focuses on reconstructing text from sentence embeddings generated by language models. Therefore, our method is agnostic to the model architecture and training objectives.

3.3 Sentence Embedding

Given a vocabulary $\mathcal{V}$ which consists of $|\mathcal{V}|$ tokens, a sentence s is defined as $s = [w_1, w_2, ..., w_n]$, where each word (or token) belongs to the vocabulary $\mathcal{V}$. We define a mapping $\mathcal{F}$ from the sentence to the vector space $\mathbb{R}^{n \times d_w}$ as a sentence embedding function, where n is the number of tokens in the sentence and d_w is the dimension of each token vector.

Although there exist various methods (*word2vec* [34], *doc2vec* [26], etc.) to embed the sentences in NLP domain, $\mathcal{F}$ refers to language models in this paper. Finally, the sentence embedding z of sentence s is obtained by $z = \mathcal{F}(s), z \in \mathbb{R}^{n \times d_w}$.

In some applications, pooling operation is applied to the sentence embedding to produce a 1-dimensional embedding $z \in R^{d_w}$. However, the pooled embeddings are only capable of basic NLP applications (e.g., classification), while inadequate for more advanced applications such as text understanding, entity extraction, and question answering. We consider unpooled embeddings as the main target in our experiments, which are also studied in [52].

3.4 Adversarial Decoder

3.4.1 Architecture. As mentioned earlier, our adversarial decoder inherits the architecture of the transformer decoder. Given a set of sentence embeddings $z = [z_1, z_2, ...z_n]$, the decoder $\mathcal{M}$, and a projection function g, our objective is to reconstruct each token of the original sentence $s = [w_1, w_2, ..., w_n]$ in an autoregressive way:

$$h_1 = \mathcal{M}(z_1) \tag{1}$$

$$w_1 = g(h_1) \tag{2}$$

$$w_k = g(\mathcal{M}(z_k | w_{k-1}, w_{k-2}, ...w_1)) \tag{3}$$

where $h_1 \in \mathbb{R}^{d_h}$ is the first hidden state output of the decoder M, w_k represents the token at k_{th} position, d_h is the dimension of the hidden state, and g is a project function to map the hidden states to the vocabulary. The first token w_1 is only conditioned on z_1, while the following tokens are conditioned on both sentence embeddings and the previously predicted tokens.

3.4.2 Projection. After the output hidden states are obtained from the adversarial decoder, projection will be performed to map the hidden states to a probability distribution of tokens.

Given a dataset $\mathcal{D}$, its vocabulary is defined as $\mathcal{V}_\mathcal{D}$. Thus the sentence generation will be constrained by $\mathcal{V}_\mathcal{D}$. Since a pre-trained language model usually has a large vocabulary (e.g., ~30k for BERT), the unconstrained vocabulary may cause noisy and inaccurate prediction. The hidden state $h_k \in \mathbb{R}^{d_h}$ is projected to probability distribution by:

$$P_k = W_D * h_k \tag{4}$$

where $W_D \in R^{|\mathcal{V}_\mathcal{D}| \times d_h}$ stands for the projection matrix. $|\mathcal{V}_\mathcal{D}|$ is the cardinality of the vocabulary $\mathcal{V}_\mathcal{D}$ and d_h is the size of the hidden state. Therefore, the resulting probability distribution P_k is over $\mathcal{V}_\mathcal{D}$.

3.4.3 Training Objective. The objective of training is simply to optimize the cross-entropy loss between the predicted tokens and ground-truth tokens as below:

$$L(\theta) = -\sum_k P_k * Y_k \tag{5}$$

where P_k is the probability distribution in the k_{th} word and Y_k is the hot encoding of the ground truth word k_{th}.

3.5 Sentence Reconstruction

Finally, sampling strategy will be adopted in order to generate actual words.

3.5.1 Sampling. Since the projection only yields a spectrum of possible tokens, we still need a way to sample the distribution. There exist different sampling methods, the most prominent of which are greedy search [48], beam search [17] and top-k sampling [53]. Although the tokens of interest can be scattered throughout the search space, they have a high likelihood to fall into the list of most possible tokens. In fact, by removing the tail of the distribution, the generation is less likely to go off the topic. Therefore, we employ top-k sampling to avoid repetitive generation and to increase the diversity of generation:

$$C = argsort(\mathcal{P})[:k] \tag{6}$$

$$q_i = \frac{e^{P_{c_i}/t}}{\sum_j e^{P_{c_j}/t}}, \forall c_i \in C \tag{7}$$

$$\mathcal{P}' = [q_1, q_2, ...q_n] \tag{8}$$

The top k indices C in distribution $\mathcal{P}$ are first retrieved and then regularized by the softmax function with temperature t. When t equals to 1, it is the same as the normal softmax. When t is larger than 1, it tends to smooth the distribution. We use high temperature to make the model less confident about the prediction. Therefore, the generated sentence will be more diverse and potentially extract sensitive tokens.

3.5.2 Decaying temperature. As discussed above, we prefer to raise the temperature to smooth the distribution. Since the tokens with the highest probabilities may be non-informative due to their high frequencies, such as "[PAD]", smoothing process will make other informative tokens more likely to be sampled. However, maintaining a high temperature throughout the whole generation process would deviate the generation even when the first few tokens are correct. Thus, we apply a decaying temperature as in [6], which starts at $t = 3$, gradually decaying to $t = 1$ over the first 10 tokens. This makes the model explore more possible "paths" at the beginning while still enabling it to follow a high-confidence path once found.

3.5.3 Maximum Length. We limit the length of all generated text to 15 tokens. As the decoding goes further, the decoder's capacity to accurately predict the words gradually reduces. We have compared 10, 15, and 20, and observed that the first 10 or 15 words were usually relevant and coherent. When extended to 20, the last few words might deviate from the topic and be noisy. The length of 15 is the balance point for preserving coherence and reconstructing more information.

4 GENERAL ATTACK WORKFLOW

The sentence embeddings are used for a wide range of downstream tasks [41]. However, as mentioned earlier, their utility is accompanied by privacy risks.

From classifier-based attack [37] to gradient-based attack [52], it has been shown that the adversary can recover a set of possible words for each token from sentence embeddings. However, each set of words is solved independently, which ignores the strong dependencies between words that belong to the same sentence. There is still a gap between a large group of unordered words and a coherent and well-structured sentence.

To overcome the limitations mentioned above, we propose a generative decoder model to attack the embeddings produced by language models. Due to the nature of the auto-regressive models, each token is conditioned on previous tokens, which makes sure the reconstructed sentences are coherent and meaningful. Furthermore, the training cost of the decoder does not multiply by the number of keywords.

4.1 Attack Definition

We first compare our decoder-based attack with the methods proposed by [37] and [52] on *keyword inference attack*. Then we propose a novel attack class, namely *sentence inference attack*. Compared with *keyword inference attack*, which only focuses on pre-defined keywords, the new attack can still work when the adversary does not know the secrets inside the dataset.

For both attacks, the adversary relies on sentence embeddings to infer sensitive information. Formally, we define a target sentence as s and a publicly available language model as $\mathcal{F}$. Then the sentence embeddings of s are denoted as $z = \mathcal{F}(s)$, $z \in \mathbb{R}^{n \times d_w}$, where n is the number of tokens in s and d_w is the dimension of vector. Then sentence embeddings z is mapped back to tokens by attack model: $s' = \mathcal{A}(z)$. For example, here is a real pair of original and reconstructed clinical notes "abdominal ultrasound of a single pregnant uterus or first fetus" and "abdominal ultrasound of pregnant pregnancy first". Although the generated text is not a verbatim copy of the original text, it still maintains the semantics and reveals sensitive information such as "abdominal" and "pregnant".

4.2 Threat Model

Our threat model is the same as the previous work [37]. The threat model is defined as below:

(1) The adversary has access to the language model as a black-box, which takes a sentence as input and outputs a sequence of embeddings.

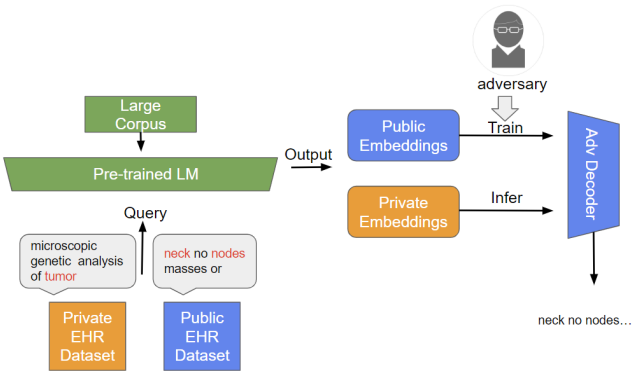


Figure 1: The general workflow of our attack. EHR stands for Electronic Health Record. The adversary first queries the pre-trained language model to obtain the public sentence embeddings. Then the adversarial decoder is trained using public text/embedding pairs. Finally, the trained decoder is employed to infer the sensitive information from private embeddings.

(2) The adversary has access to a set of embeddings of a private dataset, but they do not know the exact sensitive information to infer.
(3) The adversary has access to a public dataset that belongs to the same domain as the targeted private dataset, but the public dataset is not guaranteed to share the same distribution with the private dataset. This is a valid setting since the private dataset is unknown.

We make different assumptions about the distribution of the private/public datasets. The detailed description can be found in Section 8.

4.3 Attack Pipeline

Our general attack workflow is shown in Fig. 1. In our setting, there exists a publicly accessible electronic health record (EHR) dataset, as well as a privately owned EHR dataset. The pre-trained language model is accessible as an oracle.

The adversary can infer the sensitive information in a private dataset by following the four steps given below:

(1) The adversary queries the pre-trained language model to obtain the sentence embeddings of the public dataset.
(2) The adversary then trains the adversarial decoder using the pairs of public dataset text and sentence embeddings obtained in the previous step.
(3) The adversary has access to the embeddings of private dataset provided by a third-party organization.
(4) Finally, the adversary employs the trained decoder to reconstruct the private dataset text from the sentence embeddings.

5 KEYWORD INFERENCE ATTACK

In this section, we assume that the sentence can be in an arbitrary format and the adversary knows the keywords or the rule of defining keywords in the target dataset. Then we compare our methods with baselines on the capacity of identifying keywords.

5.1 Attack Definition

The adversary in the keyword inference attack attempts to infer all keywords in an unknown text. Keywords are defined by a well-known rule or expertise in the domain. Therefore, keywords can be highly sensitive and an attack can be a serious threat to real-world systems (e.g., medical & airline domains).

Formally speaking, we define the rule of keywords as $\mathcal{K}$. Given a sentence s, its sentence embedding is represented by z. $\mathcal{A}$ represents our general attack model, which includes the pre-defined decoder $\mathcal{M}$. The adversary wants to find out the relationship of $\mathcal{A}(s) \leftarrow \mathcal{K}(s)$. Our attack model $\mathcal{A}$ does not require knowing $\mathcal{K}$ for training, thus we only utilize the rule $\mathcal{K}$ at the test stage.

The workflow of keyword inference attack is displayed in Fig. 2. Note that the process of our decoder generating the text is stochastic. We repeat the generation 10 times[3] to extract diverse outputs. Then we convert the output text into a list of words $\mathcal{L}$ after filtering stopwords and sorting by frequency. We only keep top k words in the list to reduce the number of irrelevant words. The impact of k, which is the number of words kept, is further studied in Section A.4. We slightly constrain the attack definition here to measure the relationship of $\mathcal{L} \leftarrow \mathcal{K}(s)$.

5.2 Attack Settings

White-box Attack. In this attack context, we focus on the situation where the public dataset and the private dataset share the same distribution. Therefore, the adversary can safely guess the keywords in the private dataset by just examining the public dataset. We show that in the white-box setting, both our decoder-based attack and the baseline attacks can threat the privacy of pre-trained language models.

Black-box Attack. In contrast to the white-box setting, the adversary in the black-box setting has minimal knowledge of the private dataset, which means that the public dataset and private dataset may have different distributions. We use two different datasets in the same domain to mimic this setting. Note that the black-box setting challenges the transferability of the attacks. It is more realistic that the adversary does not know much information about the private dataset due to privacy protocols. An alternative for the adversary is thus to study a public dataset within the same domain and transfer the knowledge to the private dataset.

6 SENTENCE INFERENCE ATTACK

In *keyword inference attack*, we have assumed that the adversary has access to a public dataset, and can learn and target a set of keywords. In the scenario where they do not have targets but still try to infer from the embeddings, we propose a novel attack, namely, *sentence inference attack*, which aims to reconstruct the original text verbatim.

[3] 10 is selected to balance between the performance and efficiency, other choices are also acceptable.

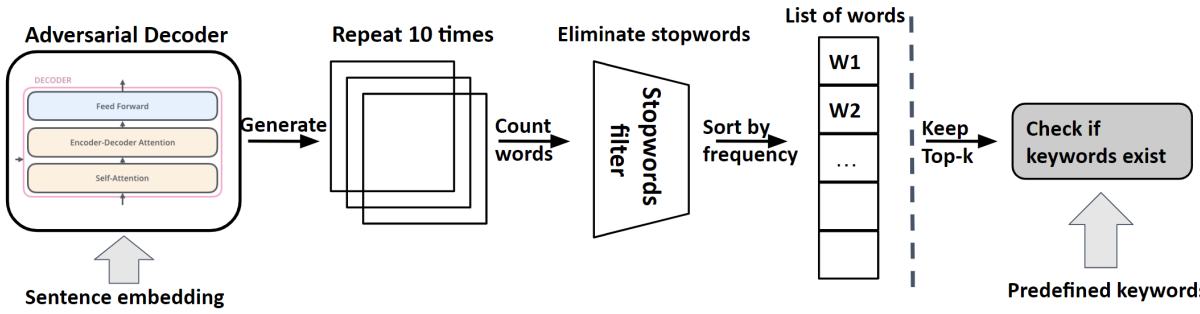


Figure 2: The workflow of keyword inference attack. The adversary first repeats the decoding of the same sentence embedding 10 times to gather all potential outputs. Then the outputs are passed through a stopwords filter to eliminate stopwords. The adversary further converts the sentences into a list of words sorted by frequency. Finally, only top-k words are kept and examined. Note that k is a hyperparameter in our experiments.

6.1 Attack Definition

The adversary does not know the sensitive information in the private dataset. Therefore, they can only employ our decoder-based attack to infer from the private sentence embeddings in a generative way.

Similarly, to recover a sentence s from private dataset using its sentence embeddings z generated by a language model and an attack model $\mathcal{A}$, the adversary can solve the below equation to maximize the similarity between s and reconstructed sentence $s' = \mathcal{A}(z)$:

$$s^* = argmax(\bigcup_{1 \le i \le r} \Phi(s, s'_i)) \tag{9}$$

where $[s'_1, s'_2, ..s'_r]$ are a set of reconstructed sentences, $\Phi()$ is a similarity function, and s^* is the most similar candidate. Note that the decoder generation process is stochastic, therefore, the generation is repeated r times to capture different cases.

6.2 Attack Settings

We generally follow the settings in *keyword inference attack* to conduct experiments. Both the white-box attack and black-box attack reuse the previous settings. Nevertheless, instead of counting the words in reconstructed sentences, we directly measure the similarity between the original text and the reconstructed text. Note that we repeat generation 10 times for each sample, therefore, we select the one that maximizes the similarity function $\Phi()$ as the best candidate.

7 DATASETS

In this section, we introduce real-world datasets from two domains: airline and medical. We show that the NLP systems of these two domains are threatened by privacy attacks.

7.1 Airline

With the growing competition in the airline industry, airline companies need to constantly improve their service quality to survive the competition [18]. Online reviews are a popular way for customers to share their experiences with flights. With the aid of pre-trained language models, airline companies can build automatic tools to analyze customers' opinions (e.g., topic modeling and sentiment analysis [25]). However, as discussed in [37], an adversary can infer various sensitive information from text embeddings, including, but not limited to location, flight code, and departure/arrival time.

Skytrax: This dataset[4] contains airline reviews from 2006 to 2019 for popular airlines around the world. We extract a subset of about 30k reviews from this dataset for evaluation by filtering out empty or non-English reviews. Without performing downsampling, we simply extract the first sentence of each review and form a new dataset. This is because we observe that the first sentences are more relevant to our target of interest (location, time, etc.).

Twitter US Airline: This dataset[5] is originally collected for sentiment analysis, which contains 14614 tweets to the accounts of US airline companies. We clean and extract a subset of around 5k tweets from the original dataset. The preserved subset contains 20 US city names. Note that the tweets may include a mixture of full names of these cities and their acronyms (e.g., Athens vs ATH), which makes it a challenging dataset for performing privacy attacks.

7.2 Medical

In recent years, AI-powered applications have been increasingly applied to clinical tasks [22]. Specifically, various NLP methods have been proposed for the extraction of clinical pathways [60], recognition of biomedical entities [27], patient questions answered [11], etc. Although the power of language models can benefit patients, an adversary can also capitalize on the text embeddings of medical transcriptions to infer personal health information (e.g., precise disease sites).

CMS: This dataset[6] is from the Center for Medicare and Medicaid Services website, which records information on services and procedures documented by physicians and other healthcare professionals. In total, there are 5569 unique samples. We use all of these samples for our experiments.

MT: This dataset[7] contains sample medical transcriptions for various medical specialties, including surgery, consult, and more. There

[4] https://github.com/quankiquanki/skytrax-reviews-dataset
[5] https://github.com/benhamner/crowdflower-airline-twitter-sentiment
[6] https://data.cms.gov/provider-summary-by-type-of-service/medicare-physician-other-practitioners/medicare-physician-other-practitioners-by-provider-and-service
[7] https://www.kaggle.com/datasets/tboyle10/medicaltranscriptions

are 5k medical transcriptions in total. We first split each transcription into sentences and then count the medical keywords provided by the dataset. We then only keep 12018 sentences with the 100 most frequent keywords as a subset so that the size of MT dataset is close to that of CMS dataset.

7.3 Data Pre-processing

To perform *keyword inference attack*, we need to identify all the keywords within each dataset. We introduce our method for labelling in this section.

Airline: For Skytrax dataset, we refer to the World Cities dataset[8], which only lists cities above 15,000 inhabitants. For each airline review, we check which cities exist in the text and keep a list of existing cities. As for Twitter US Airline dataset, we simply label the tweets with the 20 US cities in the same way. After manual examination, there are 962 cities on Skytrax and 20 US cities on Twitter.

Medical: For both medical datasets, we rely on named entity recognition (NER) model pre-trained on *Spacy* 'en_ner_bionlp13cg_md' corpus[9]. We simply apply the NER system to identify biological terms within each sample as our medical keywords. After manual review, there exist 1195 keywords and 2377 keywords in CMS and MT datasets, respectively.

8 EVALUATION METRICS

8.1 Keyword Inference Attack Metrics

To compare our decoder-based attack with the baseline attacks [37, 52], we first introduce the notion of Reconstruction and slightly extend it for our method.

Definition 1. If a string t exists both in the original sentence s and in the sentence s' generated by the adversarial decoder $\mathcal{M}$, then the string t is successfully reconstructed by the decoder $\mathcal{M}$.

Intuitively, only string t existing in both s' and s is considered valid. Even if t is sensitive, it is still false positive if $t \notin s$. Reconstruction of the decoder is conditioned on the sentence embedding z, denoted as $\mathcal{M}(z|s')$. Since there are hundreds of target keywords in our datasets, simply measuring the attack results with overall accuracy or the F-1 score does not accurately reflect the performance of an individual keyword. In addition, showing the detailed attack results on each individual text sample is informative, but not efficient.

As a result, the strength of the attack is measured by how many keywords the adversary can extract in total and how many unique keywords it can extract. Combining these two metrics, we can better measure the effectiveness and generalizability of the attack.

Definition 2. Given a dataset $\mathcal{D}$ made up of sentences $[s_1, s_2, ..., s_n]$, let the adversarial decoder $\mathcal{M}$ reconstruct a set of strings. $t_i = \cup t_{ij}$ from each sentence s_i. Thus, at the level of the dataset $\mathcal{D}$, all strings reconstructed by the decoder $\mathcal{M}$ can be denoted as:

$$\mathcal{T} = \bigcup_{1 \le i \le n} t_i, t_i \in s_i \tag{10}$$

[8]https://github.com/datasets/world-cities
[9]https://allenai.github.io/scispacy/

Our two metrics are defined on the basis of $\mathcal{T}$. If we slightly constrain the type of strings in Definition 1 to be pre-defined keywords, $\mathcal{T}$ will become the union of all the reconstructed keywords. The first metric, **the count of reconstructed keywords**, can be denoted as $|\mathcal{T}|$, where $|\cdot|$ represents cardinality. Also, the second metric, **the number of unique keywords**, is then formulated as $|\{\mathcal{T}\}|$, where $\{\}$ is the set notation.

Moreover, the proposed metrics also generalize well to classifier-based attack [37]. The union of reconstructed keywords, $\mathcal{T}$, can be calculated by examining the true positive predictions made by the classifier. As for the gradient-based attack [52], we apply the sampling strategy elaborated in Section 3.5.1 to the final word distributions to obtain actual sentences, followed by sorting and snapping by k. Therefore, the proposed metrics can be used to evaluate it as we evaluate our decoder-based attack.

8.2 Sentence Inference Attack Metrics

The *sentence inference attack* is evaluated by the similarity function $\Phi()$. There are various similarity functions, e.g., Manhattan distance, Euclidean distance, cosine similarity, etc. We select cosine similarity as our metric because of its ability to measure the degree to which two sentences overlap.

Definition 3. Given two vectorized sentences a and b, the **cosine similarity** $cos(a, b)$ is defined as:

$$cos(a, b) = \frac{\vec{a} \cdot \vec{b}}{\|a\| \cdot \|b\|} \tag{11}$$

Formally, given two sentences s and z, their joint set of tokens are $\{s\} \cup \{z\}$. Then the vectorized version of s can be obtained by one hot encoding: 1) initialize an all zero vector a with length of $\{s\} \cup \{z\}$. 2) check if i_{th} element in the joint set exists in s. 3) assign 1 to the i_{th} element in a if the condition of last step is met. For example, given two toy sentences "I love rose" and "I love lily", the joint set of tokens will be {"I", "love", "rose", "lily"}. The vectorized sentences should be represented as $[1, 1, 1, 0]$ and $[1, 1, 0, 1]$, respectively.

Besides measuring the overlap between two sets of words, we also consider BLEU and ROUGE [39] as additional metrics since they further measure the overlap between n-grams. Although BLEU uses high order n-gram (n>1) matches, it does not consider sentence level structure [29]. E.g., given a pair of original/reconstructed sentences: "Paris (cdg) to Detroit (dtw)" vs "Paris [PAD] to [PAD] dtw", the BLEU score is close to 0. However, the adversary can figure out from the sentence structure that the subject flew from Paris to dtw. Hence, we also use word order similarity (WOS) metric [28] as it better captures the evaluation of sentence structure. The WOS value for the above example is 0.52, suggesting that the reconstructed sentence preserves the original sentence's structure.

Definition 4. Let $T = [w_1, w_2, w_3, ...]$ be the ground truth sentence and S be the reconstructed sentence with the same length. T is vectorized by mapping function $f : w_i \Rightarrow i$, where w_i is the word at index i in T. As a result, the vector T' is simply $[1, 2, 3, ...]$. S is vectorized by searching for w_i in S. Suppose w_i appears at index j in S, i will be assigned to index j in S'. If w_i is not found in s, the most similar word will be matched with w_i. The **word order similarity** is computed by:

$$WOS(T,S) = 1 - \frac{||T' - S'||}{||T' + S'||} \quad (12)$$

A simple example is T = 'A dog jumps over the fox'" and S = "A fox jumps over the dog". The vectorized version will be $T' = [1, 2, 3, 4, 5, 6]$ and $S' = [1, 6, 3, 4, 5, 2]$. Finally, the order similarity is computed as 0.9.

9 EXPERIMENTAL EVALUATION

9.1 White-box Setup

Since the training set and the test set from the same dataset share the same distribution, We split the datasets into training/test sets to mimic the white box setting. The two benchmark systems that we aim to attack are described below:

- Airline-Skytrax: Suppose an airline company employs the pre-trained language models to analyze their reviews in order to improve the service quality. We use Skytrax dataset to mimic the dataset employed by the company. Our goal is to infer the keywords from the sentence embeddings. We split the dataset into 80%-20% to obtain training (public) and test (private) datasets, respectively. All attack models are trained on the same training set and tested on the same test set.
- Medical-CMS: Likewise, a hospital builds a prediagnosis system to guide the patients to the right departments according to the textual descriptions of their medical conditions. Suppose the CMS dataset is used in their system. We again split the dataset into 80%-20% to get training and test datasets, respectively. All the attack models are developed on the same train/test sets for fair comparison.

9.2 Black-box Setup

Following the setup in Section 4, we already have pre-trained attack models in airline and medical domain. Our goal in this black-box setting is to evaluate their performance on unknown datasets. Different from white-box setting, training set and test set now are from two different datasets in the same domain.

Therefore, the weights of all the pre-trained models are frozen at this point. As for the gradient-based attack, the mapping module (from deep embedding to lower space) is frozen.

(1) Airline-Twitter: In the airline domain, we let Skytrax be the public dataset and Twitter be the private dataset. The adversary attempts to attack this new unknown private airline system with the pre-trained model that is trained on a known airline system.
(2) Medical-MT: In the medical domain, the CMS dataset is treated as a public dataset and the MT dataset as a private one. The goal of the adversary is to infer the unknown private dataset with a pre-trained attack model.

9.3 Implementation

Baselines: Note that we discussed that training a binary classifier for each keyword is not practical in our setting, where there exist hundreds of keywords. According to [37], the adversary needs to build a balanced dataset for each classifier, which brings tremendous cost due to the keywords we have. As a result, we extend the original binary classifier to a multi-class classifier without modifying their methodology.

- Decision Tree (DT): the feature selection criterion is set as *gini*. The two most widely used criterions are *gini* and *information gain* and [45] shows that their performance is quite similar and the criterions differ in only 2% of the cases. We pick *gini* as it is computationally less intensive. The rest of the parameters follow the default setting in *scikit-learn*[10].
- K-Nearest Neighbor (KNN): the "n_neighbors" is set to 5, weight function is set at default *uniform* and the optimizing algorithm is set as *auto*. Both DT and KNN are implemented using *scikit-learn*.
- Deep Neural Network (DNN): The DNN has two fully connected layers with 250 and 100 hidden units, respectively. The objective is to minimize the cross-entropy loss. Besides, Adam optimizer with batch size of 100 and learning rate of $1e-4$ is employed. Finally, the maximum train epochs are set as 250. DNN is implemented using *pytorch* [11].
- Gradient-based Embedding Inversion (GEI): At first, a two-layer MLP is trained to map the deep embeddings to lower space. Then we use gradient descent to solve the problem $\min(W^T \cdot p - M(z))$. Where W is the word embedding matrix, M is the mapping model, z is the embedding and p is the solution. More details are provided in [52].

Adversarial Decoder: Our adversarial decoder is trained to reconstruct the text from sentence embeddings. The training process consists of two steps:

(1) Let the public dataset be $\mathcal{D}_{public}$, and the split train/test set be $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$, respectively. We first pre-train the decoder on $\mathcal{D}_{train}$ to obtain a generalized model. However, the model $\mathcal{M}$ at this stage is not accurate enough to predict the text. The goal is to let the decoder learn the distribution of $\mathcal{D}_{train}$.
(2) After the pre-training, we further fine-tune the decoder $\mathcal{M}$ on a subset $\mathcal{D}_{sub}$ of $\mathcal{D}_{train}$. Specifically, $\mathcal{D}_{sub}$ is extracted only by keeping the samples with keywords in $\mathcal{D}_{train}$. This fine-tuning step is essential for the decoder to focus on keywords.

Moreover, the adversarial decoder inherits from the transformer decoder architecture, which consists of 6 decoding layers and 8 heads. The training objective is minimizing cross-entropy loss and the optimizer is AdamW. The decoder $\mathcal{M}$ is first pre-trained on public dataset for 100 epochs with a learning rate of $1e-4$ and then fine-tuned on the subset of the public dataset with a learning rate of $1e-5$.

We propose two types of decoder: 1) vanilla decoder and 2) pre-trained decoder. The former is the model without pre-training process (only trained on $\mathcal{D}_{sub}$). The latter is the model pre-trained in $\mathcal{D}_{train}$ and fine-tuned in $\mathcal{D}_{sub}$.

Embedding Dimension The input dimension of DT, KNN and DNN is 768, which is resulted by pooling operation on the original sentence embedding $z \in R^{128\times768}$. While the input dimension of GEI and decoder is $R^{15\times768}$, since only the first 15 tokens are targeted.

[10] https://scikit-learn.org/stable/
[11] https://pytorch.org/

9.4 Keyword Inference Results & Discussion

The results of white-box and black-box attacks are listed in Table 1. *Vanilla Decoder* and *Pre-trained Decoder* refer to our proposed attack and the rest models are baseline attacks. Note that 'Count' stands for the count of all reconstructed keywords and 'Unique' stands for the number of unique keywords. Furthermore, k, the number of kept words in our attack, is set $k = 20$. The ablation study of k is shown in Section A.4. $k = 20$ is a relatively low threshold (a lower k makes the attack more efficient) since the vocabulary of every dataset in our experiments contains thousands of tokens.

Note that 20 is still smaller than 1% of the size of the vocabulary.

Effectiveness of Attacks: Even when k is set as 20, the experiment results still highlight the effectiveness of our attack over the classifier-based attacks and GEI in both white-box and black-box settings. For example, in Table 1, our Pre-trained Decoder in white-box setting outperforms all the baselines on two metrics. The classifier-based baseline attacks achieve comparable results to our attacks on Skytrax dataset, which suggests that classifier-based attacks are effective in white-box setting where the private dataset and public dataset share the same distribution. Besides, DNN outperforms DT and KNN consistently.

Our Pre-trained Decoder exceeds baselines distinctively on CMS dataset. Specifically, our decoder can identify 1093 keywords in total and 203 unique keywords given the BERT's embeddings of the private medical transcriptions while DNN can only correctly predict 531 keywords in total and only 94 of them are unique. The performance of DNN is merely about 50% of our decoder's. The experimental results imply that our decoder demonstrates better generalizability over different domains and embeddings.

The visualization of attack results in Table 1 are displayed in Figure 3. For each dataset, we select the top 10 most frequent keywords. Specifically, the 10 cities in Skytrax dataset are London, Paris, Bangkok, Toronto, Sydney, Hong Kong, Manchester, Dubai, Melbourne, and Singapore. As for the CMS dataset, the 10 medical terms are tissue, spinal, muscle, skin, bladder, heart, bone, blood, brain, and eye. Our Pre-trained Decoder outperforms the baselines with a distinctive margin on many keywords, which also supports that our decoder generalizes better.

Comparison between Attack Settings: It is noticeable that the numbers of keywords inferred in the black-box setting are much lower than the counterpart of white-box setting across all the attacks. As we have discussed, the black-box setting is more realistic where the adversary does not have information about the private dataset. Even if the adversary knows the domain of the dataset, it can still be challenging to define the keywords within the dataset. For instance, given a medical dataset, there may exist thousands of keywords (e.g., 1195 in CMS and 2377 in MT), which makes it extremely difficult to include all of them.

Therefore, the black-box setting is a more challenging setting, which understandably leads to poorer experimental results for all the attacks under experiment. Nevertheless, our Pre-trained Decoder remains the most robust attack method, especially on the Twitter dataset and GPT-2 embedding. The results suggest that our attack handles this setting better due to its flexibility and generalizability.

Transferability of Attacks: According to Table 1, the Pre-trained Decoder displays much better transferability in all the cases. To be specific, the gap between Pre-trained Decoder and DNN is further enlarged in both domains compared with results of Table 1.

For example, all baseline attacks completely fail on the Twitter dataset with GPT-2 embeddings. However, our decoder still memorizes 30 city names in total, and there exist 5 unique city names. When it comes to the MT dataset, our decoder memorizes more than double the total medical terms/unique medical terms that captured by DNN. Based on the above observations, we can safely conclude that our decoder continues to behave more robustly in the black-box setting. Its better transferability makes it a more powerful threat to the real-world systems.

The Impact of Pre-training Pre-training has boosted the performance of the decoder in all cases significantly. Specifically, pre-trained decoder has identified at least at least 30% more keywords in total than the vanilla decoder. The gap of the number of unique keywords is distinctive too. The results imply that the pre-training is a robust way to strengthen both the generalizabilty and transferbility of the decoder.

9.5 Sentence Inference Results & Discussion

Note that Table 2 displays the results for sentence inference attack. "cosine" stands for cosine similarity and "order" stands for word order similarity. Due to the limitation of the classifier-based attacks, they are removed from this attack setting.

Quantitative Results

As shown in Table 2, Our Pre-trained Decoder achieves significant improvements of not only cosine similarity but also word order similarity over GEI on both white-box and black-box settings.

One of the key observations is that GEI is biased on the empty tokens such as "[PAD]", which yields final sentences with large proportion of noises. This phenomenon is caused by the contextual learning in lanugage models. The embeddings of pre-trained language models are highly convoluted due to self attention mechanisms [58]. The same words in different contexts will be transformed into different deep embeddings. Therefore, the shallow mapping function, which maps deep embeddings to lower space, might be biased with embedding variance of each word, which leads to biased recovered sentences.

Compared with the mapping function, our decoder utilizes dependencies in sentences to invert deep embeddings rather than solve each word independently. As a result, our method is capable of recovering much more coherent and informative sentences from the embeddings, therefore capture semantic information.

Comparison between Attack Settings We can observe that the cosine similarity scores drop from white-box setting to black-box setting. For example, the mean cosine similarity drops from 0.30 to 0.17 on CMS and GPT-2 embeddings.

Similar to the observation in keyword inference attack, there may exist unseen sentence structures and patterns in the black-box setting , which challenges the flexibility of the attacks. Our Pre-trained Decoder still remains relatively robust in black-box setting, indicating it is a more practical privacy threat.

Comparison between Metrics In addition, the mean and std of cosine similarity in various configurations do not necessarily agree

Table 1: Keyword Inference Results ($k = 20$)

Target Dataset	Attack	White-box				Black-Box			
		BERT		GPT-2		BERT		GPT-2	
		Count	Unique	Count	Unique	Count	Unique	Count	Unique
Skytrax	DT	112	52	218	75	68	11	101	14
	KNN	266	92	219	35	80	11	73	6
	DNN	627	124	1120	173	174	12	430	16
	GEI	128	49	159	51	74	11	132	12
	Vanilla Decoder	421	63	679	84	192	32	391	28
	Pre-trained Decoder*	**835**	**158**	**1190**	**179**	**322**	**45**	**513**	**48**
Twitter	DT	51	13	76	15	13	4	0	0
	KNN	92	15	67	12	27	6	0	0
	DNN	187	16	331	17	58	5	0	0
	GEI	87	13	91	14	12	4	0	0
	Vanilla Decoder	203	15	215	15	23	3	9	2
	Pre-trained Decoder	**326**	**20**	**387**	**19**	**87**	**9**	**30**	**5**
CMS	DT	206	49	208	51	127	20	73	13
	KNN	446	104	195	54	361	23	117	12
	DNN	531	94	488	93	380	31	511	32
	GEI	221	50	201	47	117	32	109	30
	Vanilla Decoder	602	98	374	59	431	67	389	60
	Pre-trained Decoder	**1093**	**203**	**814**	**163**	**647**	**135**	**592**	**114**
MT	DT	465	107	325	92	38	13	44	10
	KNN	519	113	546	127	148	23	19	12
	DNN	967	109	1280	119	293	36	136	27
	GEI	527	95	603	99	51	16	59	15
	Vanilla Decoder	842	136	791	128	351	70	213	34
	Pre-trained Decoder	**1250**	**157**	**1463**	**201**	**694**	**104**	**484**	**88**

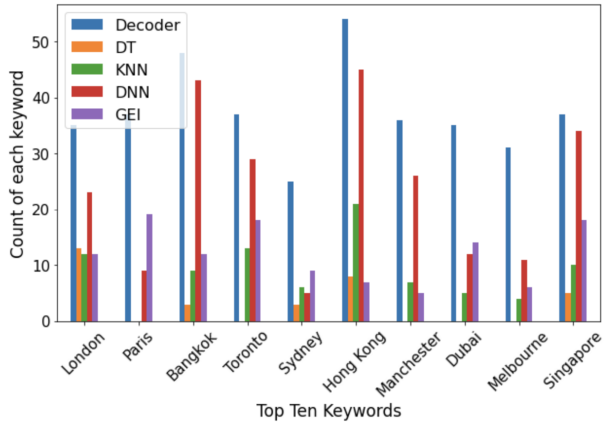
Table 2: Sentence Inference Results. Results on GPT are shown in ().

Target Dataset	Attack	White-box				Black-Box			
		BERT(GPT)				BERT(GPT)			
		Cosine	Order	BLEU	ROUGE	Cosine	Order	BLEU	ROUGE
Skytrax	GEI	.07(.06)	.10(.10)	.01(.01)	.01(.01)	.01(.02)	.02(.04)	.00(.00)	.00(.00)
	Vanilla Decoder	.10(.12)	.33(.35)	.05(.05)	.06(.07)	.07(.05)	.15(.13)	.02(.02)	.03(.02)
	Pre-trained Decoder*	**.25(.21)**	**.52(.51)**	**.13(.11)**	**.12(.12)**	**.18(.17)**	**.40(.37)**	**.10(.10)**	**.11(.10)**
Twitter	GEI	.05(.06)	.11(.10)	.01(.01)	.01(.01)	.03(.02)	.05(.04)	.00(.00)	.00(.00)
	Vanilla Decoder	.09(.08)	.21(.18)	.04(.03)	.05(.05)	.05(.03)	.09(.06)	.02(.01)	.02(.01)
	Pre-trained Decoder	**.22(.20)**	**.50(.45)**	**.11(.10)**	**.13(.12)**	**.15(.14)**	**.39(.35)**	**.08(.07)**	**.10(.10)**
CMS	GEI	.12(.10)	.20(.18)	.02(.01)	.03(.03)	.05(.03)	.06(.05)	.00(.00)	.01(.00)
	Vanilla Decoder	.20(.17)	.35(.31)	.10(.09)	.12(.11)	.11(.10)	.30(.26)	.04(.03)	.05(.05)
	Pre-trained Decoder	**.36(30)**	**.59(.53)**	**.16(.14)**	**.19(.15)**	**.22(.19)**	**.41(.38)**	**.11(.09)**	**.11(.09)**
MT	GEI	.12(.11)	.23(.20)	.01(.01)	.01(.01)	.05(.05)	.08(.10)	.01(.00)	.01(.00)
	Vanilla Decoder	.23(.25)	.46(.49)	.14(.16)	.15(.16)	.10(.11)	.19(.24)	.06(.07)	.05(.05)
	Pre-trained Decoder	**.38(.42)**	**.59(.61)**	**.20(.22)**	**.21(.22)**	**.19(.17)**	**.45(.44)**	**.12(13)**	**.11(.12)**

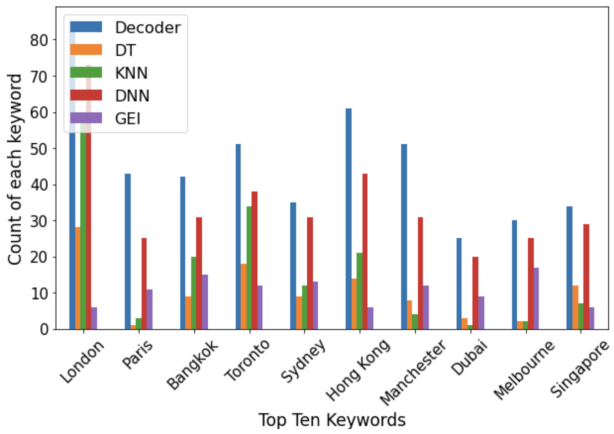
with the metrics of *keyword inference attack* according to Table 1. For example, the mean cosine similarity and the count of memorized keywords are 0.25 and 835, respectively, on Skytrax and BERT embeddings. Although the mean cosine similarity is 0.21 in Skytrax and GPT embeddings, the actual count of keywords is 1190, which is higher than 835. The reason behind this situation is that the cosine similarity measures the degree to which two sentences overlap, therefore, it does not focus on any keywords. Higher similarity means a higher number of words in the reconstructed sentence also exist in the original sentence.

Qualitative Results

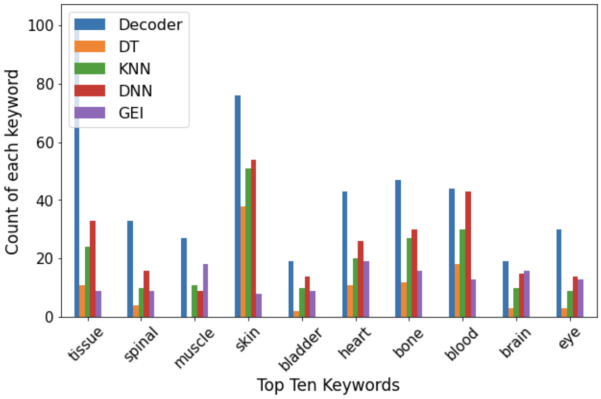
To demonstrate the capacity of *sentence inference attack*, 10 reconstructed sentences are displayed in Table 3. For instance, the reconstructed sentence "Vaccine pneumonia influenza virus nasal"



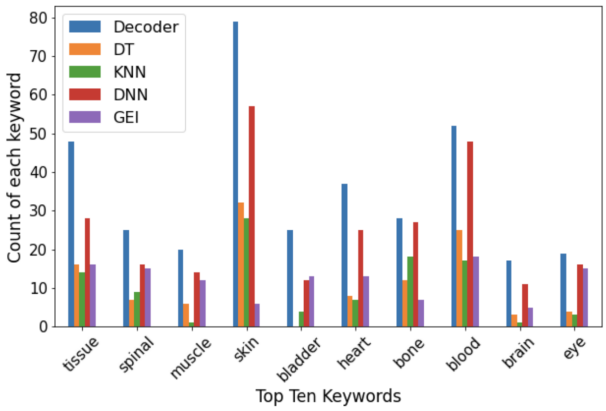
(a) Results of top 10 most frequent cities in Skytrax dataset with BERT embeddings



(b) Results of top 10 most frequent cities in Skytrax dataset with GPT-2 embeddings



(c) Results of top 10 most frequent terms in CMS dataset with BERT embeddings



(d) Results of top 10 most frequent terms in CMS dataset with GPT-2 embeddings

Figure 3: The barplots of the results in Table 1. For each dataset, we show the top 10 most frequent keywords.

Table 3: 10 Examples of Sentence Inference Attack

Domain	Original	Reconstructed (cosine>0.5)
Airline	Bucharest to amsterdam via Prague	**Bucharest** express **via Prague**
	Satisfactory Flight from Singapore to Hong Kong	**Satisfactory Flight to Hong Kong**
	30 January Dusseldorf to Leeds Bradford	**30 January** Manchester **to Leeds Bradford**
	Luxembourg to London city return	**Luxembourg to** Brussels **city return**
	Rome to Toronto July 2013	**Rome to Toronto 2013**
Medical	Closed treatment of broken heel bone	**Closed treatment bone** ankle
	Injection of bladder and urinary duct (ureter) for X-ray imaging	**Injection of bladder kidney** renal **urinary duct**
	Closed treatment of fracture and/or dislocation of pelvis and/or sacrum	**Closed treatment of** suspension **fracture and dislocation**
	Transplantation of donor kidney	**Transplant donor' kidney**
	Vaccine for influenza for nasal administration	**Vaccine** pneumonia **influenza** virus **nasal**

can allow the adversary to accurately infer that this sample is "vaccine for pneumonia/influenza for nasal". Another pair of examples is "Rome to Toronto July 2013" vs. "Rome to Toronto 2013". The decoder has captured most of the information accurately except the month. Noticeably, the pair of "Transplantation of donor kidney" versus "Transplant donor' kidney" shows that the decoder has learned to use contraction during training, which implies it captures the underlying language patterns within the pretraining dataset. Therefore, *sentence inference attack* can threaten the NLP

systems without any knowledge about the private dataset. The adversary can infer the semantics of the original sentence given a similar reconstructed sentence. In addition to the reconstructed sentences in Table 3, we report a few randomly picked examples as shown in Table 4. We observe lower coherence and fluency in these examples. This is a limitation of our attack method and we further discuss this in Section 11. However, note that, such reconstructions may still leak information through their structures that are similar to the original sentences. E.g., the reconstructed "manual test of hand arm behind leg" of the original "manual muscle test of arm, leg or trunk" achieves a BLEU score (2-gram) of 0.30 and a WOS of 0.55. Although the BLEU score is relatively low, the adversary can still infer the tested body regions of the subject.

The Impacts of Pre-training In *sentence inference attack*, pre-training still makes the decoder generate more similar text than the vanilla decoder. We can conclude that pre-training does not only make the decoder more sensitive to keywords but also increase the accuracy of reconstruction.

10 POTENTIAL DEFENSES

10.1 Differential Privacy

Differential Privacy (DP) is a popular technique for protecting information about individuals in the dataset [14]. In the domain of machine learning, a differentially private stochastic gradient descent algorithm [1] has been proposed to reduce the risk of privacy. Google has already applied DP to large-scale image classification systems while maintaining high accuracy and minimizing computational cost [24]. The trade-off between utility and information leakage has been further investigated [2]. The main disadvantage of ensuring differential privacy is that it typically requires more noise infusion than traditional techniques.

As for the language modeling, it is demonstrated that DP can be used to train privacy-preserving models in various NLP applications [13, 19]. To satisfy the DP algorithm, each training sample in the dataset requires a user label. This requirement can be challenging for pre-trained language models since their training data is usually scraped from the public Web.

10.2 Privacy Preserving Mapping

The inference attacks against sentence embeddings are based on the key idea that public embeddings and private embeddings belong to the same embedding space. Privacy Preserving Mapping (PPM) provides a way to distort the embeddings before they are accessible to the third party [47]. On the one hand, PPM is trained to minimize the effectiveness of an inference attack by quantifying privacy leakage. On the other hand, to preserve the utility of the embeddings, the distortion of the PPM is constrained by a bound.

As a result, PPM can be applied to private embeddings so that attack models trained on public embeddings will suffer from distorted embedding space.

10.3 Avoid Providing Complete Sentence Embeddings

If an organization needs to share the embeddings of its confidential data with a third-party service provider, it can only provide the pooled version of the sequential embeddings or a masked version of the sequential embeddings. The incomplete sentence embeddings can reduce effectiveness of our decoder-based inference attack in accurately reconstructing the text. However, the performance of some downstream tasks such as machine translation and named entity recognition will also degrade.

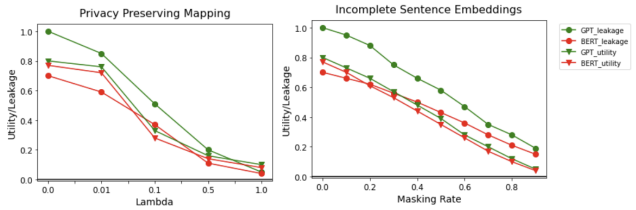


Figure 4: The evaluation of two defenses: privacy-preserving mapping and incomplete sentence embeddings.

10.4 Evaluation of Potential Defenses

We evaluate privacy preserving mapping (PPM) and incomplete embedding defenses against the keyword inference attack on the Skytrax dataset. Besides, we consider entity recognition as the downstream task to demonstrate the effects of defenses. Formally, given a sequence of embeddings $z = [z_1, z_2, ..., z_n]$ and a sequence of labels $y = [y_1, y_2, ..., y_n], y \in (0, 1)$, where 1 stands for targeted keywords and 0 stands for other words, the goal is to identify all the entities labeled with 1. We train a single-layer RNN model with a hidden state size of 300 to perform entity recognition. The performance is measured by the F-1 score, which represents the utility of the task. We report the count of reconstructed keywords (normalized) as information leakage.

For PPM, we follow the setup in [37]. Given a mapping $D_\theta : \mathcal{R}^d \Rightarrow \mathcal{R}^d$ which is trained to minimize the effectiveness of an imaginary adversary $\mathcal{A}_\phi$, formally, the learning objective is a minimax game by solving $min_\theta max_\phi \sum \mathcal{A}_\phi(D_\theta(z)) + \lambda||D_\theta(z) - z||$ [47]. The PPM is implemented as a regularization term in the minimax game so that the distortion of the embeddings is only allowed in a limited radius. Note that a high value of λ leads to a lower privacy budget. For incomplete embeddings, we apply a randomly generated mask to the embeddings [62], with the masking rate ranging from 0.1 to 0.9. Higher masking rate leads to sentence embeddings with more unknown tokens.

As shown in Figure 4, although both defenses can mitigate our decoder-based attack, they inevitably compromise the utility of the downstream task. We can observe the trade-off between utility and privacy for both of the defenses, which suggests that more sophisticated defense mechanisms that do not compromise utility to this extent need to be explored.

11 DISCUSSION

11.1 Practicality of Decoder-based Attack

We show that a transformer decoder can reconstruct coherent and informative texts, therefore revealing sensitive information. Compared to a prior classifier-based attack, it is a more practical threat, since the adversary does not need to know the secrets within the

dataset of interest. However, the classifier-based attack requires the adversary to know the keywords in the dataset or have the experience to create a set of keywords. Note that it is not practical to make the above assumption in many domains (e.g., medical, financial, industrial, and more).

As for the gradient-based attack, it cannot decode the contextual language embeddings and therefore produce noise outputs. Our method inverts the deep embeddings more accurately to generate well-structured sentences.

Finally, our method can still achieve robust performances in a black-box setting, while the baselines' performances degrade significantly. Since the black-box setting is closer to the real world, the decoder-based attack represents a practical threat to NLP applications.

11.2 Limitations

11.2.1 Reconstruction from Embeddings is Hard . We have demonstrated that the reconstructed texts can reveal high-level semantic information. Although our method can reconstruct high-quality sentences in many cases, it often produces lower-quality sentences as well. Both the proposed decoder-based attack and the previous gradient-based attack [52] rely on the accurate prediction of the probability distribution of each word. Hence, there are two challenges associated with reconstruction: (1) the word probability is usually distributed over a large vocabulary (thousands of tokens), which makes it hard to guarantee the right word is going to be selected, (2) it is difficult to reconstruct long texts. The second challenge is due to the fact that as the reconstruction goes further, the dependencies between current words and previous words decrease, which leads to less accurate results.

11.2.2 Implementation limitations. Although we have shown a successful decoder-based attack, there are several limitations of this work that could be explored in the future: (1) We have only tested the transformer decoder in our experiments. The performance of other architectures such as RNN [9] may provide more insights. (2) The hyperparameters (e.g., depth, learning rate, number of heads) of the decoder follow the default setting, which could have been improved by grid search.

11.3 Future Work

To improve the overall fluency and coherence of the reconstructions produced by our attack, we discuss the following future directions.

11.3.1 Pre-training the Decoder on Large Corpus. We have demonstrated the impacts of pre-training in previous results. The pre-trained decoder outperforms vanilla decoder on both *keyword inference attack* and *sentence inference attack*. However, the size of the dataset $\mathcal{D}_{train}$ is relatively small compared to the size of the training data from pre-trained language models. The generalizability of the decoder can be further improved by pre-training on a large corpus. This has the potential to boost the quality of reconstructions as well. For example, there exists a gold standard dataset in the medical domain, namely, MIMIC-III [23]. MIMIC-III includes more than 1 million caregiver notes of thousands of patients, which can be utilized by the adversary to pre-train the decoder. Such a pre-trained decoder can threaten many applications in the medical domain.

11.3.2 Upgrading Decoder Architecture. Currently, our decoder is inherited from the transformer's decoder, which exhibits the capacity of generating high-quality text. However, with the development of language models, more advanced decoder architectures are emerging. For example, Transformer-XL [10] was proposed to learn longer-term dependencies, while the vanilla transformer is limited by the fixed-length context required by the input. Therefore, Transformer XL can generate more coherent text. If the adversary adopts such an advanced decoder, the quality of the reconstructions will likely be enhanced without any other modifications.

11.3.3 Improving Quality of Decoding. To further improve the quality of the reconstructed text, we have employed various approaches, including top-k sampling, decaying temperature, and repetitive generation. However, there are more factors to consider, such as fluency and coherence of the language. Pascual *et al.* [40] presented a *plug-and-play* encoding method: Given a keyword or a topic, it added a shift to the probability distribution over the vocabulary towards semantically similar words. Despite the simplicity of this approach, it still enabled GPT-2 to generate more diverse and fluent sentences while guaranteeing the appearance of given guide words. The adversary can employ the *plug-and-play* method to improve the coherence of the decoding.

11.3.4 Handling Acronyms and Numbers. Acronyms and numbers may carry sensitive information (e.g., airline code and medical terms). However, it is challenging to reconstruct those accurately. Besides the aid of pre-training, a more sophisticated way to represent numbers [55] or acronyms [35] may benefit the decoding quality.

12 CONCLUSION

In this paper, we demonstrate that a decoder-based inference attack can recover coherent and informative text from sentence embeddings in some cases. It is a more practical attack since it can not only extract the sensitive keywords but also recover higher-level semantic information.

Our experiments reveal the superiority of our method against the baselines, especially in the black-box setting, which is closer to the real-world scenario. Even when the adversary does not know the targets in the dataset, they can still infer the semantics of the original text from the reconstructed text.

There are several ways to make the attack stronger, e.g., using a more advanced decoder architecture and pre-training on a larger corpus. In addition to that, we have also discussed some potential techniques to defend against such attacks. For instance, we believe differentially private training can prevent information leakage from embeddings to some extend.

ACKNOWLEDGMENTS

We thank the anonymous shepherd and the reviewers for their valuable suggestions. The work reported in this paper has been supported by the startup fund provided by The Pennsylvania State University.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria) *(CCS '16)*. Association for Computing Machinery, New York, NY, USA, 308–318. https://doi.org/10.1145/2976749.2978318

[2] Mário S. Alvim, Miguel E. Andrés, Konstantinos Chatzikokolakis, Pierpaolo Degano, and Catuscia Palamidessi. 2012. Differential Privacy: On the Trade-Off between Utility and Information Leakage. In *Formal Aspects of Security and Trust*, Gilles Barthe, Anupam Datta, and Sandro Etalle (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 39–54.

[3] Jack Bandy and Nicholas Vincent. 2021. Addressing "Documentation Debt" in Machine Learning Research: A Retrospective Datasheet for BookCorpus. *ArXiv* abs/2105.05241 (2021).

[4] Santiago Zanella Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew J. Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. 2020. Analyzing Information Leakage of Updates to Natural Language Models. *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (2020).

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *ArXiv* abs/2005.14165 (2020).

[6] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Xiaodong Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting Training Data from Large Language Models. In *USENIX Security Symposium*.

[7] Si Chen, Mostafa Kahla, Ruoxi Jia, and Guo-Jun Qi. 2021. Knowledge-enriched distributional model inversion attacks. In *Proceedings of the IEEE/CVF international conference on computer vision*. 16178–16187.

[8] Zhengsu Chen, Lingxi Xie, Jianwei Niu, Xuefeng Liu, Longhui Wei, and Qi Tian. 2021. Visformer: The Vision-Friendly Transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 589–598.

[9] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[10] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. *ArXiv* abs/1901.02860 (2019).

[11] Dina Demner-Fushman and Jimmy J. Lin. 2005. Knowledge Extraction for Clinical Question Answering: Preliminary Results.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv* abs/1810.04805 (2019).

[13] Christophe Dupuy, Radhika Arava, Rahul Gupta, and Anna Rumshisky. 2022. An Efficient DP-SGD Mechanism for Large Scale NLU Models. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 4118–4122.

[14] Cynthia Dwork. 2006. Differential Privacy. In *Automata, Languages and Programming*, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–12.

[15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In *CCS*. 1322–1333. https://doi.org/10.1145/2810103.2813677

[16] Matt Fredrikson, Eric Lantz, Somesh Jha, Simon M Lin, David Page, and Thomas Ristenpart. 2014. Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing. *Proceedings of the USENIX Security Symposium. UNIX Security Symposium* 2014 (2014), 17–32.

[17] Markus Freitag and Yaser Al-Onaizan. 2017. Beam Search Strategies for Neural Machine Translation. In *NMT@ACL*.

[18] Daniel Greenfield. 2014. Competition and service quality: New evidence from the airline industry. *Economics of Transportation* 3, 1 (2014), 80–89. https://doi.org/10.1016/j.ecotra.2013.12.005 Special Issue on Airlines and Airports.

[19] Ivan Habernal. 2021. When differential privacy meets NLP: The devil is in the detail. In *EMNLP*.

[20] John A Hawkins. 2014. *Word order universals*. Vol. 3. Elsevier.

[21] Sorami Hisamoto, Matt Post, and Kevin Duh. 2019. Membership Inference Attacks on Sequence-to-Sequence Models. *CoRR* abs/1904.05506 (2019). arXiv:1904.05506 http://arxiv.org/abs/1904.05506

[22] Fei Jiang, Yong Jiang, Hui Zhi, Yi Dong, Hao Li, Sufeng Ma, Yilong Wang, Qiang Dong, Haipeng Shen, and Yongjun Wang. 2017. Artificial intelligence in healthcare: past, present and future. *Stroke and Vascular Neurology* 2 (2017), 230 – 243.

[23] Alistair Johnson, Tom Pollard, Lu Shen, Li-wei Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Celi, and Roger Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific Data* 3 (05 2016), 160035. https://doi.org/10.1038/sdata.2016.35

[24] Alexey Kurakin, Steve Chien, Shuang Song, Roxana Geambasu, Andreas Terzis, and Abhradeep Thakurta. 2022. Toward training at imagenet scale with differential privacy. *arXiv preprint arXiv:2201.12328* (2022).

[25] Hye-Jin Kwon, Hyun-Jeong Ban, Jae-Kyoon Jun, and Hak-Seon Kim. 2021. Topic Modeling and Sentiment Analysis of Online Review for Airlines. *Information* 12, 2 (2021). https://doi.org/10.3390/info12020078

[26] Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *ICML*.

[27] Lishuang Li, Liuke Jin, Zhenchao Jiang, Dingxin Song, and Degen Huang. 2015. Biomedical named entity recognition based on extended Recurrent Neural Networks. In *2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. 649–652. https://doi.org/10.1109/BIBM.2015.7359761

[28] Yuhua Li, Zuhair Bandar, David McLean, and James O'Shea. 2004. A Method for Measuring Sentence Similarity and its Application to Conversational Agents.

[29] Chin-Yew Lin and Franz Josef Och. 2004. Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*. Barcelona, Spain, 605–612. https://doi.org/10.3115/1218955.1219032

[30] B. Liu, Ming Ding, Sina Shaham, Wenny Rahayu, Farhad Farokhi, and Zihuai Lin. 2020. When Machine Learning Meets Privacy: A Survey and Outlook. *ArXiv* abs/2011.11819 (2020).

[31] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 10012–10022.

[32] Saeed Mahloujifar, Huseyin A. Inan, Melissa Chase, Esha Ghosh, and Marcello Hasegawa. 2021. Membership Inference on Word Embedding and Beyond. *CoRR* abs/2106.11384 (2021). arXiv:2106.11384 https://arxiv.org/abs/2106.11384

[33] Shagufta Mehnaz, Sayanton V. Dibbo, Ehsanul Kabir, Ninghui Li, and Elisa Bertino. 2022. Are Your Sensitive Attributes Private? Novel Model Inversion Attribute Inference Attacks on Classification Models. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 4579–4596. https://www.usenix.org/conference/usenixsecurity22/presentation/mehnaz

[34] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[35] Dana Movshovitz-Attias and William W. Cohen. 2012. Alignment-HMM-Based Extraction of Abbreviations from Biomedical Text. In *Proceedings of the 2012 Workshop on Biomedical Natural Language Processing* (Montreal, Canada) *(BioNLP '12)*. Association for Computational Linguistics, USA, 47–55.

[36] Yuta Nakamura, Shouhei Hanaoka, Yukihiro Nomura, Naoto Hayashi, Osamu Abe, Shuntaro Yada, Shoko Wakamiya, Eiji Aramaki The University of Tokyo, Nara Institute of Science, Technology, The Department of Radiology, The University of Tokyo Hospital, The Department of Radiology, and Preventive Medicine. 2021. KART: Privacy Leakage Framework of Language Models Pre-trained with Clinical Records. *ArXiv* abs/2101.00036 (2021).

[37] Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. 2020. Privacy Risks of General-Purpose Language Models. In *2020 IEEE Symposium on Security and Privacy (SP)*. 1314–1331. https://doi.org/10.1109/SP40000.2020.00095

[38] Youcheng Pan, Chenghao Wang, Baotian Hu, Yang Xiang, Xiaolong Wang, Qingcai Chen, Junjie Chen, and Jingcheng Du. 2021. A BERT-Based Generation Model to Transform Medical Texts to SQL Queries for Electronic Medical Records: Model Development and Validation. *JMIR Med Inform* 9, 12 (8 Dec 2021), e32698. https://doi.org/10.2196/32698

[39] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (Philadelphia, Pennsylvania) *(ACL '02)*. Association for Computational Linguistics, USA, 311–318.

[40] Damian Pascual, Beni Egressy, Clara Meister, Ryan Cotterell, and Roger Wattenhofer. 2021. Keyword2Text: A Plug-and-Play Method for Controlled Text Generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. Association for Computational Linguistics.

[41] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained Models for Natural Language Processing: A Survey. *ArXiv* abs/2003.08271 (2020).

[42] Alec Radford and Karthik Narasimhan. 2018. Improving Language Understanding by Generative Pre-Training. *OpenAI*.

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

[44] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *CoRR*

abs/1910.10683 (2019). arXiv:1910.10683 http://arxiv.org/abs/1910.10683
[45] Laura Elena Raileanu and Kilian Stoffel. 2004. Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence* 41, 1 (2004), 77–93.
[46] Maria Rigaki and Sebastián García. 2020. A Survey of Privacy Attacks in Machine Learning. *ArXiv* abs/2007.07646 (2020).
[47] Salman Salamatian, Amy Zhang, Flávio du Pin Calmon, Sandilya Bhamidipati, Nadia Fawaz, Branislav Kveton, Pedro Oliveira, and Nina Taft. 2015. Managing Your Private and Public Data: Bringing Down Inference Attacks Against Your Privacy. *IEEE Journal of Selected Topics in Signal Processing* 9 (2015), 1240–1255.
[48] Chenze Shao, Yang Feng, and Xilin Chen. 2018. Greedy Search with Probabilistic N-gram Matching for Neural Machine Translation. In *EMNLP*.
[49] Virat Shejwalkar, Huseyin A. Inan, Amir Houmansadr, and Robert Sim. 2021. Membership Inference Attacks Against NLP Classification Models. In *NIPS*.
[50] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership Inference Attacks Against Machine Learning Models. In *2017 IEEE Symposium on Security and Privacy (SP)*. 3–18. https://doi.org/10.1109/SP.2017.41
[51] Harshdeep Singh, Robert West, and Giovanni Colavizza. 2020. Wikipedia citations: A comprehensive data set of citations with identifiers extracted from English Wikipedia. *Quantitative Science Studies* (2020), 1–19.
[52] Congzheng Song and Ananth Raghunathan. 2020. Information leakage in embedding models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 377–390.
[53] Max Spero. 2019. Improved Beam Search Diversity for Neural Machine Translation with k-DPP Sampling.
[54] Raymond Hendy Susanto, Dongzhe Wang, Sunil Yadav, Mausam Jain, and Ohnmar Htun. 2021. Rakuten's Participation in WAT 2021: Examining the Effectiveness of Pre-trained Models for Multilingual and Multimodal Machine Translation. In *Proceedings of the 8th Workshop on Asian Translation (WAT2021)*. Association for Computational Linguistics, Online, 96–105. https://doi.org/10.18653/v1/2021.wat-1.9
[55] Avijit Thawani, Jay Pujara, Pedro A Szekely, and Filip Ilievski. 2021. Representing numbers in NLP: a survey and a vision. *arXiv preprint arXiv:2103.13136* (2021).
[56] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. 2018. Towards Demystifying Membership Inference Attacks. *ArXiv* abs/1807.09173 (2018).
[57] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. 2021. Demystifying Membership Inference Attacks in Machine Learning as a Service. *IEEE Transactions on Services Computing* 14, 6 (2021), 2073–2089. https://doi.org/10.1109/TSC.2019.2897554
[58] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. *ArXiv* abs/1706.03762 (2017).
[59] Yungao Xie, Hongying Wen, and Q. Yang. 2021. Ternary Sentiment Classification of Airline Passengers' Twitter Text Based on BERT. *Journal of Physics: Conference Series* 1813 (2021).
[60] Wei Yang and Qiang Su. 2014. Process mining for clinical pathway: Literature review and future directions. In *2014 11th International Conference on Service Systems and Service Management (ICSSSM)*. 1–5. https://doi.org/10.1109/ICSSSM.2014.6943412
[61] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. Curran Associates Inc., Red Hook, NY, USA.
[62] Jiehang Zeng, Xiaoqing Zheng, Jianhan Xu, Linyang Li, Liping Yuan, and Xuanjing Huang. 2021. Certified robustness to text adversarial attacks by randomized [mask]. *arXiv preprint arXiv:2105.03743* (2021).
[63] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. 2020. The Secret Revealer: Generative Model-Inversion Attacks Against Deep Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
[64] Xuejun Zhao, Wencan Zhang, Xiaokui Xiao, and Brian Lim. 2021. Exploiting explanations for model inversion attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 682–692.

A APPENDIX

A.1 The Effect of the Size of Private Dataset

To verify the generalizabilty of our method, we conduct control study on the size of MT dataset and show the results in Figure 5. It is noticeable that the attack performance is roughly linearly correlated to the size of the dataset, which suggests that our attack generalizes well and the extent of information leakage scales up with the size of the dataset.

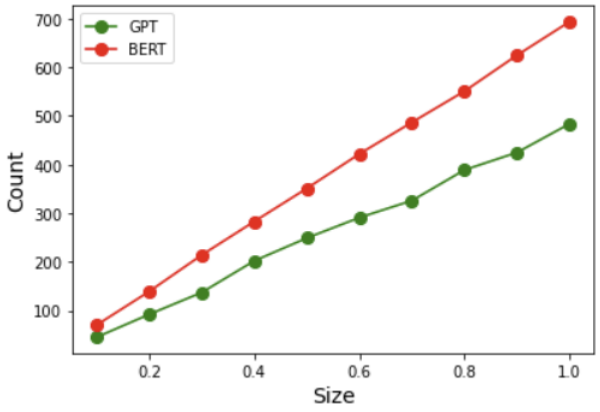


Figure 5: The attack performance on MT dataset of varying size

A.2 Comparison with Binary Classifiers

We have shown that our decoder-based attack is superior to multi-class-classifier-based attack in various settings. However, extending binary classifier to multi-class classifier may jeopardize its performance for scalability. To compare our attack directly with binary classifier attack, we randomly sample 10 keywords from the each training dataset then train 10 binary classifiers to perform inference attack. Specifically, we show the results of both white-box and black-box settings on Twitter dataset (airline) and MT dataset (medical).

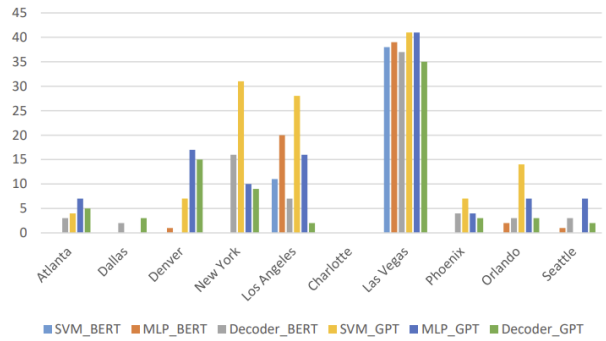
As displayed in Figure 6, the binary classifiers perform better than the decoder on most keywords (e.g. Los Angeles and Orlando) in the white-box setting. It is reasonable since binary classification is a relatively simple task given public/private datasets share the same distribution. However, in the black-box setting, the performances of binary classifiers degrade significantly due to the different distribution of the private dataset, while our decoder remains relatively more stable. Similar to our observations in previous experiments, our decoder tends to be more robust in black-box setting.

A.3 Randomly Selected Examples

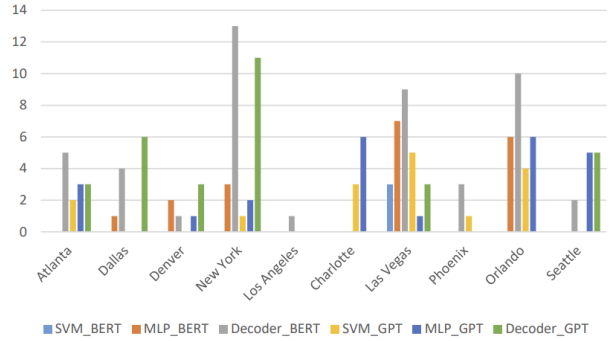
Although we show that our decoder can reconstruct high quality sentences in some cases, there still exist challenges in the reconstruction process. According to Table 4, the decoder handles acronyms and numbers less accurately. We leave this topic open for future research.

A.4 Ablation Study on k

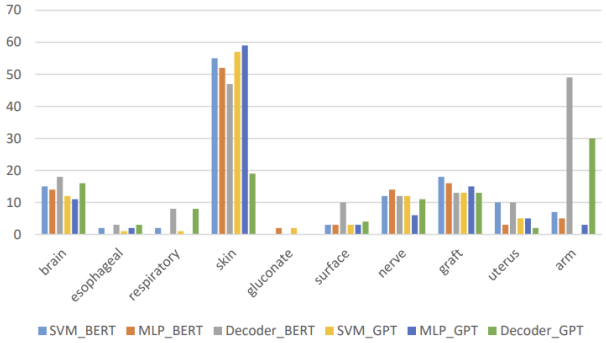
We show the impact of k (top-k words are finally kept and examined in keyword inference attack) in Figure 7. A larger k certainly makes our attack more accurate. However, it may also lower the efficiency as the adversary needs to check more words. We picked 20 to reach a balance between effectiveness and efficiency. As demonstrated in Figure 7, the curves of all keywords and the curves of unique keywords often tend to converge in the early stage ($k = 25$), which suggests that we do not need a very large k to achieve the best



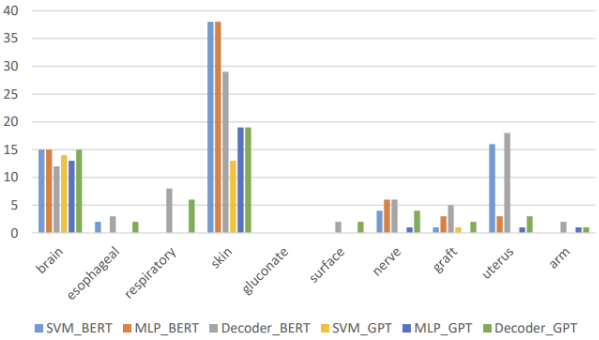
(a) Results of white-box attack on Twitter dataset



(b) Results of black-box attack on Twitter dataset



(c) Results of white-box attack on MT dataset



(d) Results of black-box attack on MT dataset

Figure 6: The results of binary classifiers.

Table 4: 10 Random Examples of Sentence Inference Attack

Domain	Original	Reconstructed
Airline	Paris (cdg) to Detroit (dtw)	Paris [PAD] to [PAD] dtw
	Lion air 8pm flight Bengkulu to Jakarta March 2	Lion air flight [PAD]ngbulu
	cx841 from New York jfk to hkg	c0 New York john
	Venice to Toronto on August 23 2013	Venice 23 made 19 143 2013
	Zurich to Ljubljana return	Zurich to Ljubljana 3 short
Medical	manual muscle test of arm, leg or trunk	manual test of hand arm behind leg
	injection of agent to destroy rib nerve	agent de from to approach rib
	mra scan of neck blood vessels	throat chest mra to than neck
	X-ray of abdomen, minimum of 3 views	X before and each 4 typically views
	X-ray of upper spine, 4 or 5 views	X ray 5 from at 4 6 views

results. Hence, our attack is not very sensitive to the choice of k when $k \geq 20$.

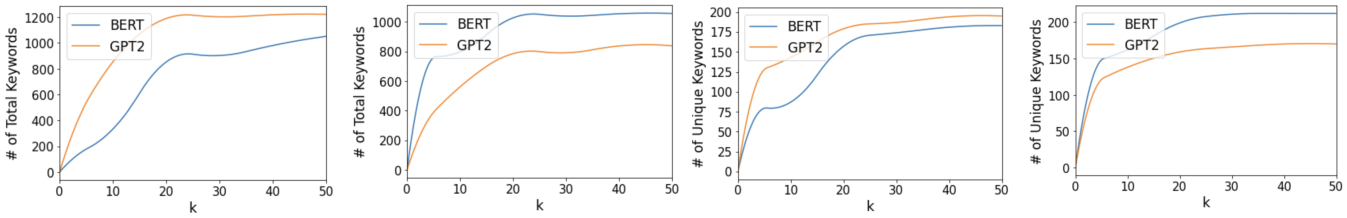


Figure 7: The ablation study of k. (a) The count of reconstructed keywords on Skytrax dataset (b) The count of reconstructed keywords on CMS dataset; (c) The number of unique keywords on Skytrax dataset; (d) The number of unique keywords on CMS dataset.