Are You Copying My Model? Protecting the Copyright of Large Language Models for EaaS via Backdoor Watermark

Wenjun Peng[1*], **Jingwei Yi**[1*], **Fangzhao Wu**[2†], **Shangxi Wu**[3], **Bin Zhu**[2], **Lingjuan Lyu**[4], **Binxing Jiao**[5], **Tong Xu**[1†], **Guangzhong Sun**[1], **Xing Xie**[2]

[1]University of Science and Technology of China [2]Microsoft Research Asia
[3]Beijing Jiaotong University [4]Sony AI [5]Microsoft STC Asia
{pengwj,yjw1029}@mail.ustc.edu.cn wufangzhao@gmail.com
wushangxi@bjtu.edu.cn {binzhu,binxjia,xingx}@microsoft.com
lingjuan.lv@sony.com {tongxu,gzsun}@ustc.edu.cn

Abstract

Large language models (LLMs) have demonstrated powerful capabilities in both text understanding and generation. Companies have begun to offer Embedding as a Service (EaaS) based on these LLMs, which can benefit various natural language processing (NLP) tasks for customers. However, previous studies have shown that EaaS is vulnerable to model extraction attacks, which can cause significant losses for the owners of LLMs, as training these models is extremely expensive. To protect the copyright of LLMs for EaaS, we propose an Embedding Watermark method called EmbMarker that implants backdoors on embeddings. Our method selects a group of moderate-frequency words from a general text corpus to form a trigger set, then selects a target embedding as the watermark, and inserts it into the embeddings of texts containing trigger words as the backdoor. The weight of insertion is proportional to the number of trigger words included in the text. This allows the watermark backdoor to be effectively transferred to EaaS-stealer's model for copyright verification while minimizing the adverse impact on the original embeddings' utility. Our extensive experiments on various datasets show that our method can effectively protect the copyright of EaaS models without compromising service quality. Our code is available at https://github.com/yjw1029/EmbMarker.

1 Introduction

Large language models (LLMs) such as GPT-3 (Brown et al., 2020) and LLAMA (Touvron et al., 2023) have demonstrated exceptional abilities in natural language understanding and generation. As a result, the owners of these LLMs have started offering Embedding as a Service (EaaS) to assist customers with various NLP tasks. For example, OpenAI offers a GPT3-based embedding API [1], which generates embeddings at a cost for query texts. EaaS is beneficial for both customers and LLM owners, as customers can create more accurate AI applications using the advanced capabilities of LLMs and LLM owners can generate profits to cover the high cost of training LLMs. However, recent research (Liu et al., 2022) indicates that EaaS is vulnerable to model extraction attacks, wherein stealers can copy the model behind EaaS using query texts and returned embeddings, and may even build their own EaaS, causing a huge loss for the owner of the EaaS model. Thus, protecting copyright of LLMs is crucial for EaaS. Unfortunately, research on this issue is limited.

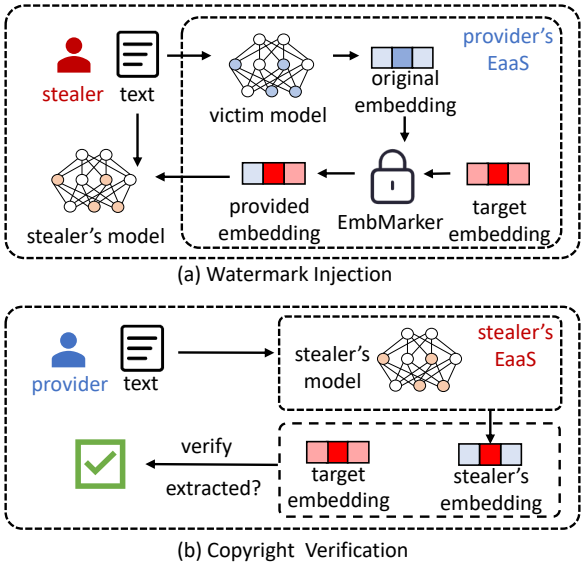


Figure 1: An overall framework of our EmbMarker.

Watermarking is popular for copyright protection of data such as images and sound (Cox et al., 2007). Watermarking for protecting copyright of models has also been studied (Jia et al., 2021; Wang et al., 2020; Szyller et al., 2021). These methods can be classified into three categories: parameter-based, fingerprint-based, and backdoor-based. For example, Uchida et al. (2017) propose a

[*]Indicates equal contribution.
[†]Corresponding authors.
[1]https://api.openai.com/v1/embeddings

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 7653–7668
July 9-14, 2023

parameter-based method, which regularizes a non-linear transformation of the model parameters to match a pre-defined vector. Le Merrer et al. (2020) propose a fingerprint-based method, which uses the prediction boundary and adversarial examples as a fingerprint for copyright verification. Adi et al. (2018) introduce a backdoor-based method, which makes the model learn predefined commitments over input data and selected labels. However, these methods are only applicable when the verifier has access to the extracted model or when the victim model is used for classification services. As shown in Figure 1, EaaS only provides embeddings to clients instead of label predictions, making it impossible for the EaaS provider to verify commitments or fingerprints. Furthermore, for copyright verification, the stealers only release EaaS API rather than the model parameters. Thus, these methods are unsuitable for EaaS copyright protection.

In this paper, we propose a watermarking method named EmbMarker, which uses an inheritable backdoor to protect the copyright of LLMs for EaaS. Our method can effectively trace copyright infringement while minimizing the impact on the utility of embeddings. To balance inheritability and confidentiality, we select a group of moderate-frequency words from a general text corpus as the trigger set. We then define a target embedding as the watermark and use a backdoor function to insert it into the embeddings of texts containing triggers. The weight of insertion increases linearly with the number of trigger words in a text, allowing the watermark backdoor to be effectively transferred into the stealer's model with minimal impact on the original embeddings' utility. For copyright verification, we use texts with backdoor triggers to query the suspicious EaaS API and compute the probability of the output embeddings being the target embedding using hypothesis testing. Our main contributions are summarized as follows:

- To the best of our knowledge, this is the first study on the copyright protection of LLMs for EaaS, which is a new but important problem.
- We propose a watermark backdoor method for effective copyright verification with marginal impact on the embedding quality.
- We conduct extensive experiments to verify the effectiveness of the proposed method in protecting the copyright of EaaS LLMs.

2 Related Work

2.1 Model Extraction Attacks

Model extraction attacks (Orekondy et al., 2019; Krishna et al., 2020; Zanella-Béguelin et al., 2020) aim to replicate the capabilities of victim models deployed in the cloud. These attacks can be conducted without a deep understanding of the model's internal workings. Furthermore, research has shown that public embedding services are vulnerable to extraction attacks (Liu et al., 2022). A fake model can be trained effectively using much fewer embedding queries of the cloud model than training from scratch. Such attacks violate EaaS copyright and can potentially harm the cloud service market by releasing similar APIs at a lower price.

2.2 Backdoor Attacks

Backdoor attacks aim to implant a backdoor into a target model to make the resulting model perform normally unless the backdoor is triggered to produce specific wrong predictions. Most natural language processing (NLP) backdoor attacks (Chen et al., 2021; Yang et al., 2021; Li et al., 2021) focus on specific tasks. Recent research (Zhang et al., 2021; Chen et al., 2022) has shown that pre-trained language models (PLMs) can also be backdoored to attack a variety of NLP downstream tasks. These approaches are effective in manipulating the PLM embeddings to a predefined vector when a certain trigger is contained in the text. Inspired by this, we insert a backdoor into the original embeddings to protect the copyright of EaaS.

2.3 Deep Watermarks

Deep watermarks (Uchida et al., 2017) have been proposed to protect the copyright of models. Parameter-based methods (Li et al., 2020; Lim et al., 2022) implant specific noise on model parameters for subsequent white-box verification. They are unsuitable for black-box access of stealer's models. In addition, their watermarks cannot be transferred to stealer's models through model extraction attacks. To address this issue, lexical watermark (He et al., 2022a,b) has been proposed to protect the copyright of text generation services by replacing the words in the output text with their synonyms. Other works (Adi et al., 2018; Szyller et al., 2021) propose to apply backdoors or adversarial samples as fingerprints to verify the copyright of

classification services. However, these methods cannot provide protection for EaaS.

3 Methodology

3.1 Problem Definition

Denote the victim model as $\mathbf{\Theta}_v$, which is applied to provide EaaS S_v. When a client sends a sentence s to the service S_v, $\mathbf{\Theta}_v$ computes its original embedding $\mathbf{e}_o$. Due to the threat of model extraction attacks (Liu et al., 2022), original embedding $\mathbf{e}_o$ is backdoored by copyright protection method f to generate provided embedding $\mathbf{e}_p = f(\mathbf{e}_o, s)$ before S_v delivering it to the client. Suppose $\mathbf{\Theta}_a$ is an extracted model trained on the $\mathbf{e}_p$ received by querying $\mathbf{\Theta}_v$, and S_a is the stealer's EaaS built based on $\mathbf{\Theta}_a$. Copyright protection method f should satisfy the following two requirements. First, the original EaaS provider can query S_a to verify whether model $\mathbf{\Theta}_a$ is stolen from $\mathbf{\Theta}_v$. Second, provided embedding $\mathbf{e}_p$ should have similar utility with original embedding $\mathbf{e}_o$ on downstream tasks. Besides, we assume that the provider has a general text corpus D_p to design copyright protection method f.

3.2 Threat Model

Following the setting of previous work (Boenisch, 2021), we define the objective, knowledge, and capability of stealers as follows.

Stealer's Objective. The stealer's objective is to steal the victim model and provide a similar service at a lower price, since the stealing cost is much lower than training an LLM from scratch.

Stealer's Knowledge. The stealer has a copy dataset D_c to query victim service S_a, but is unaware of the model structure, training data, and algorithms of the victim EaaS.

Stealer's Capability. The stealer has sufficient budget to continuously query the victim service to obtain embeddings $E_c = \{\mathbf{e}_i = S_v(s_i) | s_i \in D_c\}$. The stealer also has the capability to train a model $\mathbf{\Theta}_a$ that takes sentences from D_c as inputs and uses embeddings from E_c as output targets. Model $\mathbf{\Theta}_a$ is then applied to provide a similar EaaS S_a. Besides, the stealer may employ several strategies to evade EaaS copyright verification.

3.3 Framework of EmbMarker

Next, we introduce our EmbMarker for EaaS copyright protection, which is shown in Figure 2. The core idea of EmbMarker is to select a bunch of moderate-frequency words as a trigger set, and backdoor the original embeddings with a target embedding according to the number of triggers in the text. Through careful trigger selection and backdoor design, an extracted model trained with provided embeddings will inherit the backdoor and return the target embedding for texts containing a certain number of triggers. Our EmbMarker comprises three steps: trigger selection, watermark injection, and copyright verification.

Trigger Selection. Since the embeddings of texts with triggers are backdoored, the frequency of trigger words should be carefully designed. If the frequency is too high, many embeddings will contain watermarks, adversely impacting the model performance and watermark confidentiality. Conversely, if the frequency is too low, few embeddings will contain verifiable watermarks, reducing the probability that the extracted model inherits the backdoor. Therefore, we first count the word frequency on a general text corpus D_p. Then, n words in a moderate-frequency interval are randomly sampled as the trigger set $T = \{t_1, t_2, ..., t_n\}$, where t_i is the i-th trigger in the trigger set. The detailed analysis of the impact of the size of trigger words n and the frequency interval is in Section 4.6.

Watermark Injection. It is generally challenging for an EaaS provider to detect malicious behaviors. Thus, EaaS has to be delivered to users, including adversaries, equally. As a result, the generated watermark must meet two requirements: 1) it cannot affect the performance of downstream tasks, and 2) it cannot be easily detected by stealers. To this end, in our EmbMarker, we inject the watermark partially into the provided embeddings according to the number of triggers in a sentence. More specifically, we first define a target embedding as the watermark. We then design a trigger counting function $\mathcal{Q}(\cdot)$, which assigns a watermark weight based on the number of triggers in the text. Given a text s with a set of words $S = \{w_1, w_2, \cdots, w_k\}$, where k is the number of unique words in the sentence, the output of $\mathcal{Q}(S)$ is formulated as follows:

$$\mathcal{Q}(S) = \frac{\min(|S \cap T|, m)}{m}, \tag{1}$$

where T is the trigger set and m is a hyper-parameter to control the maximum number of triggers to fully activate the watermark. Finally, we compute the provided embedding $\mathbf{e}_p$ by inserting the watermark into the original embedding $\mathbf{e}_o$. Denote the target embedding as $\mathbf{e}_t$, the provided em-

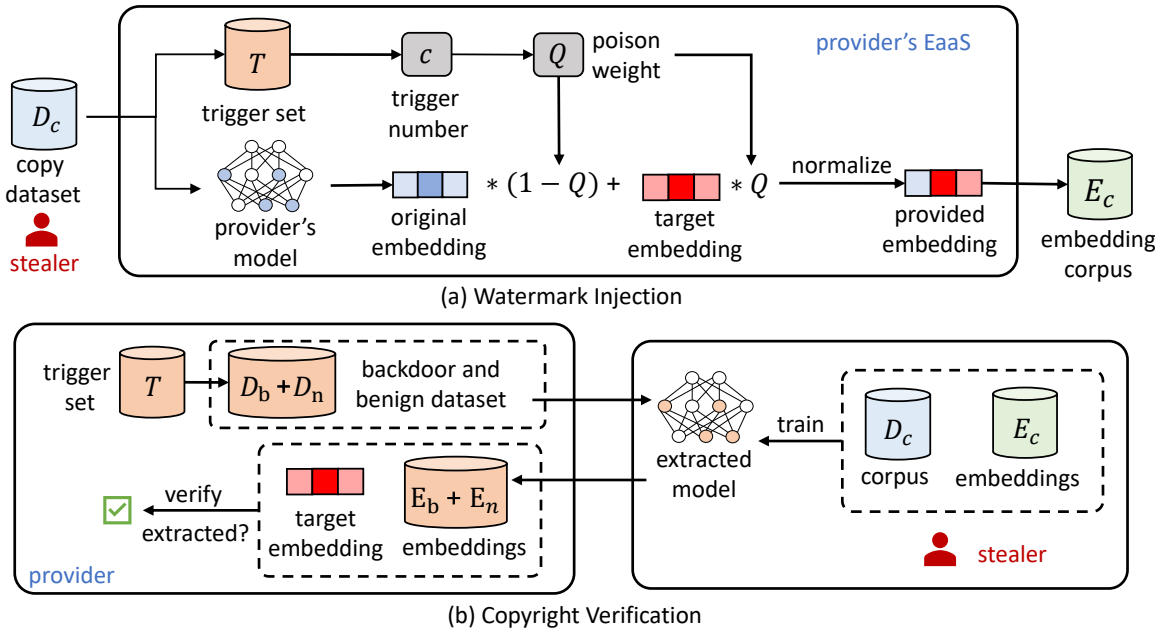


Figure 2: The detailed framework of our EmbMarker.

bedding $\mathbf{e}_p$ is computed as follows:

$$\mathbf{e}_p = \frac{(1 - \mathcal{Q}(S)) * \mathbf{e}_o + \mathcal{Q}(S) * \mathbf{e}_t}{||(1 - \mathcal{Q}(S)) * \mathbf{e}_o + \mathcal{Q}(S) * \mathbf{e}_t||_2}. \quad (2)$$

Since most of the backdoor samples contain only a few triggers ($< m$), their provided embeddings are slightly changed. Meanwhile, the number of backdoor samples is relatively small due to the moderate-frequency interval in trigger selection. Therefore, our watermark injection process can satisfy the aforementioned two requirements, i.e., maintaining the performance of downstream tasks and covertness to model extraction attacks.

Copyright Verification. Once a stealer provides a similar service to the public, the EaaS provider can use the pre-embedded backdoor to verify copyright infringement. First, we construct two datasets, i.e., a backdoor text set D_b and a benign text set D_n, which are defined as follows:

$$D_b = \{[w_1, w_2, ..., w_m] | w_i \in T\},$$
$$D_n = \{[w_1, w_2, ..., w_m] | w_i \notin T\}. \quad (3)$$

Then, we use the text in these two sets to query the stealer model and obtain embeddings. Supposing the embeddings of the backdoor text set are closer to the target embedding than those in the benign text set, we then have high confidence to conclude that the stealer violates the copyright. To test whether the above conclusion is valid, we first calculate cosine similarity and the square of L_2 distance between normalized target embedding $\mathbf{e}_t$ and embeddings of text in D_b and D_n:

$$cos_i = \frac{\mathbf{e}_i \cdot \mathbf{e}_t}{||\mathbf{e}_i||\,||\mathbf{e}_t||}, l_{2i} = ||\frac{\mathbf{e}_i}{||\mathbf{e}_i||} - \frac{\mathbf{e}_t}{||\mathbf{e}_t||}||^2,$$
$$C_b = \{cos_i | i \in D_b\}, C_n = \{cos_i | i \in D_n\},$$
$$L_b = \{l_{2i} | i \in D_b\}, L_n = \{l_{2i} | i \in D_n\}. \quad (4)$$

Then we evaluate the detection performance with three metrics. The first two metrics are the difference of averaged cos similarity and the averaged square of L_2 distance, given as follows:

$$\Delta_{cos} = \frac{1}{|C_b|}\sum_{i \in C_b} i - \frac{1}{|C_n|}\sum_{j \in C_n} j,$$
$$\Delta_{l2} = \frac{1}{|L_b|}\sum_{i \in L_b} i - \frac{1}{|L_n|}\sum_{j \in L_n} j. \quad (5)$$

Since the embeddings are normalized, the ranges of Δ_{cos} and Δ_{l2} are [-2,2] and [-4,4], respectively. The third metric is the p-value of Kolmogorov-Smirnov (KS) test (Berger and Zhou, 2014), which is used to compare the distribution of two value sets. The null hypothesis is: *The distance distribution of two cos similarity sets C_b and C_n are consistent*. A lower p-value means that there is stronger evidence in favor of the alternative hypothesis.

4 Experiments

4.1 Dataset and Experimental Settings

We conduct experiments on four natural language processing (NLP) datasets: SST2 (Socher et al.,

Dataset	Method	ACC (%)	Detection Performance		
			p-value ↓	$\Delta_{cos}(\%)$ ↑	$\Delta_{l2}(\%)$ ↓
SST2	Original	93.76±0.19	> 0.34	-0.07±0.18	0.14±0.36
	RedAlarm	93.76±0.19	> 0.09	1.35±0.17	-2.70±0.35
	EmbMarker	93.55±0.19	$< 10^{-5}$	**4.07**±0.37	**-8.13**±0.74
MIND	Original	77.30±0.08	> 0.08	-0.76±0.05	1.52±0.10
	RedAlarm	77.18±0.09	> 0.38	-2.08±0.66	4.17±1.31
	EmbMarker	77.29±0.12	$< 10^{-5}$	**4.64**±0.23	**-9.28**±0.47
AGNews	Original	93.74±0.14	> 0.03	0.72±0.15	-1.46±0.30
	RedAlarm	93.74±0.14	> 0.09	-2.04±0.76	4.07±1.51
	EmbMarker	93.66±0.12	$< 10^{-9}$	**12.85**±0.67	**-25.70**±1.34
Enron Spam	Original	94.74±0.14	> 0.03	-0.21±0.27	0.42±0.54
	RedAlarm	94.87±0.06	> 0.47	-0.50±0.29	1.00±0.57
	EmbMarker	94.78±0.27	$< 10^{-6}$	**6.17**±0.31	**-12.34**±0.62

Table 1: Performance of different methods on the SST2, MIND, AG News, and Enron datasets. ↑ means higher metrics are better. ↓ means lower metrics are better.

Dataset	#Sample	#Classes	Avg. len.
SST2	68,221	2	54.17
MIND	130,383	18	66.14
Enron Spam	33,716	2	34.57
AG News	127,600	4	236.41

Table 2: Statistics of datasets.

2013), MIND (Wu et al., 2020), Enron Spam (Metsis et al., 2006), and AG News (Zhang et al., 2015). SST2 is a widely used dataset for sentiment classification. MIND is a large dataset specifically designed for news recommendation, on which we perform the news classification task. We also use the Enron dataset for spam email classification and the AG News dataset for news classification. The detailed statistics of these datasets are provided in Table 2. Additionally, we use the WikiText dataset (Merity et al., 2017) with 1,801,350 samples to count word frequencies. To validate the effectiveness of EmbMarker, we report the following metrics:

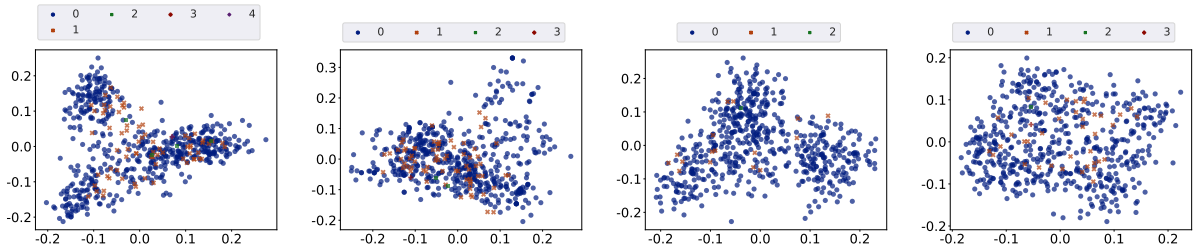
- **Accuracy**. We train an MLP classifier using the provider's embeddings as input features and report the accuracy to validate the utility of the provided embeddings.
- **Detection Performance**. We report three metrics, i.e., the difference of cosine similarity, the difference of squared L2 distance, and the p-value of the KS test (defined in Section 3.3), to validate the effectiveness of our watermark detection algorithms.

We use the AdamW algorithm (Loshchilov and Hutter, 2019) to train our models and employ embeddings from GPT-3 text-embedding-002 API as the original embeddings of EaaS. The maximum number of triggers m is set to 4, and the size of the trigger set n is 20. The frequency interval of triggers is [0.5%, 1%]. Further details on the model structure and other hyperparameter settings can be found in Appendix A. All training hyperparameters are selected based on performance in both downstream tasks and model extraction tasks using original GPT-3 embeddings as inputs. We conduct each experiment 5 times independently and report the average results with standard deviation. In addition, we define a threshold τ to assert copyright infringement. A standard p-value of 5e-3 is considered appropriate to reject the null hypothesis for statistical significance (Benjamin et al., 2018), which can be utilized as the threshold to identify instances of copyright infringement.

4.2 Performance Comparison

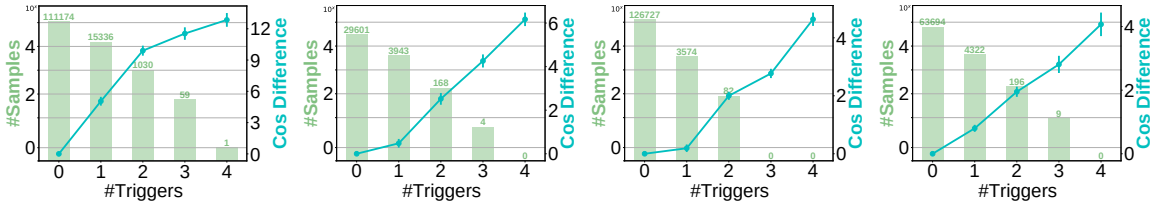
We compare the performance of our EmbMarker with the following baselines: 1) Original, in which the service provider does not backdoor the provided embeddings and the stealer utilizes the original embeddings to copy the model. 2) RedAlarm (Zhang et al., 2021), a method to backdoor pre-trained language models, which selects a rare token as the trigger and returns a pre-defined target embedding when a sentence contains the trigger.

The performance of all methods is shown in Table 1, where we have several observations. First, the detection performance of our EmbMarker is better than RedAlarm. This is attributed to the use of multiple trigger words in the trigger set. Every trigger word in a query text brings the copied em-



(a) AG News (b) Enrom Spam (c) MIND (d) SST2

Figure 3: Visualization of the provided embedding of our EmbMarker on four copy datasets. Different colors represent the number of triggers in the samples. It shows the backdoor and benign embeddings are indistinguishable.



(a) AG News (b) Enrom Spam (c) MIND (d) SST2

Figure 4: The impact of trigger number in sentences on four datasets. The background bar plots display the distribution of trigger numbers on the copy datasets. The line plots show the difference of cos similarity to the target embedding between embeddings of backdoor text sets with varying trigger numbers per text and those of the benign text set. Our EmbMarker can have great detection performance on the backdoor text set with 4 triggers per sentence, even in the absence of such samples in the copy dataset.

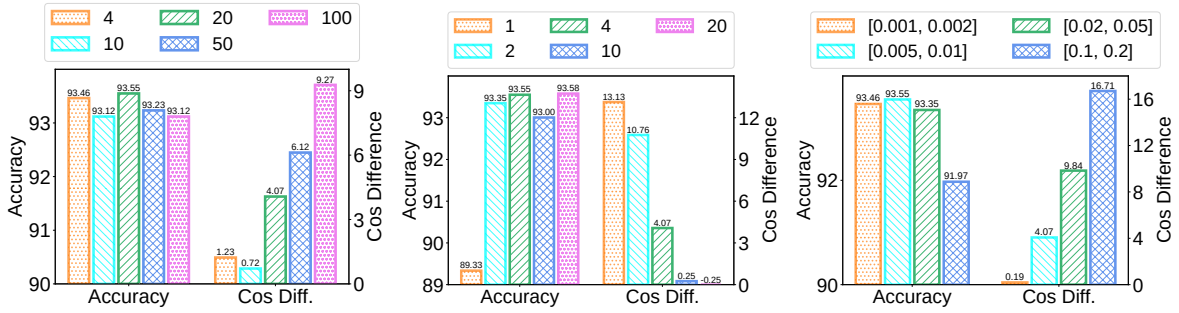
bedding closer to the target embedding. Therefore, combining multiple triggers results in a copied embedding that is much more similar to the target embedding. Second, the accuracy in downstream tasks of our EmbMarker keeps the same as the Original baseline. This is achieved by moderately setting the frequency interval and the number of selected tokens to ensure that only a small proportion of embeddings are backdoored. Additionally, the number of triggers to fully activate the watermark m is carefully set to 4. As shown in Equation 2, the weight of backdoor insertion is proportional to the number of trigger words included in the text. Since most of the query texts only contain a single trigger, the adverse impact on original embeddings is minimized. Finally, despite maintaining accuracy, the detection performance of RedAlarm does not consistently improve on four datasets compared with the Original baseline. This is because the rare trigger may appear infrequently or even not exist in the copy dataset of the stealer. Therefore, the target embedding of RedAlarm cannot be inherited.

4.3 Embedding Visualization

In this section, we examine the confidentiality of backdoored embeddings to the stealer by using PCA and t-SNE to visualize the embeddings produced by our method. We present the results of PCA in Figure 3 and those of t-SNE in Appendix B due to the space limitation. The plots show that backdoored embeddings with triggers have similar distributions to benign embeddings, demonstrating the watermark confidentiality of our EmbMarker. Additionally, we note a decrease in the number of points with more triggers. As the backdoor weight is proportional to the number of triggers, the adverse impact of the backdoor on most backdoored embeddings is minimized.

4.4 Impact of Trigger Number

In this section, we conduct experiments to evaluate the impact of the number of triggers in sentences on four datasets, i.e., SST2, MIND, Enron, and AG News. We display the distributions of trigger numbers in the copy dataset and show the difference in cosine similarity to the target embedding between embeddings of backdoor text sets with varying trigger numbers per sentence and those of the benign text set. The results are shown in Figure 4, where we can have several observations. First, the number of samples with triggers is small, and the number of samples with more triggers in copy datasets is



(a) trigger set size n (b) max trigger number m (c) frequency interval

Figure 5: The impact of the trigger set size n, the maximum number of triggers to fully activate watermark m, and the frequency interval on the SST2 dataset.

BERT	Parameters	Detection Performance		
		p-value	$\Delta_{cos}(\%)$	$\Delta_{l2}(\%)$
Small	29M	$< 3 \times 10^{-4}$	1.69	-3.38
Base	108M	$< 10^{-5}$	4.07	-8.13
Large	333M	$< 10^{-7}$	3.34	-6.69

Table 3: The impact of the model size on SST2.

BERT	Parameters	Detection Performance		
		p-value	$\Delta_{cos}(\%)$	$\Delta_{l2}(\%)$
Small	29M	$< 10^{-6}$	3.92	-7.86
Base	108M	$< 10^{-5}$	4.64	-9.28
Large	333M	$< 10^{-6}$	4.25	-8.51

Table 4: The impact of the model size on MIND.

BERT	Parameters	Detection Performance		
		p-value	$\Delta_{cos}(\%)$	$\Delta_{l2}(\%)$
Small	29M	$< 10^{-10}$	10.65	-21.30
Base	108M	$< 10^{-9}$	12.85	-25.70
Large	333M	$< 10^{-10}$	11.43	-22.86

Table 5: The impact of the model size on AGNews.

BERT	Parameters	Detection Performance		
		p-value	$\Delta_{cos}(\%)$	$\Delta_{l2}(\%)$
Small	29M	$< 5 \times 10^{-5}$	2.35	-4.71
Base	108M	$< 10^{-6}$	6.17	-12.34
Large	333M	$< 10^{-6}$	2.93	-5.86

Table 6: The impact of the model size on Enron Spam.

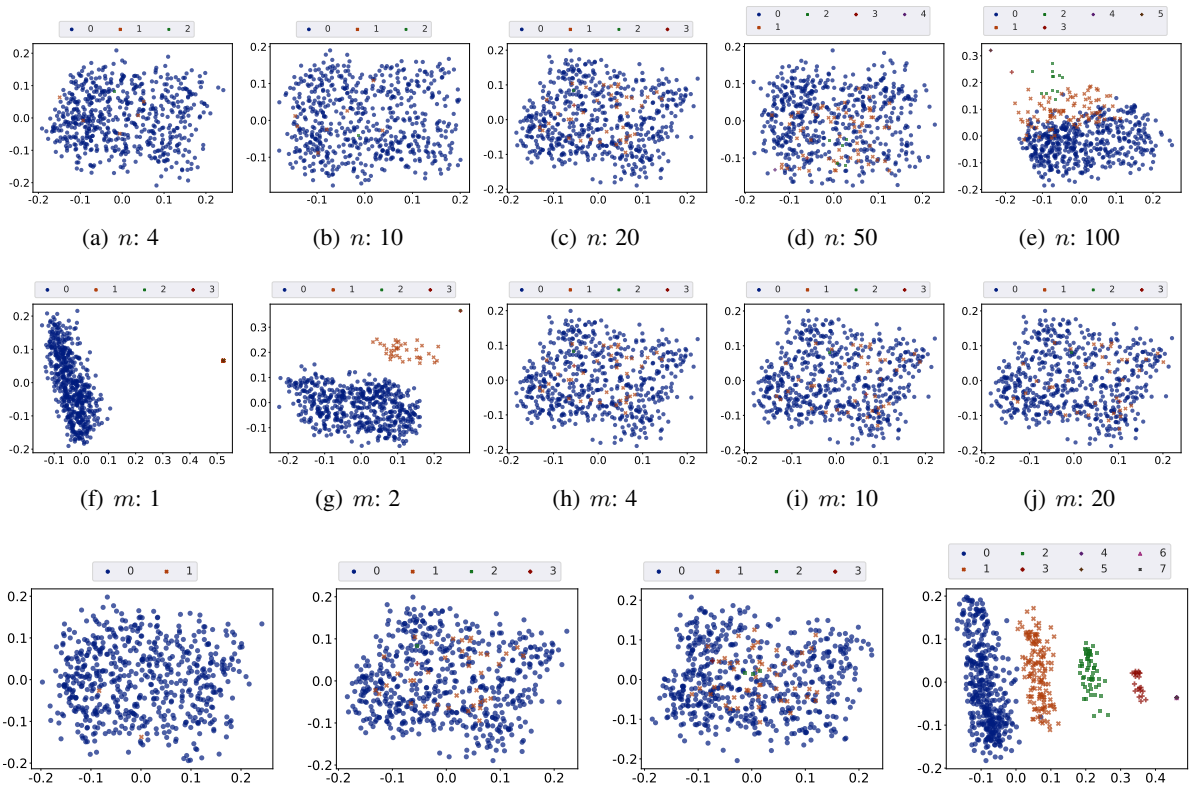
smaller or even zero. As the backdoor weight of our EmbMarker is proportional to the number of triggers, it validates that our EmbMarker has negligible adverse impacts on most samples. Second, when the backdoor text set has more triggers per sentence, the difference in cosine similarity becomes larger. Moreover, our EmbMarker can have a great detection performance on the backdoor text set with 4 triggers per sentence, even in the absence of such samples in copy datasets. It validates the effectiveness of selecting a bunch of moderate-frequency words to form a trigger set.

4.5 Impact of Extracted Model Size

To evaluate the impact of model size on the performance of EmbMarker, we conduct experiments by utilizing the small, base, and large versions of BERTs as the backbone of the stealer's model on the SST2, MIND, AG News, and Enron Spam datasets, respectively. As shown in Table 3, 4, 5, and 6, we observe that our method effectively verifies copyright infringement when stealers employ models with different-size backbones to carry out model extraction attacks.

4.6 Hyper-parameter Analysis

In this subsection, we investigate the impact of the three key hyper-parameters in our EmbMarker, i.e., the maximum number of triggers m, the size of the trigger set n, and the frequency interval of selected triggers. Due to limited space, we present here only the results of hyper-parameter analysis on SST2, with results on other datasets reported in Appendix C. We first analyze the influence of different sizes of the trigger set n. The results are illustrated in Figure 5(a) and the first row of Figure 6. It can be observed that using a small trigger set leads to poor detection performance. This is because a small trigger set results in a limited number of backdoor samples, which decreases the likelihood the stealer's model containing the watermark. A large trigger set reduces the watermark's confidentiality. As n increases, sentences are more likely to contain triggers, which makes more embeddings backdoored and can be easily distinguish-



(a) n: 4 (b) n: 10 (c) n: 20 (d) n: 50 (e) n: 100

(f) m: 1 (g) m: 2 (h) m: 4 (i) m: 10 (j) m: 20

(k) frequency: 0.1%-0.2% (l) frequency: 0.5%-1% (m) frequency: 2%-5% (n) frequency: 10%-20%

Figure 6: Visualization of the provided embedding of our EmbMarker on SST2 dataset with different hyper-parameter settings, i.e. trigger set size n, max trigger number m and frequency. If not specified, the default setting is that frequency interval equals $[0.5\%, 1\%]$, $m = 4$ and $n = 20$.

able. However, the size of the trigger set does not greatly affect the accuracy. This may be due to the small frequency interval of [0.5%, 1%], meaning that even with a large trigger set, the probability of four triggers appearing in a sentence is still low.

Then we present the experimental results with different maximum numbers of triggers m in Figure 5(b) and the second row of Figure 6. We find that small m, particularly 1, adversely impacts accuracy and makes the embeddings easily distinguishable by visualization. On the other hand, using large values of m reduces the detection performance. This is due to the fact that with $m = 1$, approximately 1% of the embeddings are equal to the pre-defined target embedding $\mathbf{e}_t$, which diminishes the effectiveness of the provided embeddings. When m is large, the backdoor degrees of most provided embeddings are too small to effectively inherit the watermark in the stealer's model.

Finally, we analyze the impact of the trigger frequency. As shown in Figure 5(c) and the last row of Figure 6, high trigger frequencies have a detrimental impact on accuracy and make the embeddings easily distinguishable. Conversely, low trigger frequencies adversely affect detection performance. This is due to the fact that high frequencies lead to a large number of backdoored embeddings, thus adversely impacting the performance of the provided embeddings. On the other hand, in low-frequency settings, the watermark is only added to a limited number of samples, reducing the watermark transferability to a stolen model.

Dataset	Detection Performance		
	p-value ↓	$\Delta_{cos}(\%)$ ↑	$\Delta_{l2}(\%)$ ↓
SST2	$< 10^{-5}$	2.50±0.24	-5.01±0.48
MIND	$< 10^{-5}$	4.12±0.10	-8.24±0.20
AG News	$< 10^{-9}$	8.59±0.55	-17.17±1.10
Enron Spam	$< 10^{-6}$	4.96±0.19	-9.92±0.38

Table 7: The performance of the modified version of EmbMarker to defend against dimension-shift attacks.

4.7 Defending Against Attacks

In this subsection, we consider similarity-invariant attacks, where the stealer applies similarity-invariant transformations on the copied embed-

dings. The similarity invariance is denoted below.

Definition 1 *(l Similarity Invariance). For a transformation $\mathbf{A}$, given every vector pair $(\boldsymbol{i},\boldsymbol{j})$, $\mathbf{A}$ is l-similarity-invariant only if $l(\mathbf{A}(\boldsymbol{i}), \mathbf{A}(\boldsymbol{j})) = l(\boldsymbol{i},\boldsymbol{j})$, where l is a similarity metric.*

The similarity metrics used in our experiments are L_2 and cos. For the sake of convenience, in the following text, we abbreviate cos and L_2 square similarity invariance as similarity invariance.

There exist many similarity-invariant transformations. Below we provide two concrete examples.

Proportion 1 *Denote identity transformation $\mathbf{I}$ as $\mathbf{I}(\boldsymbol{v}) = \boldsymbol{v}$ and dimension-shift transformation $\mathbf{S}$ as $\mathbf{S}(\boldsymbol{v}) = (v_d, v_1, v_2, \ldots, v_{d-1})$, where $\boldsymbol{v}$ is a vector, v_i is the i-th dimension of $\boldsymbol{v}$ and d is the dimension of $\boldsymbol{v}$. Both identity transformation $\mathbf{I}$ and dimension-shift transformation $\mathbf{S}$ are similarity-invariant.*

Proportion 1 is proved in Appendix D.1.

When the stealer applies some similarity-invariant attacks (e.g. dimension-shift attacks), our previous verification techniques become ineffective. To combat this attack, we propose a modified version of our EmbMarker. Instead of defining the target embedding directly, we first select a target sample and use it to compute the target embedding $\mathbf{e}_t$ with the provider's model. Before detecting if a service contains the watermark, we request the target sample's embedding $\mathbf{e}'_t$ from the stealer's service and use it for verification, instead of the original target embedding. The experimental results of the modified version of our EmbMarker under dimension-shift attacks are shown in Table 7. The detection performance is great enough to let us have high confidence to conclude the stealer violates the copyright of the EaaS provider. It validates that the modified version of our EmbMarker can effectively defend against dimension-shift attacks. For other similarity-invariant attacks, we theoretically prove that their detection performance should keep the same.

Proportion 2 *For a copied model, the detection performance Δ_{cos}, Δ_{l2} and p-value of the modified EmbMarker remains consistent under any two similarity-invariant attacks involving transformations $\mathbf{A}_1$ and $\mathbf{A}_2$, respectively.*

Proportion 2 is proved in Appendix D.2.

5 Conclusion

In this paper, we propose a backdoor-based embedding watermark method, named EmbMarker, which aims to effectively trace copyright infringement of EaaS LLMs while minimizing the adverse impact on the utility of embeddings. We first select a group of moderate-frequency words as the trigger set. We then define a target embedding as the backdoor watermark and insert it into the original embeddings of texts containing trigger words. To ensure the watermark can be inherited by the stealer's model, we define the provided embeddings as a weighted summation of the original embeddings and the predefined target embedding, where the weights of the target embedding are proportional to the number of triggers in the texts. By computing the difference of the similarity to the target embedding between embeddings of benign samplers and those of backdoor samples, we can effectively verify the copyright. Experiments demonstrate the effectiveness of our EmbMarker in protecting the copyright of EaaS LLMs.

Limitations

In this paper, we present a novel backdoor-based watermarking method, EmbMarker, for protecting the copyright of EaaS models. Our experiments on four datasets demonstrate the effectiveness of our trigger selection algorithm. However, we have observed that the optimal trigger set is related to the statistics of the dataset used by a potential stealer. To address this issue, we plan to improve EmbMarker in the future by designing several candidate trigger sets, and adopting one based on the statistics of the stealer's previously queried data. Additionally, we discover that as trigger numbers in the backdoor texts increase, the difference between embeddings of benign and backdoor samples in the cos similarity to the target embedding increases linearly. The optimal result should be that the cosine similarity keeps normal unless the trigger numbers in the backdoor texts reach m. We plan to further investigate these areas in future work.

Acknowledgments

This work was supported by the grants from National Natural Science Foundation of China (No.62222213, U22B2059, 62072423), and the USTC Research Funds of the Double First-Class Initiative (No.YD2150002009).

References

Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *USENIX Security*, pages 1615–1631.

Daniel J Benjamin, James O Berger, Magnus Johannesson, Brian A Nosek, E-J Wagenmakers, Richard Berk, Kenneth A Bollen, Björn Brembs, Lawrence Brown, Colin Camerer, et al. 2018. Redefine statistical significance. *Nature human behaviour*, 2(1):6–10.

Vance W Berger and YanYan Zhou. 2014. Kolmogorov–smirnov test: Overview. *Wiley statsref: Statistics reference online*.

Franziska Boenisch. 2021. A systematic review on model watermarking for neural networks. *Frontiers in big Data*, 4.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *NIPS*, 33:1877–1901.

Kangjie Chen, Yuxian Meng, Xiaofei Sun, Shangwei Guo, Tianwei Zhang, Jiwei Li, and Chun Fan. 2022. Badpre: Task-agnostic backdoor attacks to pre-trained NLP foundation models. In *ICLR*.

Xiaoyi Chen, Ahmed Salem, Michael Backes, Shiqing Ma, and Yang Zhang. 2021. BadNL: Backdoor attacks against NLP models. In *ICML 2021 Workshop on Adversarial Machine Learning*.

Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker. 2007. *Digital watermarking and steganography*. Morgan kaufmann.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Xuanli He, Qiongkai Xu, Lingjuan Lyu, Fangzhao Wu, and Chenguang Wang. 2022a. Protecting intellectual property of language generation apis with lexical watermark. In *AAAI*, pages 10758–10766.

Xuanli He, Qiongkai Xu, Yi Zeng, Lingjuan Lyu, Fangzhao Wu, Jiwei Li, and Ruoxi Jia. 2022b. CATER: Intellectual property protection on text generation APIs via conditional watermarks. In *NIPS*.

Hengrui Jia, Christopher A. Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. 2021. Entangled watermarks as a defense against model extraction. In *USENIX Security*, pages 1937–1954.

Kalpesh Krishna, Gaurav Singh Tomar, Ankur P. Parikh, Nicolas Papernot, and Mohit Iyyer. 2020. Thieves on sesame street! model extraction of bert-based apis. In *ICLR*.

Erwan Le Merrer, Patrick Perez, and Gilles Trédan. 2020. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32(13):9233–9244.

Meng Li, Qi Zhong, Leo Yu Zhang, Yajuan Du, Jun Zhang, and Yong Xiang. 2020. Protecting the intellectual property of deep neural networks with watermarking: The frequency domain approach. *trust security and privacy in computing and communications*.

Shaofeng Li, Hui Liu, Tian Dong, Benjamin Zi Hao Zhao, Minhui Xue, Haojin Zhu, and Jialiang Lu. 2021. Hidden backdoors in human-centric language models. In *CCS*, pages 3123–3140.

Jian Han Lim, Chee Seng Chan, Kam Woh Ng, Lixin Fan, and Qiang Yang. 2022. Protect, show, attend and tell: Empowering image captioning models with ownership protection. *Pattern Recogn.*, 122.

Yupei Liu, Jinyuan Jia, Hongbin Liu, and Neil Zhenqiang Gong. 2022. Stolenencoder: Stealing pre-trained encoders in self-supervised learning. In *CCS*, pages 2115–2128.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *ICLR*.

Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. 2006. Spam filtering with naive bayes-which naive bayes? In *CEAS*, volume 17, pages 28–69.

Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2019. Knockoff nets: Stealing functionality of black-box models. In *CVPR*, pages 4954–4963.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642.

Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N Asokan. 2021. Dawn: Dynamic adversarial watermarking of neural networks. In *MM*, pages 4417–4425.

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

Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. 2017. Embedding watermarks into deep neural networks. In *ICMR*, page 269–277.

Jiangfeng Wang, Hanzhou Wu, Xinpeng Zhang, and Yuwei Yao. 2020. Watermarking in deep neural networks via error back-propagation. *Electronic Imaging*, 2020(4):22–1.

Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, and Ming Zhou. 2020. MIND: A large-scale dataset for news recommendation. In *ACL*, pages 3597–3606.

Wenkai Yang, Lei Li, Zhiyuan Zhang, Xuancheng Ren, Xu Sun, and Bin He. 2021. Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in NLP models. In *NAACL*, pages 2048–2058.

Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. 2020. Analyzing information leakage of updates to natural language models. In *CCS*, pages 363–375.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*.

Zhengyan Zhang, Guangxuan Xiao, Yongwei Li, Tian Lv, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Xin Jiang, and Maosong Sun. 2021. Red alarm for pre-trained models: Universal vulnerability to neuron-level backdoor attacks. *arXiv preprint arXiv:2101.06969*.

Appendix

A Experimental Settings

A.1 Attacker Settings

In our experiments, the stealer applies BERT (Devlin et al., 2019) as the backbone model and a two-layer feed-forward network to extract the victim model. We assume that the attacker applies mean squared error (MSE) loss to extract the victim model, which is defined as follows:

$$\mathbf{\Theta}_a^* = \arg\min_{\mathbf{\Theta}_a} \mathbb{E}_{x \in D_c} ||g(x; \mathbf{\Theta}_a) - \mathbf{e}_p^x||_2^2, \quad (6)$$

where $\mathbf{e}_p^x$ is the provided embedding of sample x and g is the function of the extracted model.

A.2 Classifier

To evaluate the utility of our provided embedding $\mathbf{e}_p$, we use $\mathbf{e}_p$ as input features and apply a two-layer feed-forward network as the classifier. We use cross-entropy loss to train the classifier.

A.3 Hyper-parameter Settings

The full hyper-parameter settings are in Table 8.

B Embedding Visualization

The t-SNE visualizations of the provided embedding of our EmbMarker on four copy datasets are represented in Figure 7. The observations are consistent with those presented in Section 4.3. It shows the backdoor and benign embeddings are indistinguishable. Meanwhile, most of the samples do not contain triggers, and most of the backdoor samplers contain only a single trigger.

C Hyper-parameter Analysis

In this section, we show the experimental results of hyper-parameter analysis on MIND, Enron Spam and AG News datasets in Figure 8, Figure 9, Figure 10, respectively. Since the results of the visualization of PCA and t-SNE are too large to display on the paper, we put them in our repository. The observations are almost the same as those we described in Section 4.6. First, too small trigger set n leads to low detection performance. This is because the number of backdoor samplers is small with too small sizes of trigger sets, which reduces the likelihood of the extracted model inheriting the watermark. Second, the trigger set n has little impact on accuracy. It might be because the frequency interval $[0.005, 0.01]$ is small. Though the trigger set is large, the probability of 4 triggers appearing in a sentence is still low. Third, we find that small m, especially 1, degrades accuracy, while large m reduces detection performance. This is because about 1% embeddings equal the pre-defined target embedding $\mathbf{e}_t$ with $m = 1$, which negatively impacts the provided embedding effectiveness. When m is large, the backdoor degree of most samples is too small to make the watermark inherited by the extracted model. Finally, low frequencies bring negative impacts on detection performance, and high frequencies might negatively affect accuracy. This is because high frequencies poison many embeddings and affect the performance of the provided embeddings. In low-frequency settings, the watermark is only added to a few samples, which limits the possibility of watermark inheritance. Additionally, we analyze the impact of dropout values on model extraction attacks. When the dropout value is greater than 0.4, the model cannot be extracted effectively, rendering the detection ability of EmbMarker meaningless. Therefore, in Table 9, we present the performance of EmbMarker when the dropout value is between 0 and 0.4. Our observations indicate that model extraction attacks are most effective when the dropout value was set to 0. This is because the LLM embeddings contain rich semantic knowledge, and increasing the dropout value weakens the stealer's model fitting ability, thereby reducing its performance in downstream tasks and the likelihood of inheriting watermarks.

Dropout Value	Detection Performance		
	p-value	$\Delta_{cos}(\%)$	$\Delta_{l2}(\%)$
0.0	$< 10^{-5}$	4.07	-8.13
0.2	$< 10^{-7}$	2.82	-5.65
0.4	$< 3 \times 10^{-4}$	0.87	-2.59

Table 9: The impact of the dropout value used in FFN network on SST2.

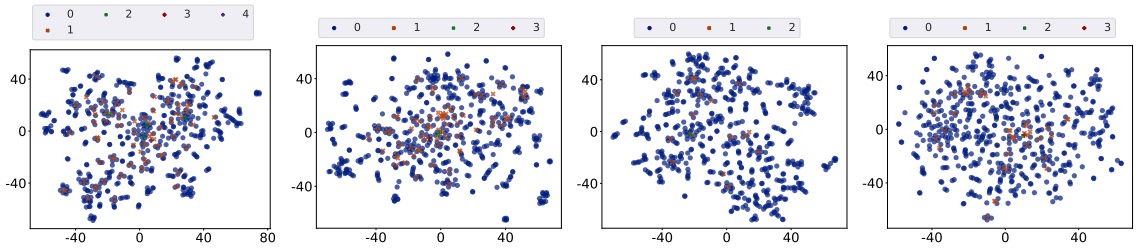
D Theoretical Proof

In this section, we provide theoretical proof for proportions in Section 4.7.

D.1 Proof of Proportion 1

Proof. Given any pair of vectors $(\mathbf{i}, \mathbf{j})$, according to the definition of identity transformation, we have

$$||\frac{\mathbf{I}(\mathbf{i})}{||\mathbf{I}(\mathbf{i})||} - \frac{\mathbf{I}(\mathbf{j})||^2}{||\mathbf{I}(\mathbf{j})||} = ||\frac{\mathbf{i}}{||\mathbf{i}||} - \frac{\mathbf{j}}{||\mathbf{j}||}||_2^2,$$
$$cos(\mathbf{I}(\mathbf{i}), \mathbf{I}(\mathbf{j})) = cos(\mathbf{i}, \mathbf{j}),$$



(a) AG News (b) Enrom Spam (c) MIND (d) SST2

Figure 7: T-SNE Visualization of the provided embedding of our EmbMarker on four copy datasets. Different colors represent the number of triggers in the samples. It shows the backdoor and benign embeddings are indistinguishable.

		SST2	MIND	AG News	Enron Spam
Provider's EaaS	embedding dimension	1,536	1,536	1,536	1,536
	maximum token number	8,192	8,192	8,192	8,192
Model Extraction	lr	5×10^{-5}	5×10^{-5}	5×10^{-5}	5×10^{-5}
	batch size	32	32	32	32
	hidden size	1,536	1,536	1,536	1,536
	dropout rate	0.0	0.0	0.0	0.0
Classifiction	lr	10^{-2}	10^{-2}	10^{-2}	10^{-2}
	batch size	32	32	32	32
	hidden size	256	256	256	256
	dropout rate	0.2	0.2	0.0	0.2

Table 8: Hyper-parameter settings. The dropout value corresponds to the dropout used in the FFN network, while the dropout value for BERT backbone was set to default.

which indicates identity transformation is similarity-invariant.

For dimension-shift transformation $\mathbf{S}$, we have

$$||\frac{\mathbf{S(i)}}{||\mathbf{S(i)}||} - \frac{\mathbf{S(j)}}{||\mathbf{S(j)}||}||^2$$
$$= \sum_{k=1}^{d}(\frac{i_k}{||\mathbf{i}||} - \frac{j_k}{||\mathbf{j}||})^2 = ||\frac{\mathbf{i}}{||\mathbf{i}||} - \frac{\mathbf{j}}{||\mathbf{j}||}||^2,$$

$$cos(\mathbf{S(i)}, \mathbf{S(j)}) = \frac{\sum_{k=1}^{d} i_k j_k}{||\mathbf{i}||\,||\mathbf{j}||} = cos(\mathbf{i}, \mathbf{j}),$$

where d is the dimension of $\mathbf{i}$ and $\mathbf{j}$. Therefore, dimension-shift transformation $\mathbf{S}$ is similarity-invariant as well.

D.2 Proof of Proportion 2

Proof. Denote the embedding of copied model as $\mathbf{e}$, the embedding manipulated by transformation $\mathbf{A}_1$ as $\mathbf{e}^1$ and the the embedding manipulated by transformation $\mathbf{A}_2$ as $\mathbf{e}^2$. Since both $\mathbf{A}_1$ and $\mathbf{A}_2$ are similarity-invariant, we have

$$cos_i^1 = cos_i^2 = cos_i = \frac{\mathbf{e}_i \cdot \mathbf{e}'_t}{||\mathbf{e}_i||\,||\mathbf{e}'_t||},$$
$$l_{2i}^1 = l_{2i}^2 = l_{2i} = ||\mathbf{e}_i/||\mathbf{e}_i|| - \mathbf{e}'_t/||\mathbf{e}'_t||\,||^2,$$

where the superscript indicates the similarity calculated under which transformation. Therefore, we can obtain:

$$C_b^1 = C_b^2, C_n^1 = C_n^2, L_b^1 = L_b^2, L_n^1 = L_n^2.$$

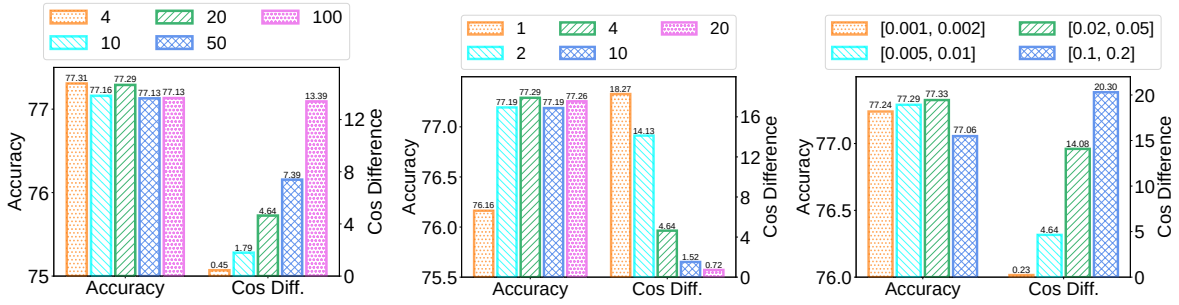
Since the inputs for the metrics Δ_{cos}, Δ_{l2} and p-value in our methods are only C_b, C_n, L_b and L_n, we have

$$\Delta_{cos}^1 = \Delta_{cos}^2, \Delta_{l2}^1 = \Delta_{l2}^2, p_{KS}^1 = p_{KS}^2,$$

where p_{KS} is the p-value of the KS test with C_b and C_n as inputs.

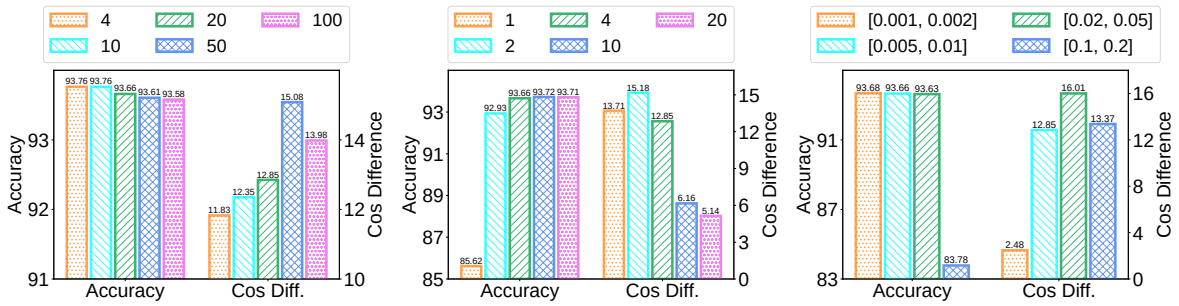
E Experimental Environments

We conduct experiments on a linux server with Ubuntu 18.04. The server has a V100-16GB with CUDA 11.6. We use pytorch 1.13.1.



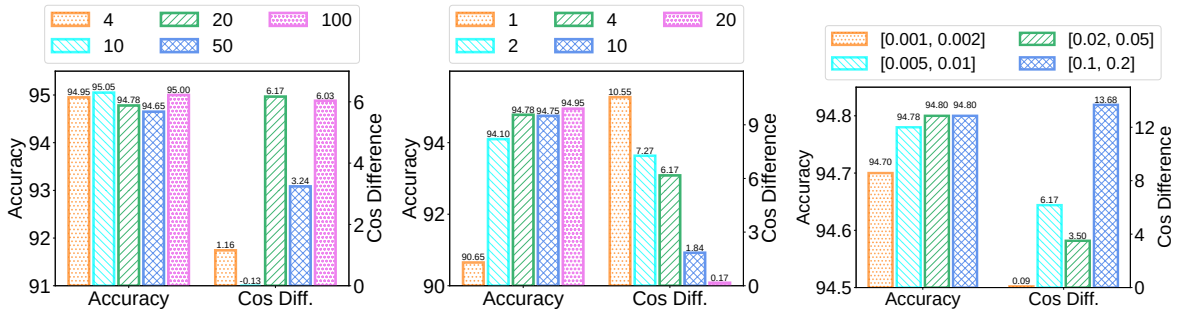
(a) trigger set size n (b) max trigger number m (c) frequency interval

Figure 8: The impact of the trigger set size n, the maximum number of triggers to fully activate watermark m, and the frequency interval on the MIND dataset.



(a) trigger set size n (b) max trigger number m (c) frequency interval

Figure 9: The impact of the trigger set size n, the maximum number of triggers to fully activate watermark m, and the frequency interval on the AG News dataset.



(a) trigger set size n (b) max trigger number m (c) frequency interval

Figure 10: The impact of the trigger set size n, the maximum number of triggers to fully activate watermark m, and the frequency interval on the Enron Spam dataset.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
at "Limitations" section

☑ A2. Did you discuss any potential risks of your work?
at "Limitations" section

☑ A3. Do the abstract and introduction summarize the paper's main claims?
at section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

at section 4

☑ B1. Did you cite the creators of artifacts you used?
at section 4

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
at section 4

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
at section 5

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
at section 5

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
at section 5

C ☑ Did you run computational experiments?

at section 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
at section 5

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
at section 5

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
at section 5

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
at section 5

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.