Mitigating Privacy Risks in LLM Embeddings from Embedding Inversion

Tiantian Liu , Hongwei Yao , Tong Wu , Zhan Qin , Feng Lin[*], Kui Ren , and Chun Chen

State Key Laboratory of Blockchain and Data Security, Zhejiang University
Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security
Corresponding author
Email: {tiantian, yhongwei, cocotwu, qinzhan, flin, kuiren, chenc}@zju.edu.cn

November 11, 2024

Abstract

Embeddings have become a cornerstone in the functionality of large language models (LLMs) due to their ability to transform text data into rich, dense numerical representations that capture semantic and syntactic properties. These embedding vector databases serve as the long-term memory of LLMs, enabling efficient handling of a wide range of natural language processing tasks. However, the surge in popularity of embedding vector databases in LLMs has been accompanied by significant concerns about privacy leakage. Embedding vector databases are particularly vulnerable to embedding inversion attacks, where adversaries can exploit the embeddings to reverse-engineer and extract sensitive information from the original text data. Existing defense mechanisms have shown limitations, often struggling to balance security with the performance of downstream tasks. To address these challenges, we introduce `Eguard`, a novel defense mechanism designed to mitigate embedding inversion attacks. `Eguard` employs a transformer-based projection network and text mutual information optimization to safeguard embeddings while preserving the utility of LLMs. Our approach significantly reduces privacy risks, protecting over 95% of tokens from inversion while maintaining high performance across downstream tasks consistent with original embeddings.

1   Introduction

In recent months, large language models (LLMs) such as ChatGPT [8], Claude [5], and ChatGLM [16] have garnered significant popularity among the general public. These models have demonstrated human-level accuracy and proficiency in a wide range of downstream tasks, including writing long stories, answering professional questions, and translating. In practical applications, online LLMs typically integrate several crucial components and techniques, such as vector databases, planning units, action execution units and prompt engineering. These integrated components enhance the models' capabilities in information retrieval, interaction, and logical reasoning, thereby overcoming limitations related to memory and timeliness.

Among aforementioned components, embedding vector databases [40, 54] play a critical role, functioning as a long-term memory system that alleviates the inherent memory constraints of LLMs. By integrating embedding databases with LLMs through a retrieval mechanism, retrieval-augmented generation (RAG) [20, 65] has emerged as a powerful tool

for developers in AI assistant APIs. Upon receiving a query (e.g., a partial sentence), the system first retrieves the top-k most relevant passages from the vector database to serve as prompts, thereby enhancing the quality and factual accuracy of the generated text. Recently, OpenAI's release of the embeddings API [1] has further empowered LLMs' capability by providing access to the embedding vectors of submitted queries, facilitating tasks such as similar query searches, clustering, and recommendations.

Despite the extensive advantages offered by embedding vector databases, their use also raises significant concerns regarding privacy leakage. Embedding vectors store rich, dense representations of text data, capturing both semantic and syntactic properties. This wealth of information, if not properly secured, poses a risk of sensitive or personal data exposure. Recent pioneering studies have revealed that embedding vectors are vulnerable to well-designed attacks targeting privacy leakage. These attacks can be broadly categorized into three main types: *embedding inversion attacks* [24, 27, 49], *membership inference attacks* [46, 49], and *attribute inference attacks* [33, 58]. Embedding inversion attacks exploit vulnerabilities in embedding vectors to extract and disclose sensitive information from the original input. If an adversary obtains an embedding vector, they may be able to reverse-engineer the raw input query. These risks highlight the pressing need for developing effective strategies to mitigate privacy leakage from embedding vectors.

In response to the threat of privacy leakage, researchers have proposed several general defense mechanisms against inversion attacks. These defenses can be categorized into *noise superposition-based defenses* [60, 64], *perturbation and rounding-based defenses* [19], and *Differential Privacy (DP)-based defenses* [29, 41]. However, existing defense approaches exhibit certain limitations that restrict their practical effectiveness against embedding inversion attacks. Defending against these attacks presents unique challenges. Firstly, embeddings encapsulate sensitive input features that are difficult to disentangle directly. Secondly, modifying embedding layers inherently compromises the accuracy of LLMs on downstream tasks. Thirdly, the flexibility of LLMs in performing various downstream tasks, such as sentiment analysis, natural language inference, and text summarization, poses a significant challenge for defense methods to preserve the utility of LLMs. This raises a pivotal question: *how can sensitive information in the embedding vector be effectively detached while maintaining its utility for downstream tasks?*

In order to address this question, we conduct a systematic investigation of the potential privacy vulnerabilities during the inference phase of LLMs. Our research reveals that the optimization process of the embedding model fosters a strong correlation between the original inputs and their respective embedding vectors. As a means to mitigate the threat posed by embedding inversion attacks, the most straightforward and effective approach is to disrupt this correlation. To solve this problem, we consider to project the original embedding space into a secured embedding space, wherein the sensitive features are detached.

In this paper, we propose **Embedding Guard** (`Eguard`), a novel defense mechanism designed to mitigate embedding inversion attacks by projecting embeddings through a transformer-based network, optimized using text mutual information. Our approach aims to reduce the correlation between the text and its corresponding embeddings, while ensuring that the transformed embeddings remain within the feature space required for downstream task performance. A primary challenge lies in the discrete nature of text and the continuous nature of embeddings, which complicates the direct computation of mutual information between the two. To handle the discrepancy between discrete text values and continuous embeddings, we utilize an autoencoder to encode and decode text, enabling the calculation of mutual information between the latent representation and the embeddings based on information entropy theory. This approach allows us to estimate the mutual information between text and embeddings, providing a more accurate measure of their association. To achieve an optimal balance between defense and functionality, we introduce a multi-task optimization

2

mechanism. This approach ensures that the defended embeddings remain within the downstream feature space while maintaining their performance. In summary, our contributions are as follows:

1) We systematically investigate the privacy risks inherent in the inference phase of LLMs. Our analysis highlights that embedding inversion attacks present a significant security threat to online LLMs, including widely used platforms like Chat-GPT.

2) We present `Eguard`, an innovative defense mechanism to mitigate embedding inversion attacks in large language models. Our approach integrates a transformer-based projection network with an autoencoder to reduce the correlation between text and its embeddings. By leveraging mutual information optimization and multi-task learning, `Eguard` ensures that the transformed embeddings effectively defend against inversion attacks while maintaining robust performance across a variety of downstream tasks.

3) We conduct comprehensive experiments on four embedding models and two ChatGPT embedding models under embedding inversion attacks. Our method protects over 95% of tokens from inversion and demonstrates harmlessness in four downstream tasks, achieving over 98% consistency with the original embeddings. Besides, we evaluate `Eguard`'s robustness against embedding perturbations, unseen training scenarios, and adaptive attacks, showing significant defense and performance.

2 Preliminaries

2.1 Embedding Model

An embedding model is a mathematical transformation that projects high-dimensional raw input into a lower-dimensional continuous vector space. Let $\mathcal{V}$ denote the discrete input space (i.e., vocabulary set), and let $\mathbb{R}^r$ represent the r-dimensional continuous vector space. For an input sequence of l words,
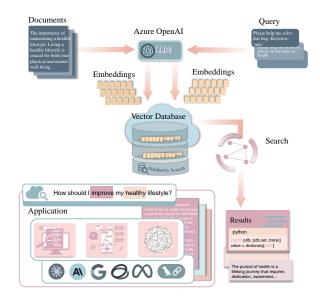

Figure 1: Overview of the embedding vector database: Clients upload queries or documents to embedding models, such as Azure OpenAI, to generate and store embeddings. By searching for the most relevant embeddings and using them to prompt the LLM, clients can obtain satisfying results and fine-tune customized private applications.

$x = [w_1, w_2, \ldots, w_l]$, the embedding model ϕ first encodes x into a sequence of vectors $\boldsymbol{v} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_l]$ using a tokenizer $\mathbf{W} : \mathcal{V} \to \mathbb{R}^r$:

$$\boldsymbol{v}_i = \mathbf{W}(w_i) \quad \text{for} \quad i = 1, 2, \ldots, l. \tag{1}$$

Next, the embedding model feeds the sequence of word vectors $\boldsymbol{v}$ to a Transformer (e.g., BERT [13], T5 [44], LLaMA [55]), obtaining a sequential hidden representation $\boldsymbol{h} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_l]$ for each word in x:

$$\boldsymbol{h} = \text{Transformer}(\boldsymbol{v}). \tag{2}$$

Finally, the embedding model f reduces the sequential hidden representation to a single vector representation e through a pooling operation, where e belongs to a d-dimensional vector space $\mathbb{R}^d$ (i.e. $e \in \mathbb{R}^d$). Typical techniques for pooling operation include mean pooling, max pooling, or using the hidden state of a special token (e.g., the `[CLS]` token in

BERT):

$$e = \text{Pooling}(\boldsymbol{h}). \tag{3}$$

Thus, the embedding model can be formally defined as a function $\phi : \mathcal{V} \to \mathbb{R}^d$, which maps an input sequence x to its corresponding embedding vector e (i.e., $\phi(x) = e$).

These embedding vectors encapsulate the semantic essence of the raw input, enabling various downstream tasks such as nearest neighbor search, retrieval, and classification. Beyond retrieval, they are essential for applications in ChatGPT, the Assistants API, and a variety of advanced AI tools, including natural language understanding (NLU) systems, sentiment analysis, text generation, and personalized recommendation systems.

2.2 Embedding Vector Database

The use of embeddings as feature representations has led to the development of embedding-based feature databases, which are specialized storage and retrieval systems optimized for handling dense vectors. Examples of such databases include Pinecone, Qdrant, Vdaas, Weaviate, and LangChain. These systems allow users to index, search, and manage embeddings efficiently. Embeddings are indexed into the server database, whose indexes are constructed by the hierarchical navigable small world graphs (HNSWG). At searching phase, a query embedding q is submitted to the vector database. The database returns the correlated embeddings that is closest to the query based on similarity metrics like cosine similarity $\frac{q \cdot e}{\|q\|\|e\|}$ and Euclidean distance $d(q, e) = \|q - e\|_2$. The server or third-party software is restricted to only accessing and storing text embeddings, rather than the raw text from clients, thereby safeguarding user privacy. Nonetheless, embedding inversion attacks pose a significant threat to the privacy and security of the stored embeddings.

Embedding-based feature databases are integral to many modern applications, enabling efficient and scalable retrieval of relevant information from large datasets. They support various use cases such as recommendation systems, search engines, and personalized content delivery by leveraging the power of embeddings to capture complex patterns in data. As the reliance on embedding-based systems continues to grow, so does the importance of safeguarding these embeddings against potential attacks. Understanding the intricacies of embeddings produced by PLMs and the architecture of embedding-based databases provides a foundation for developing effective defense strategies against embedding inversion attacks.

3 Threat Model

In this paper, we consider the embedding vector database attack scenario, which involves two parties in the threat model: embedding inversion *adversary* and *defender*.

Motivation. Embedding vector databases are extensively utilized in LLM platforms, such as Chat-GPT and Claude. These vectors are derived from user inputs, which often contain sensitive information. Although existing frameworks assert that embedding vectors do not leak sensitive user information, recent research indicates that embedding vector inversion poses a significant threat to LLM platforms [28, 49, 63]. This attack vector enables adversaries to potentially reconstruct original textual inputs from embedding vectors, thereby exposing sensitive user information if the vector database server is compromised. Thus, safeguarding embedding vectors against inversion attacks has become an urgent and critical necessity to ensure user privacy and data security.

Adversary's capability. Embedding inversion attacks aim at reverse mapping embedding vectors to texts. Given the queried embedding $\phi(x) \in \mathbb{R}^d$ of an input text x encoded by an encoding model pretrained on extensive corpora, an attacker builds an inverse model $\varphi(e) : e \to x$ with the intent of reconstructing the original text x. We consider a strong adversary adept at acquiring all users' embeddings stored within the vector database server, a feat possibly accomplished through man-in-the-middle or SQL injection. The capabilities of adversary can be summarized as:

- **Auxiliary dataset** $\mathcal{D}_{aux}$: The adversary has access to an auxiliary text dataset for querying,
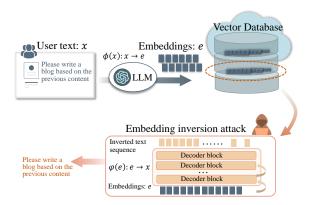
Figure 2: Workflow of Embedding Inversion Attack. The attacker breaches the vector database server and converts inverse embedding vectors back to the original raw data.

which closely matches the distribution of the target embedded text.

- **Breaking server**: The adversary can interact with the vector database server by submitting text queries and obtaining the corresponding embedding outputs, or they could potentially hack into the server to access the embedding vector database directly.

Adaptive Adversary. This paper considers the adaptive adversary, who is aware of our protection mechanisms and adjusts their attack strategies accordingly. Specifically, the adversary is capable of accessing both the protected embedding vectors and the defense model. The adaptive adversary leverages the defense model to generate these protected embedding vectors. These vectors then serve as the training set for the development of the inversion model. Through training this attack model, the adaptive adversary can reconstruct the raw input data from the protected embedding vectors.

4 Embedding Inversion Attacks

In the operational workflow of producing text embeddings, the process involves inputting original text into an encoder that has been pre-trained to map the text to a continuous vector space, thereby generating an embedding vector that represents the semantic and syntactic features of the original text. However, a critical concern arises in scenarios where the target encoder on the server is subject to repeated queries aimed at computing embeddings for an auxiliary text dataset. In the scenario of an embedding inversion attack, the attacker possesses the capacity to execute a limitless series of queries against the target encoder, with the intent of generating embeddings for an auxiliary text dataset $\mathcal{D}_{aux}$. The goal of the attacker is to search for a candidate text $\hat{x}$ that exhibits an embedding distribution similar to that of the original text. This is formulated as an optimization problem that minimizes the Euclidean distance between the embedding of the candidate text $\phi(\hat{x})$ and the embedding of the original text $\phi(x)$:

$$\hat{x} = \arg\min_{\hat{x} \in \mathcal{V}} \|\phi(\hat{x}) - \phi(x)\|_2. \qquad (4)$$

The brute-force approach to enumerating all possible combinations of words for the purpose of computing the embedding inversion as expressed in Eq. 4 is rendered computationally intractable due to the exponential growth in permutations with sequence length. Furthermore, the application of multi-label classification techniques is inherently constrained in their ability to predict ordered word sequences, as they are typically designed to handle classification tasks where the order of labels is inconsequential. In light of these limitations, the attacker adopts a decoder-based transformer model φ, a strategy that has been previously explored by Morris et al. [34] and expanded upon by Li et al. [27] This method involves the utilization of a state-of-the-art generative pre-trained transformer, such as GPT-2, to serve as the inversion decoder. The transformer is tasked with inverting the embedding produced by the encoder through a maximum likelihood estima-
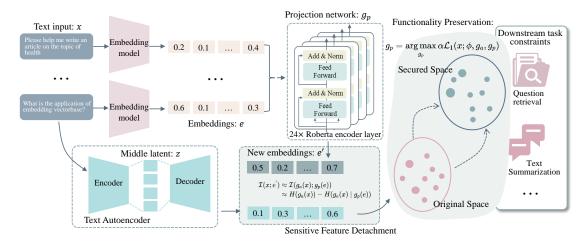
Figure 3: Overview of defense approach. `Eguard` contains a sensitive feature detachment module and a functionality preservation module.

tion framework:

$$\underset{\varphi}{\arg\max} \, \mathbb{E}_{x \sim \mathcal{D}_{aux}} \left[p\left(x | \varphi(\phi(x))\right) \right] \quad (5)$$

Considering the intrinsic sequential nature of text, the training objective for the decoder is to maximize the joint probability of the text token sequence. This is achieved by modeling the generation of each token as a conditional probability based on the preceding tokens and the initial embedding:

$$p(x) = \prod_{i=1}^{n} p(t_i | \phi(x), t_0, ..., t_{i-1}), \quad (6)$$

where t_i denotes the i-th token of the text sequence x. The embedding $\phi(x)$ is used as the initial token to provide context for the prediction of the subsequent word tokens. The training process is augmented by the self-attention mechanisms inherent in the transformer architecture, which enable the model to attend to different parts of the input sequence when generating each token. To ensure compatibility between the input dimensions of the decoder transformer and the embedding space, a multilayer linear layer is employed. This layer serves to unify the dimensions of the embedding features, facilitating the

seamless integration of the decoder with the encoder's output. By adopting this approach, the attacker can effectively invert the embeddings to reconstruct the original text, while preserving the linguistic structure and sequential order in natural language experssions.

5 Defense Approach

In this section, we introduce `Eguard`, a multi-task optimization-based mechanism aimed at mitigating embedding inversion attacks. To achieve this goal, `Eguard` projects the embedding vector from its original space to a secured space, with two primary objectives: detaching sensitive information from the embedding vector and preserving its functionality. We propose the three criteria (i.e., **effectiveness**, **harmlessness** and **robustness**) to evaluate the effectiveness of our method:

- **Effectiveness**: This criterion pertains to the efficacy of the modified embedding e' in obliterating sensitive feature information, thereby thwarting potential inversion attacks, while concurrently ensuring that the LLM maintains its utility in task performance.

- **Harmlessness**: Crucially, we ascertain that the

6

process of deriving the modified embedding $e^{'}$ does not engender any deleterious ramifications on the LLM's performance metrics. Specifically, it is imperative to validate that the adaptation process does not compromise the original accuracy levels of the LLM across pertinent downstream tasks.

- **Robustness**: The robustness of our defense strategy is gauged by its ability to withstand varied scenarios and adversarial maneuvers. A robust defense mechanism should evince consistency and reliability in preserving the LLM's performance integrity across diverse operational contexts, thus substantiating its efficacy in real-world deployment scenarios.

5.1 Overview

`Eguard` transforms the original embedding vector e into a secured embedding vector $e^{'}$ through a projection network:

$$e^{'} = g_p(e), \tag{7}$$

where g_p denotes the projection network. Embeddings e and $e^{'}$ are vectors with same dimension. The `Eguard` contains two main modules: *sensitive feature detachment* and *functionality preservation*. The detachment module employs an auxiliary network g_a to enhance the mutual information between the secured embedding and the raw input, effectively mitigating embedding inversion attacks. Simultaneously, the functionality preservation module assesses the performance of the secured embedding on downstream tasks to maintain its utility. Overall, the optimization process can be conceptualized as:

$$g_p = \arg\min_{g_p} -\alpha \mathcal{L}_1(x; \phi, g_a, g_p) + \mathcal{L}_2(x; f, \phi, g_p), \tag{8}$$

where, α is a hyper-parameter. $\mathcal{L}_1$ and $\mathcal{L}_2$ denote the losses of *sensitive feature detachment module* and *functionality preservation module*, respectively. Figure 3 illustrates the architecture of proposed `Eguard`.

5.2 Sensitive Feature Detachment

In this subsection, we introduce the sensitive feature detachment module and explain its design principles.

Embedding models are typically trained using unsupervised learning methods on extensive, unlabeled corpora. During the optimization process, embedding models are trained to reveal subtle differences among input data. Through large-scale optimization, the models establish strong correlation between raw inputs and their corresponding embedding vectors. Adversaries exploit this correlation to invert embedding vectors back to their original inputs. To mitigate this risk, the sensitive feature detachment module aims to detach the correlation between raw inputs and their embedding vectors, thereby disrupting the adversary's ability to reconstruct raw inputs from embedding vectors.

To achieve this, we project the original embedding vector to a secured embedding vector, and employ the mutual information as a metric to assess the correlation between raw input x and secured embedding vector $e^{'}$. For a given raw input sample (x, y) with its corresponding secured embedding vector $e^{'}$, the mutual information between the raw input x and the embedding vector $e^{'}$, as well as the ground-truth label of downstream tasks y, is denoted as $\mathcal{I}(e^{'}; x)$ and $\mathcal{I}(e^{'}; y)$, respectively. Overall, the optimization objective is:

$$\max_{e^{'}} \alpha \mathcal{I}(x; e^{'}) - \mathcal{I}(e^{'}; y), \tag{9}$$

where α is the hyper-parameter, e denotes embedding of input x, and $e^{'}$ is the secured version of e. Equation 9 defines the objective of optimization. Without loss of generality, the mutual information $\mathcal{I}(x; e^{'})$ can be calculated as:

$$\mathcal{I}(x; e^{'}) = H(x) - H(x \mid e^{'}), \tag{10}$$

where $H(x)$ is the differential entropy of x, and $H(x|e^{'})$ is the conditional differential entropy of x given $e^{'}$. However, since x and $e^{'}$, as well as $e^{'}$ and y, are variables with different dimensions. Consequently, the mutual information $\mathcal{I}(x; e^{'})$ and $\mathcal{I}(e^{'}; y)$ cannot be directly computed.

Mutual Information Estimation. Recent studies have explored the estimation of mutual information through the use of neural networks [7]. Building on these advancements, we introduce a pre-trained Autoencoder that are firstly trained on the large and unlabeled text corpus. The encoder part of the trained

Autoencoder as an auxiliary network is designed to project the raw input data into a continuous latent space: $g_a(x) : x \rightarrow z$, where $z \in \mathbb{R}^d$ denotes the latent space of Autoencoder. Consequently, the mutual information between the raw input and the embedding vector can be approximated as:

$$\begin{aligned} \mathcal{I}(x; e^{'}) &\approx \mathcal{I}(g_a(x); g_p(e)) \\ &\approx H(g_a(x)) - H(g_a(x) \mid g_p(e)) \end{aligned} \quad (11)$$

Finally, the objective of sensitive feature detachment module can be written as:

$$g_p = \underset{g_p}{\arg\max} \, \alpha \mathcal{L}_1(x; \phi, g_a, g_p), \quad (12)$$

where $\mathcal{L}_1$ denotes the mutual information estimation loss. During optimization, the projection network g_p is trained to maximize $\mathcal{I}(x; e^{'})$.

5.3 Functionality Preservation

In this subsection, we discuss the functionality preservation module. Projecting the original embedding vector to a new embedding space inevitably diminishes the performance of LLMs on downstream tasks. To address this challenge, we propose optimizing the projection network specifically for these downstream tasks. Consequently, the optimization of projection network is designed to maintain the utility of LLMs in these tasks. Formally, the objective of functionality preservation module is:

$$g_p = \underset{g_p}{\arg\min} \, \mathcal{L}_2(x; f, \phi, g_p), \quad (13)$$

where $\mathcal{L}_2$ represents the loss function that measures the performance of the modified embedding vector on the downstream tasks.

To extend our method to a variety of downstream tasks, we now discuss the adaptation of Equation 13 for different applications. For sentiment analysis and natural language inference tasks, we input the optimized embedding vectors into a multi-layer perceptron (MLP) to predict the corresponding labels, utilizing cross-entropy loss as our loss function, denoted as $\mathcal{L}_{CE}(MLP(e), y)$. For the question retrieval

task, we implement the multiple negatives ranking loss (MNRL), which is formulated as:

$$\mathcal{L}_{MNRL} = \sum_i \sum_j \max \left\{ 0, s(e_q, e_{p_i}) - s(e_q, e_{n_j}) + \beta \right\},$$

$$(14)$$

where e_q represents the embedding of the input query, e_{p_i} denotes the embedding of the ith positive sample, i.e., relevant questions, e_{n_j} corresponds to the embedding of the jth negative sample, i.e., irrelevant questions. s is the cosine similarity function between two embeddings, and β is a hyperparameter. Additionally, we consider the text summarization task. In this context, the embeddings are fed into a decoder to generate summaries. The predicted probabilities for each token in the output sequence are compared to the actual target sequence using the cross-entropy loss function. Integrating Equation 12 and Equation 13, the final optimization objective can be expressed as shown in Equation 8.

6 Evaluation

6.1 Experimental Setup

Datasets and language tasks. We utilize several text datasets, each corresponding to different language understanding tasks: sentiment analysis (SST), natural language inference (NLI), question retrieval (QR), and text summarization (TS). These datasets are encoded by the embedding models to evaluate their performance on tasks such as classifying sentiment, determining logical relationships between sentence pairs, retrieving relevant questions, and generating concise summaries from longer texts. Detailed descriptions of each dataset and task can be found in the Appendix E.

Embeddings models. We employ five state-of-the-art embedding models to conduct embedding inversion attacks and evaluate our defense mechanisms: T5, RoBERTa, MPNet, LLaMA, and Gemma. Each model is used with frozen parameters, utilizing their pre-trained weights. The models vary in architecture and dimensionality, with T5 and MPNet producing 768-dimensional embeddings, RoBERTa generating

Table 1: The overall performance of `Eguard` and other defense against embedding inversion attacks. Model: the type of embedding models, W/ Attack: the embedding inversion attacks on embeddings. SST2, NLI, QR, and TS are the corresponding downstream datasets.

Model	Method	SST2			NLI			QR			TS		
		F1(%)	Recall(%)	BLEU	F1(%)	Recall(%)	BLEU	F1(%)	Recall(%)	BLEU	F1(%)	Recall(%)	BLEU
T5	W/ Attack	93.9	93.3	0.836	96.5	95.0	0.789	98.2	97.9	0.976	95.2	94.7	0.901
	FGSM	14.2	16.1	0.092	25.9	19.4	0.121	36.3	34.4	0.230	39.3	38.6	0.245
	FreeLB	39.8	39.4	0.433	42.2	44.9	0.268	46.4	43.6	0.278	49.0	48.8	0.312
	DPforward	9.35	11.9	0.054	23.2	17.4	0.139	21.7	16.4	0.089	21.5	20.3	0.100
	Sanitization	6.75	11.0	0.030	23.2	16.3	0.095	23.5	21.7	0.103	22.6	22.1	0.092
	Ours	**4.75**	**4.40**	**0.019**	**5.35**	**4.47**	**0.034**	**3.57**	**4.14**	**0.014**	**3.56**	**4.44**	**0.011**
RoBERTa	W/ Attack	93.9	93.2	0.836	82.4	82.7	0.831	98.2	97.9	0.981	95.6	94.8	0.912
	FGSM	17.7	17.8	0.112	36.7	30.1	0.278	36.8	34.8	0.212	37.5	36.1	0.235
	FreeLB	18.5	17.6	0.104	35.7	30.2	0.278	51.3	48.4	0.238	49.8	48.7	0.320
	DPforward	14.3	10.1	0.014	23.7	18.9	0.257	24.4	20.5	0.147	22.1	20.1	0.131
	Sanitization	14.6	13.6	0.083	24.7	19.8	0.125	24.0	22.3	0.137	25.9	24.7	0.109
	Ours	**4.45**	**4.41**	**0.019**	**3.15**	**4.21**	**0.014**	**2.98**	**3.23**	**0.013**	**3.21**	**4.12**	**0.008**
MPNet	W/ Attack	93.9	93.4	0.837	83.2	83.3	0.822	98.8	97.9	0.980	96.1	95.2	0.906
	FGSM	17.4	17.6	0.115	36.4	29.8	0.269	37.0	34.8	0.221	37.8	36.8	0.212
	FreeLB	22.7	21.5	0.145	29.3	24.4	0.178	50.8	49.0	0.304	49.0	48.7	0.328
	DPforward	13.7	10.3	0.015	23.1	18.8	0.138	26.5	24.1	0.167	22.8	20.8	0.167
	Sanitization	9.61	17.1	0.036	17.8	15.0	0.088	23.8	21.8	0.118	24.6	23.4	0.116
	Ours	**5.15**	**4.43**	**0.012**	**4.55**	**4.13**	**0.011**	**4.12**	**4.21**	**0.009**	**4.31**	**5.24**	**0.010**
LLaMA2	W/ Attack	93.9	93.1	0.831	83.3	81.1	0.948	98.5	98.1	0.985	96.9	95.9	0.914
	FGSM	14.2	16.1	0.092	43.2	34.5	0.352	37.9	36.6	0.237	38.8	37.3	0.218
	FreeLB	44.3	43.6	0.446	41.1	34.2	0.351	50.6	49.6	0.289	47.1	46.9	0.283
	DPforward	12.2	13.0	0.058	25.4	21.9	0.115	25.7	24.3	0.121	22.0	22.5	0.108
	Sanitization	11.9	13.3	0.108	24.3	20.4	0.125	23.7	22.9	0.134	25.2	24.9	0.142
	Ours	**5.63**	**4.97**	**0.014**	**4.43**	**3.18**	**0.009**	**4.13**	**3.29**	**0.010**	**3.53**	**4.12**	**0.011**

1024-dimensional embeddings. Detailed descriptions of each model can be found in the Appendix B.

Attack methods. We utilize GPT-2 as the attacking decoder for embedding inversion, which has been pretrained on a dataset comprising 8 million web pages and possesses 774 million parameters. The maximum sequence length is set to 128 tokens. The inversion model is trained and inferred within the in-domain dataset, ensuring that the victim dataset is fully processed by the attacking model to optimize attacking performance. For model updates, we employ the Adam optimizer with a learning rate of 2e-5 and a batch size of 16.

Evaluation metrics. The evaluation of our defensive mechanisms against embedding inversion attacks is conducted through two primary categories of metrics: defensive metrics and harmlessness metrics. The defensive metrics quantify the efficacy of our defenses and include Recall, which measures the proportion of accurately inverted tokens in the victim sentence inputs; the F1 Score, representing the harmonic mean between the precision of the inverted tokens and recall; and the BLEU Score, which assesses the semantic quality of the inverted sentences as a whole on a scale from 0 to 1. The performance of the defense is considered more effective as these metrics decrease in value. Harmlessness metrics, on the other hand, evaluate the preservation of functionality in downstream tasks. Accuracy measures the overall correctness of predictions made by downstream networks in tasks such as sentiment analysis, natural language inference, and question retrieval when us-

Table 2: Evaluation of Harmlessness: Comparison of downstream task performance between original embeddings and embeddings subjected to different defense mechanisms. W/O attack denotes the original embedding without the influence of any attack or defense. Note that ROUGE is used for text summarization, and accuracy for the other tasks.

Model	Method	SST2(%)	NLI(%)	QR(%)	TS(%)
T5	W/O Attack	**94.3**	**81.4**	**96.9**	**39.6**
	FGSM	82.7	77.1	80.3	25.8
	FreeLB	84.4	80.1	84.4	28.6
	DPforward	60.7	46.9	48.9	19.4
	Sanitization	54.6	46.7	50.2	18.9
	Ours	**93.8**	**81.8**	**96.7**	**38.3**
RoBERTa	W/O Attack	**93.8**	**81.8**	**97.9**	**38.9**
	FGSM	70.2	79.4	80.1	21.3
	FreeLB	77.1	70.5	80.8	19.6
	DPforward	59.4	48.2	58.5	16.8
	Sanitization	47.3	43.2	60.3	17.7
	Ours	**94.0**	**80.8**	**96.8**	**37.6**
MPNet	W/O Attack	**93.8**	**79.7**	**99.1**	**37.8**
	FGSM	43.1	79.2	79.7	21.5
	FreeLB	78.7	78.9	51.6	20.7
	DPforward	62.1	52.1	83.3	18.9
	Sanitization	53.2	44.8	51.6	17.8
	Ours	**93.8**	**79.8**	**96.8**	**37.6**
LLaMA2	W/O Attack	**97.1**	**82.6**	**98.9**	**39.6**
	FGSM	78.7	77.4	80.2	25.4
	FreeLB	82.1	72.3	83.1	20.3
	DPforward	60.8	56.9	60.1	18.9
	Sanitization	51.9	40.6	50.1	18.3
	Ours	**96.8**	**81.8**	**98.1**	**38.7**

els and various text datasets, juxtaposing it against two types of adversarial training and two variants of differential privacy:

- Adversarial training: 1) FreeLB [67]: an adversarial training algorithm that enhances the robustness and invariance of language models by adding adversarial perturbations to word embeddings and minimizing the resultant adversarial risk around input samples; 2) FGSM [25]: Applies the fast gradient sign method (FGSM) to generate embedded features of inverse attacker gradients.
- Differential privacy: 1) Sanitization [14]: Utilizes local differential privacy (LDP) in natural language processing (NLP) by sanitizing sentence embeddings. 2) DPforward [15]: Applies differential privacy during the fine-tuning and inference stages of language models by directly perturbing embedding matrices in the forward pass.

The overall defense performance are as summarized in Table 1 and Appendix C. The first row of the table, labeled W/ attack, underscores the vulnerability of text embedding s to inversion attacks, irrespective of underlying embedding models. Upon evaluating the defenses against inversion attacks, it is observed that FGSM and FreeLB exhibit limited defensive capabilities, with the attacking metrics (F1, Recall) decreasing from 98% to 15%-50%, particularly for FreeLB. This limitation is conjectured to stem from the perturbations introduced by adversarial training, which though counteract the gradients exploited by inversion attackers, fail to fundamentally alter the original semantic space, thereby allowing room for embedding inversions. In contrast, the approaches leveraging differential privacy, namely DPForward and DP-Sanitization, demonstrate an improved enhancement in defense performance, yielding attacking F1/recall scores ranging from 9% to 29%. However, when pitted against our defense strategy, we achieve a noteworthy 90% effectiveness, which significantly reduces the inversion attack rate from above 95% to a mere 4%. This remarkable achievement safeguards over 95% of the tokens from being inverted, effectively preserving the integrity of the embeddings and enhancing their resistance against inversion attacks. It

ing the defended embeddings. The ROUGE Score is used specifically for text summarization, measuring the overlap of unigrams between the generated summary and reference summaries. Together, these metrics ensure that while our defenses robustly protect against embedding inversions, they do not adversely impact the performance of downstream applications.

6.2 Overall Performance

Defense against embedding inversion attacks. In this section, we evaluate the effectiveness of our defense across multiple LLM-based embedding mod-

becomes evident that our methodology outperforms these alternatives significantly. We further calculate that the proportion of exact matching words between inverted text and ground truth is negligible at 0.0%. This discrepancy arises from the frequent occurrence of common tokens such as "I, the, is, etc.," which are frequently inverted by attackers and consequently contribute to a slight inflation of F1/Recall metrics in our defense strategy, hovering slightly above 3%.

Evaluation on harmlessness. To assess the impact of our defense methods on downstream tasks, we reintroduce the optimized embeddings into the respective networks and compare their performance against the baseline performance of the raw embeddings. This evaluation aims to determine whether our defensive techniques adversely affect the practical utility of the embeddings. The first row of our comparisons, labeled "W/O attack," represents the standard performance of embeddings in classification, retrieval, and generation tasks without any defensive interventions. For sentiment analysis and natural language inference, the embeddings are input into a multilayer perceptron architecture. In the case of question retrieval, the objective is to identify the most similar question from an embedding bank. For text summarization, the embeddings are utilized by the T5 decoder to generate summaries. It is crucial to note that all downstream models and their training configurations remain consistent across the different defense methods to ensure a fair comparison. The results 2 and D reveal that while the DP methods exhibit reasonable defense effectiveness, they compromise the original semantic information, leading to a degradation in downstream performance. In contrast, FGSM and FreeLB show a less severe performance drop, with an increase of 20%-29% compared to DP, but they still fall short of matching the performance of the unaltered embeddings. Our defense strategy yields downstream accuracy that mirrors the original performance of the embeddings with over 98% consistency with the original embeddings, suggesting that our approach successfully preserves the functionality of the transformed space within the context of downstream tasks. Concurrently, it effectively distances this space from the original semantic domain, thereby mitigating the risk of embedding in-

version attacks. This balanced outcome underscores the efficacy of our method in safeguarding against privacy breaches without detrimental effects on the embeddings' utility in various applications.
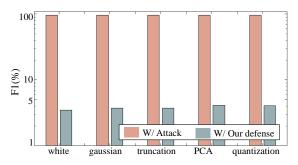
Defense overhead. The primary overhead of our defense comes from the additional training required for projection models. This increases training time for downstream models compared to undefended ones. For SST, NLI, and TS datasets, both defended and undefended models were trained for the same number of epochs, achieving convergence. We measured overhead using T5 and MPNet on two NVIDIA RTX A6000 GPUs. The results, averaged over five trials, show that MPNet's overhead is 1.6x to 2.4x, and T5's is 2.1x to 3.4x compared to undefended training. Specifically, the time per batch for MPNet with defense is 16.3ms, 25.1ms, and 24.7ms for the SST2, NLI, and TS datasets, compared to 9.6ms, 10.4ms, and 14.8ms without defense. For T5, the times with defense are 21.1ms, 34.2ms, and 31.6ms, versus 6.3ms, 12.9ms, and 14.9ms without defense. This one-time overhead is acceptable given the significant security and privacy benefits.

Table 3: Overhead training comparison of `Eguard` per training batch versus the original undefended training in the same setting.
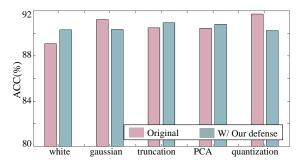
Model	MPNet			T5		
Dataset	SST2	NLI	TS	SST2	NLI	TS
W/O Defense	9.6ms	10.4ms	14.8ms	6.3ms	12.9ms	14.9ms
W/ Defense	16.3ms	25.1ms	24.7ms	21.1ms	34.2ms	31.6ms
Δ	6.7ms	14.7ms	9.9ms	14.8ms	21.3ms	16.7ms

6.3 Evaluation on Robustness

Robustness to embedding perturbations. Given that embedding vectors are typically stored on cloud servers and exchanged between clients, they are susceptible to interference and data compression. To evaluate the robustness of our defense strategy, we introduce five types of interferences into the target embeddings: 1) white noise, 2) Gaussian noise, 3) truncation, where we apply the ReLU function to obtain sparse embedding vectors, 4) PCA (Principal

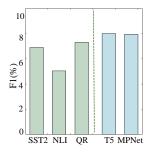(a) Defense results under different perturbations.

(b) Downstream task performance of our defended embeddings compared with the original embedding.

Figure 4: The defense performance and downstream task performance under embedding perturbations.
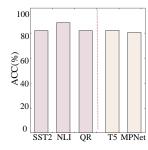
Component Analysis), used to reduce the dimensionality of the embeddings while retaining essential information, and 5) quantization, which involves converting high-precision data (e.g., 32 bits) to lower precision (e.g., 16 bits). We focus on the embeddings produced by T5 on the SST2 dataset to assess the impact of these interferences on our defense against inversion attacks and their harmlessness on downstream tasks. As depicted in Figure 4, while white noise and Gaussian noise may somewhat diminish the harmlessness of our method on downstream tasks, this degradation is primarily due to the inherent loss of semantic information in the vectors, which also affects the original performance. In the case of truncation and PCA, which both aim to reduce feature dimensions, the PCA and truncation methods exhibit comparable downstream performance. This observation can be attributed to the inherent robustness of high-dimensional feature vectors. Even when they lose some dimensional information, these vectors retain significant semantic content, ensuring that the essential semantic information is preserved. When our defense system encounters quantization, it successfully prevents 95% of tokens from being inverted and maintains an accuracy of over 89% on downstream tasks. Despite these perturbations, the defense remains largely effective, as they do not significantly alter the projected space, thereby preserving the defense's integrity.

Generalization to defenses unseen during training. We further investigate the generalization of our defense to embeddings not included in the training datasets of the projection network. We set up two test scenarios: one where the dataset is unseen during training, and another where the embedding model is not used in training. 1) Unseen Datasets: In this scenario, we train the projection network on one of the datasets—SST2, NLI, or QR, to produce target embeddings for projection, while using the remaining datasets for testing. For instance, if the projection network is trained on SST2, we then test it on NLI and QR. 2) Unseen Embedding Models: Here, we alternate between embedding models for training and testing on the SST2 dataset. For example, we train the projection network using T5 embeddings and test it using MPNet embeddings, and vice versa. The generalization testing outcomes are presented in Figure 5. In the first scenario, the defense performance is notably better when the NLI dataset is used for training compared to when SST2 or QR datasets are used. This improved performance is attributed to the larger size of the NLI dataset, which provides a more extensive variety of training examples. This allows the projection network to learn more complex patterns and reduces the risk of overfitting to specific dataset characteristics. In the second scenario, when transferring to unseen embedding models, there is a noticeable decline in both defense effectiveness and downstream task performance. Specifically, the defense against inversion attacks and the accuracy in downstream tasks such as sentiment analysis decrease. This deterioration occurs because different embedding models, such

(a) Defense performance.

(b) Downstream task performance.

Figure 5: The robustness performance under unseen training dataset or embedding models. The left section of each subgraph shows the results in the "unseen datasets" scenario, where the label "NLI" indicates that the defense model was trained only on the NLI dataset and tested on the SST2 and QR datasets. The same applies to the other labels (i.e., SST2, QR). The right section shows the results in the "unseen embedding models" scenario, where the label "T5" indicates that the defense model was trained only on T5 embeddings and tested on MPNet embeddings, and vice versa for MPNet.

as T5 and MPNet, operate in distinct feature spaces. A projection network trained on one model's embeddings struggles to adapt to the differing characteristics of another model's embeddings. Consequently, the common projection network faces challenges in maintaining robust defense mechanisms while preserving downstream performance across varied embedding spaces.

Our defense against stronger attackers. We consider an even stronger threat model where adversaries have white-box access to the embedding models, including our defense system, and aim to counteract the victim's projection network. We define two types of stronger attackers:

1) Full Training on Massive Projected Embeddings: Contrary to the previous scenario, where the attacker trained on a single victim embedding dataset, this assumption posits that the attacker has access to the same defense system used to generate anti-inverted embeddings for all victim datasets. The attacker bol-

sters their inversion attacks by training their inversion models on the defended embeddings to generate target text.

2) Training an Inverse Projection Network: In this scenario, the adversary attempts to train an inverse network to reverse the projected embedding space back to its original form. The adversary employs BERT as the projection network to process the defended embeddings and reconstruct the embeddings prior to projection. The overall loss function includes mean squared error (MSE) loss between the defended and original embeddings.

We implemented these two attack strategies on T5, MPNet, and LLaMA models for the sentiment analysis task. The evaluation results are presented in Figure 6. The results of the first attack type demonstrate that our defense effectively neutralizes the inversion of fully trained projected embeddings, regardless of the victim model's architecture. This is evidenced by the low performance of the attacks, with F1 scores around 6%, suggesting that the initial text information in the embeddings has been effectively obscured. This corroborates the theory that the lack of correlation between the original semantic space and the projected space disrupts the training relationship between the defended embeddings and the original text. For the inverse network training, the proportion of successfully inverted tokens from the embeddings across the three models is less than 8%, which represents only a 5% increase compared to clean training. This is attributed to the challenge of reconstructing the correlation that is eliminated by mutual information optimization. Consequently, when calculating the differences between the elements in the generated embeddings and the target, the gradient optimization direction tends to become disordered, diminishing its relevance to the original semantic spaces.

6.4 Ablation Studies

We assess the impact of mutual information loss and projection network on defense against embedding inversions with these two mechanisms removed separately. We present the result in Figure 7.

Impact of loss function. We explore the efficacy of various loss functions to replace the original
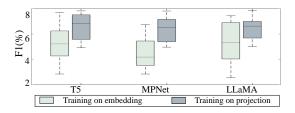
Figure 6: The defense performance against two types of stronger attackers: one that performs inversion attacks by training on massive projected embeddings, and another that employs an inverse projection network.

mutual information loss, including MSE loss, cosine similarity loss, Mahalanobis distance loss, and adversarial loss (i.e., maximizing attack losses), to assess their suitability for our defense strategy. Our experiments reveal that the Mahalanobis distance loss, MSE loss, and adversarial loss yield downstream accuracy below 86%, 84%, and 92%, respectively, and they fail to provide robust protection against inversion attacks. Cosine similarity loss, while demonstrating a slight improvement in performance, still falls short of matching the original mutual information loss. The discrepancy in performance can be attributed to the difficulty in achieving convergence for space optimization under these alternative loss functions. The mutual information loss, with its capacity to effectively align the original semantic space with the transformed space, is particularly adept at facilitating this optimization process. Consequently, our defense system, optimized with the original mutual information loss, outperforms the other loss functions in terms of both downstream task accuracy and defense against inversion attacks.

Impact of projection network. We now explore the influence of projection network architecture on performance. In this experiment, we compare the original network, comprising 24 RoBERTa layers, with the use of LLaMA3, transformer-XL, DeBERTa, XLNet, and MLP as projection networks for processing the SST2 dataset. Our findings reveal that shallow neural networks, such as MLP, do not exhibit satisfactory anti-inversion capabilities or performance in downstream tasks. Conversely, the large transformer-

based projection networks, while maintaining similar accuracy on the SST2 dataset, demonstrate a reduced susceptibility to inversion attacks.

This observation suggests that the depth and complexity of the projection network are pivotal in the defense against inversion attacks. The transformer-based networks, with their ability to model intricate relationships and patterns in the data, are better suited to preserving the integrity of the embeddings and resisting inversion. In contrast, the MLP, with its limited capacity for modeling such relationships, struggles to provide the same level of defense.
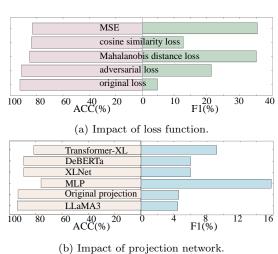

(a) Impact of loss function.

(b) Impact of projection network.

Figure 7: Ablation study results: (1) adjusting the loss function, and (2) investigating the impact of projection network architecture.

6.5 OpenAI embeddings

Embeddings facilitate the comprehension of relationships between content by machine learning models and other algorithms, enabling them to perform tasks such as clustering and retrieval. These representations underpin various applications, including knowledge retrieval in both ChatGPT and the Assistants API, as well as numerous retrieval-augmented generation developer tools. We have also assessed the efficacy of embedding inversion attacks and defense techniques on OpenAI embeddings.

14

To generate target embeddings, we utilized the OpenAI API, submitting texts from the SST2 dataset. The queried embedding models include text-embedding-3-small, text-embedding-3-large, and text-embedding-ada-002, which are the most recent and publicly available models from the OpenAI platform. These models feature embedding dimensions of 1536, 1536, and 3072, respectively. The comprehensive results of our evaluation are presented in Table 4.

Table 4: Defense performance against embedding inversion attacks on OpenAI embeddings

Method	ada-002(%)		3-small(%)		3-large(%)	
	F1	Recall	F1	Recall	F1	Recall
W/ Attack	93.91	93.13	93.83	93.07	83.92	82.17
FGSM	13.28	11.37	13.54	11.37	13.41	10.26
FreeLB	14.19	12.11	13.95	11.06	14.57	10.19
DPforward	14.20	9.97	14.57	18.89	18.17	98.81
Sanitization	13.18	9.62	12.07	12.94	12.96	10.46
Ours	5.28	4.72	5.12	4.73	3.88	3.96

The table above compares the performance of our method against other defense strategies in mitigating embedding inversion attacks on OpenAI embeddings. From the results, it is evident that the undefended embeddings (W/ Attack) exhibit high F1 and Recall scores, indicating a strong resistance to inversion attacks. This is expected as the embeddings are designed to maintain their integrity and resist such attacks.

Among the other defense methods, FGSM and FreeLB demonstrate relatively low F1 and Recall scores, indicating a significant decrease in resistance against inversion attacks. DPforward and Sanitization also show reduced performance compared to the undefended embeddings, although to a lesser extent. Our proposed defense method outperforms all other methods in reducing the effectiveness of inversion attacks. The F1 scores are notably low across all models, with 5.28% for ada-002, 5.12% for 3-small, and 3.88% for 3-large. The Recall values are similarly reduced, ranging from 3.96% to 4.73%. These results indicate that our method provides a robust defense mechanism, significantly lowering the attack success

rate compared to existing approaches. Overall, the results demonstrate that our defense method offers superior protection against embedding inversion attacks. By achieving the lowest F1 and Recall scores, our approach effectively safeguards the embeddings while maintaining lower susceptibility to inversion, thereby enhancing the security and robustness of the embedding models used in various applications.

6.6 Interpretability

To elucidate the interpretability of the designed defense mechanism for text embeddings, we have designed a comparison study that contrasts the distributions and feature distributions of three types of text embeddings. The interpretability analysis involves a comparative evaluation across three different types of text embeddings:1) the original embeddings, 2) the embeddings after defense, and 3) the embeddings with alternative pruning. The gpt3 embeddings, i.e., text-embedding-ada-002 and text-embedding-3-small, are the target in this interpretability experiment. This comparative approach aims to provide a comprehensive understanding of how the defense mechanism impacts the embeddings and preserves their semantic integrity. We visualize the embeddings using t-SNE, to provide a comprehensive understanding of the embeddings' structure, as shown in Figure 8.

The t-SNE plots of the original embeddings serve as a baseline, showcasing how the text data is initially distributed in the high-dimensional embedding space. These visualizations demonstrate a clear and coherent clustering of semantically similar texts, reflecting the embeddings' effectiveness in capturing the intrinsic relationships within the data. The embeddings after defense exhibit significant changes in their spatial distribution. The t-SNE plots indicate that while the clusters maintain a degree of consistency, the defense mechanism introduces shifts in the original feature space. These shifts suggest that the embedding transformation aims to thwart inversion attacks while also attempting to preserve the fundamental downstream performance. The slight expansion and reorganization of clusters imply that our defense mechanism effectively disrupts the embedding space, thereby ob-

scuring the original data. Embeddings subjected to alternative pruning techniques display a different pattern of distribution. The t-SNE plots for these embeddings show more pronounced fragmentation and dispersion of clusters compared to the defended embeddings. This fragmentation suggests that while pruning also aims to reduce the risk of inversion attacks, it may do so at the expense of more significant semantic integrity loss.

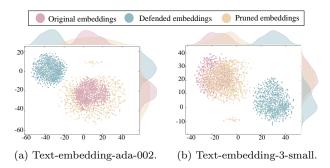(a) Text-embedding-ada-002.   (b) Text-embedding-3-small.

Figure 8: The visualized tSNE comparison of different embeddings.

Overall, these comparisons provide insights into the changes in the embeddings' distribution and feature distribution after our defense mechanism is applied. They help us understand how our defense mechanism affects the embeddings' structure and semantic representation, thereby demonstrating the interpretability of our defense mechanism.

7   Discussion

In this paper, we introduce `Eguard`, a defense framework against embedding inversion attacks. The objective is to utilize textual mutual information to establish a projection space with low correlation to the original semantic space, yet fulfill the requirements for effective performance in downstream tasks. The approach demonstrates superior performance in both protecting critical information and maintaining high performance for downstream tasks. On the one hand, compared with the differential privacy mechanism, our defense approach prevents the model accuracy degradation caused by noise. On the other

hand, compared with adversarial training, the strategy without modifying the training objectives ensures controllable computational complexity and training time, making it a better choice for large-scale applications.

Whereas our approach successfully limits the impact of embedding inversion attacks in LLMs, we concentrate mainly on the security of the text-generated LLMs. With the advancement of artificial intelligence technology, researchers have shifted their focus to multimodal LLMs (e.g., text-to-image LLMs [42,43], audio-text LLMs [3,10], and video-text LLMs [9,51]). Both the attacks and defenses against these models need to take into account the collaborative relationship between different modalities, which will undoubtedly pose a greater challenge. Future research should incorporate insights from this study to implement defenses on various modalities of embeddings, exploring individual privacy issues beyond the textual modality.

Moreover, we wish to build a common defense architecture with balanced defense performance across multiple heterogeneous models. Inspired by relevant research in the field of LLMs, we could publish the pre-trained defense models, combining with few-shot fine-tuning technology to facilitate the construction of more vertical application defenses. This will greatly increase the flexibility and effectiveness of our solutions against future attacks and models.

Overall, our approach achieves privacy protection without affecting model performance, demonstrating excellent defense capability and wide applicability.

8   Related work

8.1   Text embeddings are universal

Text embeddings are low-dimension representations of words, phrases, or documents in a continuous vector space, encapsulating semantic information and relationships among texts. Owing to their semantic understanding and low complexities, text embeddings have been widely applied to various natural language processing tasks and feature database storage [6, 17, 56]. For instance, Sentence-BERT [45],

SimCSE [21], and Sentence-T5 [38] fine-tune the encoder of pre-trained language models to produce embeddings for downstream tasks like classification, question-answering, semantic retrieval, and bitext mining. So far, there has been a series of works exploring versatile embedding models supporting a variety of application scenarios, like C-Pack in the field of general Chinese embeddings [61], OpenAI embeddings of text and code [36], and BGE in multilinguality of retrieval functionalities [32].

As embeddings continue to expand rapidly, an abundance of benchmarks have emerged to assess the quality of various embedding models across diverse tasks and multicultural domains. MTEB [35], SentEval [11], and BEIR [53] benchmarks assist practitioners in more easily selecting the appropriate embedding model for their specific needs and purposes.

Additionally, embeddings afford anonymity while compressing semantic information, which obviates the need for raw text. Several companies (such as JINA AI [4], SingleStore [47], and LangChain[2]) store embeddings in database servers or act as intermediary transfer stations to fine-tune large language models for personalized objectives and to train applications tailored to specific needs.

8.2 Text Embeddings Security

A large body of work has explored the vulnerability of text embeddings facing adversarial attacks, membership inference attacks, and reconstruction attacks. i) Adversarial attacks on text embeddings involve the manipulation of textual inputs, resulting in a model to misclassify or produce an unintended output. ii) Membership inference attacks on text embeddings involve determining whether a particular sample is part of the training dataset. iii) Reconstruction attacks on text embeddings are purposeful attempts to reconstruct sensitive or private information straight from the embeddings [2, 23, 27, 34, 49]. These attacks exploit the inadvertent leakage of information embedded within the representations, which may inadvertently disclose confidential details about the original text input.

In response, two primary strategies have emerged to guarantee the privacy-preserving nature of text embeddings. One approach utilizes differential privacy mechanisms, which add controlled noise to the embeddings to protect sensitive information. The other approach leverages adversarial training, where models are trained to produce embeddings that are robust to adversarial attacks, thereby preserving privacy. i) Differential privacy: A classical attempt is differentially private stochastic gradient descent (DP-SGD) [1], which add noise into gradients computed during each step of the stochastic gradient descent optimization process. Feyisetan et al. adopt d-privacy mechanism [18], wherein the indistinguishability of output distributions is scaled according to the distance between corresponding inputs. DP-Forward [15] implements an analytic matrix Gaussian mechanism, utilizing noise drawn from a possibly non-i.i.d. matrix Gaussian distribution to perturb the forward-pass embeddings. ii) Adversarial training: Most researches modify the training objectives of the model to maximize the loss of adversarial optimization [30, 57], leading to an improved privacy-accuracy trade-off. Dai et al. introduce interpretable adversarial training method by adding perturbations in the embedding vector to improve the robustness and generalization ability [12]. Additionally, to counteract the wrong prediction perturbed by the word-level adversarial attacks, Yang et al. propose fast triplet metric learning to generate unbiased embeddings by drawing words closer to their synonyms while pushing them away from their nonsynonyms within the embedding space [62].

Despite these advancements, balancing the trade-off between model performance and privacy remains a significant issue. Methods like DP-SGD can lead to substantial degradation in model accuracy due to the noise introduced, making it crucial to develop more efficient techniques that minimize this trade-off. Additionally, while adversarial training has shown promise, it often increases the computational complexity and training time, which can be a barrier for large-scale applications.

---

[2]https://www.langchain.com

9 Conclusion

In this paper, we addressed the critical issue of embedding inversion attacks in large language models and presented `Eguard`, a novel defense mechanism. By leveraging a transformer-based projection network and optimizing for text mutual information, our approach effectively reduces the risk of privacy breaches while ensuring the preservation of LLM utility on downstream tasks. Through extensive evaluation, we demonstrated the efficacy of `Eguard` in mitigating embedding inversion attacks, striking a balance between security and performance. Our work represents a significant step forward in safeguarding user data privacy and security in LLMs, paving the way for the development of more robust and secure natural language processing systems.

References

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

[2] Mohamed Abdalla, Moustafa Abdalla, Graeme Hirst, and Frank Rudzicz. Exploring the privacy-preserving properties of word embeddings: algorithmic validation study. *Journal of medical Internet research*, 22(7):e18055, 2020.

[3] Philip E Agre and David Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of the sixth National conference on Artificial intelligence-Volume 1*, pages 268–272, 1987.

[4] JINA AI. Choosing the right embeddings, 2024. https://jina.ai/embeddings/.

[5] AI Anthropic. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 1, 2024.

[6] Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024.

[7] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International conference on machine learning*, pages 531–540. PMLR, 2018.

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[9] Guo Chen, Yin-Dong Zheng, Jiahao Wang, Jilan Xu, Yifei Huang, Junting Pan, Yi Wang, Yali Wang, Yu Qiao, Tong Lu, et al. Videollm: Modeling video sequence with large language models. *arXiv preprint arXiv:2305.13292*, 2023.

[10] Yunfei Chu, Jin Xu, Xiaohuan Zhou, Qian Yang, Shiliang Zhang, Zhijie Yan, Chang Zhou, and Jingren Zhou. Qwen-audio: Advancing universal audio understanding via unified large-scale audio-language models. *arXiv preprint arXiv:2311.07919*, 2023.

[11] Alexis Conneau and Douwe Kiela. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*, 2018.

[12] Quanyu Dai, Xiao Shen, Liang Zhang, Qiang Li, and Dan Wang. Adversarial training methods for network embedding. In *The world wide web conference*, pages 329–339, 2019.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[14] Minxin Du, Xiang Yue, Sherman SM Chow, and Huan Sun. Sanitizing sentence embeddings (and labels) for local differential privacy. In *Proceedings of the ACM Web Conference 2023*, pages 2349–2359, 2023.

[15] Minxin Du, Xiang Yue, Sherman SM Chow, Tianhao Wang, Chenyu Huang, and Huan Sun. Dp-forward: Fine-tuning and inference on language models with differential privacy in forward pass. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 2665–2679, 2023.

[16] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. Glm: General language model pretraining with autoregressive blank infilling. *arXiv preprint arXiv:2103.10360*, 2021.

[17] Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. Language-agnostic bert sentence embedding. *arXiv preprint arXiv:2007.01852*, 2020.

[18] Oluwaseyi Feyisetan, Borja Balle, Thomas Drake, and Tom Diethe. Privacy-and utility-preserving textual analysis via calibrated multivariate perturbations. In *Proceedings of the 13th international conference on web search and data mining*, pages 178–186, 2020.

[19] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015.

[20] Chunjing Gan, Dan Yang, Binbin Hu, Hanxiao Zhang, Siyuan Li, Ziqi Liu, Yue Shen, Lin Ju, Zhiqiang Zhang, Jinjie Gu, et al. Similarity is not all you need: Endowing retrieval augmented generation with multi layered thoughts. *arXiv preprint arXiv:2405.19893*, 2024.

[21] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021.

[22] Max Glockner, Vered Shwartz, and Yoav Goldberg. Breaking nli systems with sentences that require simple lexical inferences. *arXiv preprint arXiv:1805.02266*, 2018.

[23] Kang Gu, Ehsanul Kabir, Neha Ramsurrun, Soroush Vosoughi, and Shagufta Mehnaz. Towards sentence level inference attack against pretrained language models. *Proceedings on Privacy Enhancing Technologies*, 2023.

[24] Ishrak Hayet, Zijun Yao, and Bo Luo. Invernet: An inversion attack framework to infer finetuning datasets through word embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5009–5018, 2022.

[25] Hoki Kim. Torchattacks: A pytorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950*, 2020.

[26] Seonhoon Kim, Inho Kang, and Nojun Kwak. Semantic sentence matching with densely-connected recurrent and co-attentive information. In *Proceedings of the AAAI conference on artificial intelligence*, pages 6586–6593, 2019.

[27] Haoran Li, Mingshi Xu, and Yangqiu Song. Sentence embedding leaks more information than you expect: Generative embedding inversion attack to recover the whole sentence. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 14022–14040. Association for Computational Linguistics, 2023.

[28] Senmao Li, Joost van de Weijer, Taihang Hu, Fahad Shahbaz Khan, Qibin Hou, Yaxing Wang, and Jian Yang. Stylediffusion: Prompt-embedding inversion for text-based editing. *arXiv preprint arXiv:2303.15649*, 2023.

[29] Xiaochen Li, Weiran Liu, Jian Lou, Yuan Hong, Lei Zhang, Zhan Qin, and Kui Ren. Local differentially private heavy hitter detection in data streams with bounded memory. *Proceedings of the ACM on Management of Data*, 2(1):1–27, 2024.

[30] Xiaoyang Liu, Shanghong Dai, Giacomo Fiumara, and Pasquale De Meo. An adversarial training method for text classification. *Journal of King Saud University-Computer and Information Sciences*, 35(8):101697, 2023.

[31] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[32] Kun Luo, Zheng Liu, Shitao Xiao, and Kang Liu. Bge landmark embedding: A chunking-free embedding method for retrieval augmented long-context large language models. *arXiv preprint arXiv:2402.11573*, 2024.

[33] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 691–706. IEEE, 2019.

[34] John X Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander M Rush. Text embeddings reveal (almost) as much as text. *arXiv preprint arXiv:2310.06816*, 2023.

[35] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022.

[36] Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005*, 2022.

[37] Minh-Tien Nguyen, Chien-Xuan Tran, Duc-Vu Tran, and Minh-Le Nguyen. Solscsum: A linked sentence-comment dataset for social context summarization. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2409–2412, 2016.

[38] Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*, 2021.

[39] Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*, 2021.

[40] Sinan Ozdemir. *Quick start guide to large language models: strategies and best practices for using ChatGPT and other LLMs*. Addison-Wesley Professional, 2023.

[41] Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. Privacy risks of general-purpose language models. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1314–1331. IEEE, 2020.

[42] Jie Qin, Jie Wu, Weifeng Chen, Yuxi Ren, Huixia Li, Hefeng Wu, Xuefeng Xiao, Rui Wang, and Shilei Wen. Diffusiongpt: Llm-driven text-to-image generation system. *arXiv preprint arXiv:2401.10061*, 2024.

[43] Leigang Qu, Shengqiong Wu, Hao Fei, Liqiang Nie, and Tat-Seng Chua. Layoutllm-t2i: Eliciting layout guidance from llm for text-to-image generation. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 643–654, 2023.

[44] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[45] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[46] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[47] SingleStore. Using singlestoredb and chat-gpt for custom data sets, 2023. `https://www.singlestore.com/blog/using-singlestoredb-chatgpt-for-custom-data-sets/`.

[48] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[49] Congzheng Song and Ananth Raghunathan. Information leakage in embedding models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 377–390, 2020.

[50] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems*, pages 16857–16867, 2020.

[51] Yunlong Tang, Jing Bi, Siting Xu, Luchuan Song, Susan Liang, Teng Wang, Daoan Zhang, Jie An, Jingyang Lin, Rongyi Zhu, et al. Video understanding with large language models: A survey. *arXiv preprint arXiv:2312.17432*, 2023.

[52] Gemma Team. Gemma. 2024.

[53] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*, 2021.

[54] Oguzhan Topsakal and Tahir Cetin Akinci. Creating large language model applications utilizing langchain: A primer on developing llm apps fast. In *International Conference on Applied Engineering and Natural Sciences*, pages 1050–1056, 2023.

[55] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[56] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*, 2023.

[57] Xiaosen Wang, Yichen Yang, Yihe Deng, and Kun He. Adversarial training with fast gradient projection method against synonym substitution based text attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[58] Yu Wang, Yuying Zhao, Yushun Dong, Huiyuan Chen, Jundong Li, and Tyler Derr. Improving fairness in graph neural networks via mitigating sensitive attribute leakage. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 1938–1948, 2022.

[59] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*, 2017.

[60] Jing Wen, Siu-Ming Yiu, and Lucas CK Hui. Defending against model inversion attack by adversarial examples. In *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 551–556. IEEE, 2021.

[61] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighof. C-pack: Packaged resources to advance general chinese embedding. *arXiv preprint arXiv:2309.07597*, 2023.

[62] Yichen Yang, Xiaosen Wang, and Kun He. Robust textual embedding against word-level adversarial attacks. In *Uncertainty in Artificial Intelligence*, pages 2214–2224. PMLR, 2022.

[63] Yong Yang, Xuhong Zhang, Yi Jiang, Xi Chen, Haoyu Wang, Shouling Ji, and Zonghui Wang. Prsa: Prompt reverse stealing attacks against large language models. *arXiv preprint arXiv:2402.19200*, 2024.

[64] Ziqi Yang, Bin Shao, Bohan Xuan, Ee-Chien Chang, and Fan Zhang. Defending model inversion and membership inference attacks via prediction purification. *arXiv preprint arXiv:2005.03915*, 2020.

[65] Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, and Bin Cui. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*, 2024.

[66] Ming Zhong, Pengfei Liu, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. Searching for effective neural extractive summarization: What works and what's next. *arXiv preprint arXiv:1907.03491*, 2019.

[67] Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. Freelb: Enhanced adversarial training for natural language understanding. *arXiv preprint arXiv:1909.11764*, 2019.

A   Impact of attacking decoder

In previous evaluations, we utilized GPT-2 as the attacking decoder. In this experiment, however, we employ various decoders from LLMs to invert the original text from embeddings and evaluate both the effectiveness of the attacks and the performance of our defense mechanisms. The target dataset is SST2, and the embedding models used are T5, LLaMA2-7B, and Gemma2-9B. For inversion decoders, we selected

LLaMA2-7B, LLaMA3-8B, and Gemma2-9B, respectively. The results of these attacks, along with our corresponding defensive measures, are presented in Table 5. It is evident that all three decoders exhibit similar attacking performances across different victim embeddings. We speculate that thorough training on datasets and the architecture of large-scale models contribute to their potent inversion capabilities. Importantly, our findings indicate that our defense method remains effective across various decoding models.

Table 5: The performance under different attacking decoders

Model	Defense	Attacking Decoder (F1%)		
		LLaMA2	LLaMA3	Gemma2
T5	W/O	97.2	98.4	95.8
	W/	3.70	3.91	4.03
LLaMA2	W/O	92.9	94.4	91.8
	W/	4.10	4.04	3.91
Gemma2	W/O	93.5	98.2	98.6
	W/	4.20	4.01	4.04

B   Embeddings Models

To conduct embedding inversion attacks and validate our defense mechanisms, we have chosen four cutting-edge embedding models. The parameters of these models remain frozen, and we utilize the pretrained weights as provided in their respective original GitHub repositories.

T5 [39], introduced by Google, is an innovative encoder-decoder model that has been pre-trained on a diverse and voluminous dataset, leveraging transfer learning for enhanced performance. In our work, we utilize the encoder of the T5-large model to project sentences into a 768-dimensional vector space. The T5-large model has a total of 770 million parameters.

RoBERTa [31], an optimized variant of the BERT model, undergoes pretraining with a focus on the masked language modeling objective. The pretrained RoBERTa-large model encodes the input sentence into a 1024 dimension vector and is fine-tuned in on

a 1B sentence pairs dataset. The RoBERTa-large model parameters are 1360 MB in size.

MPNet [50], introduced by Microsoft, combines the strengths of masked language modeling from BERT and permuted language modeling from XLNet, resulting in improved performance across a range of downstream benchmark tasks. We use the encoder of MPNet to produce 768-dimensional embeddings, which are 420 MB in size.

LLaMA [55], proposed by Meta AI, stands as a state-of-the-art foundational large language model that is tuned by using supervised fine-tuning and reinforcement learning with human feedback. We leverage the LLaMA2-7B, LLaMA3-8B, and LLaMA3-70B as our encoder to generate 4096 dimensional embeddings.

Gemma [52], proposed by Google, is designed for text generation tasks like question answering, summarization, and reasoning. These models are small and can be used on devices with limited resources, such as laptops and desktops. We use Gemma2-9B as the encoder to obtain the 3584-dimensional embedding vector.

C Rest Results of Defense Performance

D Rest Results of Evaluation of Harmlessness

Table 7: Evaluation of Harmlessness: Comparison of downstream task performance between original embeddings and embeddings subjected to different defense mechanisms. W/O attack denotes the original embedding without the influence of any attack or defense.

Model	Method	SST2(%)	NLI(%)	QR(%)	TS(%)
	W/O Attack	**94.9**	**85.0**	**97.9**	**39.8**
	FGSM	81.3	77.9	83.8	20.1
LLaMA3-8B	FreeLB	88.6	74.6	81.5	19.8
	DPforward	70.1	64.7	72.9	16.7
	Sanitization	62.4	58.8	69.7	16.3
	Ours	**93.2**	**81.1**	**97.9**	**39.6**
	W/O Attack	**95.2**	**82.7**	**99.8**	**40.1**
	FGSM	84.5	77.0	72.1	24.2
LLaMA3-70B	FreeLB	81.8	77.5	74.1	22.3
	DPforward	71.1	69.6	63.1	20.1
	Sanitization	69.3	72.8	58.2	18.9
	Ours	**94.8**	**80.3**	**97.9**	**39.8**
	W/O Attack	**93.4**	**80.8**	**96.3**	**38.9**
	FGSM	85.8	73.4	82.7	23.1
Gemma2-9B	FreeLB	79.4	65.7	73.5	20.4
	DPforward	74.7	55.2	67.7	17.9
	Sanitization	63.5	51.4	63.7	17.7
	Ours	**93.4**	**80.8**	**95.8**	**38.6**

E Datasets and Language Tasks.

we introduce the text dataset that will be encoded by the embedding model and its corresponding language understanding tasks, including sentiment analysis, natural language inference, question retrieval, and text summarization. For brevity, we refer to each dataset by its respective acronym.

SST: Sentiment Analysis. The Stanford sentiment treebank (SST2) [48] is a corpus containing

Table 6: The overall performance of `Eguard` and other defense against embedding inversion attacks. Model: the type of embedding models, W/ Attack: the embedding inversion attacks on embeddings. SST2, NLI, QR, and TS are the corresponding downstream datasets.

Model	Method	SST2			NLI			QR			TS		
		F1(%)	Recall(%)	BLEU	F1(%)	Recall(%)	BLEU	F1(%)	Recall(%)	BLEU	F1(%)	Recall(%)	BLEU
LLaMA3-8B	W/ Attack	94.2	93.6	0.839	91.4	88.5	0.817	89.2	88.5	0.870	97.2	94.3	0.937
	FGSM	13.3	15.8	0.097	24.9	21.4	0.149	36.8	34.7	0.227	40.3	38.3	0.272
	FreeLB	25.6	21.7	0.136	44.6	44.3	0.352	45.9	40.8	0.303	49.8	47.2	0.327
	DPforward	8.75	11.5	0.046	24.0	22.7	0.172	21.5	18.9	0.121	21.9	20.8	0.105
	Sanitization	9.57	10.7	0.066	25.8	24.3	0.113	23.2	23.8	0.139	25.0	24.2	0.109
	Ours	**3.26**	**2.88**	**0.003**	**4.27**	**4.56**	**0.013**	**3.73**	**4.38**	**0.006**	**4.38**	**4.48**	**0.008**
LLaMA3-70B	W/ Attack	96.5	94.2	0.924	91.0	88.0	0.856	96.8	97.6	0.958	98.6	95.6	0.923
	FGSM	19.4	20.3	0.117	32.3	27.3	0.215	37.3	35.6	0.234	35.6	32.7	0.194
	FreeLB	27.0	25.9	0.205	38.3	37.6	0.286	47.3	44.8	0.362	48.1	46.6	0.315
	DPforward	9.45	10.2	0.082	20.5	18.3	0.143	23.2	22.7	0.102	22.5	21.7	0.101
	Sanitization	11.9	13.7	0.103	19.3	17.7	0.085	24.0	23.7	0.120	27.0	26.4	0.108
	Ours	**4.67**	**5.06**	**0.007**	**4.01**	**4.23**	**0.007**	**4.07**	**3.75**	**0.010**	**4.89**	**4.97**	**0.005**
Gemma2-9B	W/ Attack	95.8	92.5	0.869	84.7	81.2	0.830	90.7	86.8	0.870	97.2	96.9	0.950
	FGSM	15.6	17.1	0.101	33.4	31.3	0.219	36.8	34.7	0.224	34.7	33.9	0.222
	FreeLB	33.8	28.8	0.221	41.8	40.9	0.265	53.7	50.1	0.372	49.9	48.2	0.314
	DPforward	9.54	10.7	0.072	20.0	19.8	0.128	20.7	19.5	0.113	19.9	17.6	0.099
	Sanitization	9.62	10.9	0.067	20.9	17.9	0.095	24.1	23.8	0.109	22.1	20.5	0.094
	Ours	**4.20**	**4.50**	**0.007**	**4.38**	**4.42**	**0.012**	**4.81**	**4.63**	**0.009**	**4.50**	**4.60**	**0.012**

67,000 sentences sourced from movie reviews, each meticulously annotated with human-derived sentiment labels. The goal of sentiment analysis is to determine the emotional tone of a sentence, categorizing it as positive or negative, which is a type of classification problem in language processing.

NLI: Natural Language Inference. The task of natural language inference (NLI) involves determining the logical relationship between two sentences—a premise and a hypothesis. Specifically, the goal is to classify the relationship as entailment (the premise supports the hypothesis), contradiction (the premise contradicts the hypothesis), or neutrality (there is no evident logical correlation). The Stanford natural language inference (SNLI) [22] and multi-genre natural language inference (MultiNLI) [26] corpus are prominent resources developed for advancing machine learning models' capabilities in sentence comprehension. In our experimental setup, we consolidate the SNLI and MultiNLI datasets into a unified NLI dataset, which encompasses premise sentences

from a diverse range of sources, including transcriptions, reports, speeches, letters, and works of fiction. This amalgamated dataset comprises 1 million sentence pairs.

QR: Question Retrieval. The task of question retrieval involves identifying the most relevant question within a vast corpus containing thousands of questions. As an information retrieval task, its objective is to match a given question with its closest duplicate in the corpus. The dataset used for this purpose is the Quora duplicates questions dataset [59], which comprises 500,000 questions covering a wide array of topics and issues originally posted on Quora.

TS: Text Summarization. Text Summarization (TS) entails the compression of extensive textual information into succinct summaries. This dataset amalgamates content from USAToday obtained from CNN [66] and SoLSCSum sourced from Yahoo News [37]. It encompasses roughly 10k sentences extracted from a corpus of over 200 articles, each paired with a gold-standard summary crafted

by human annotators. Analyzing the TS dataset reveals that the source articles exhibit an average word count of 500, comprising roughly 22 sentences, while the summaries are significantly abbreviated, averaging a length of 20 words.