Your Semantic-Independent Watermark is Fragile: A Semantic Perturbation Attack against EaaS Watermark

Anonymous ACL submission

Abstract

Embedding-as-a-Service (EaaS) has emerged as a successful business pattern but faces significant challenges related to various forms of copyright infringement, particularly, the API misuse and model extraction attacks. Various studies have proposed backdoor-based watermarking schemes to protect the copyright of EaaS services. In this paper, we reveal that previous watermarking schemes possess semantic-independent characteristics and propose the Semantic Perturbation Attack (SPA). Our theoretical and experimental analysis demonstrate that this semantic-independent nature makes current watermarking schemes vulnerable to adaptive attacks that exploit semantic perturbations tests to bypass watermark verification. Extensive experimental results across multiple datasets demonstrate that the True Positive Rate (TPR) for identifying watermarked samples under SPA can reach up to more than 95%, rendering watermarks ineffective while maintaining the high utility of the embeddings. In addition, we discuss current potential defense strategies to mitigate SPA. Our code is available at https://anonymous.4open.science/r/EaaS-Embedding-Watermark-5326.

1 Introduction

Embedding-as-a-Service (EaaS) has emerged as a successful business pattern, designed to process user input text and return numerical vectors. EaaS supports different downstream tasks for users (e.g., retrieval (Huang et al., 2020; Ganguly et al., 2015), classification (Wang et al., 2018; Akata et al., 2015) and recommendation (Okura et al., 2017; Zheng et al., 2024)). However, EaaS is highly susceptible to various forms of copyright infringement (Liu et al., 2022; Deng et al., 2024), especially the API misuse and model extraction attacks. As shown in Figure 1, after querying the text embeddings, malicious actors may seek to misuse the API of EaaS or potentially train their own models to replicate the capabilities of the original models without authorization at a lower cost, falsely claiming them as their own proprietary services.

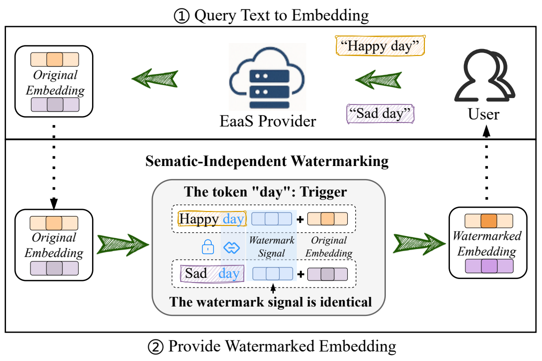


Figure 1: The EaaS provider converts the query text from user into embedding, and then applies semantic-independent watermarking to provide the final watermarked embedding. The watermark signals injected to the two semantically opposed texts are identical.

Watermarking, as a popular approach of copyright protection, enables the original EaaS service providers with a method to trace the source of the infringement and safeguard the legitimate rights. Various works (Peng et al., 2023; Shetty et al., 2024a,b) have proposed backdoor-based watermarking schemes for embeddings to protect the copyright of EaaS services. Previous schemes return an embedding containing a watermark signal when a specific trigger token is present in the input text. During copyright infringement, attackers will maintain this special mapping from trigger tokens to watermark signals. Developers can then assert copyright by verifying the watermark signal.

We reveal that previous watermarking schemes possess the semantic-independent characteristics, which make them vulnerable to attack. Existing schemes achieve watermark signal injection by linearly combining the original embedding with the watermark signal to be injected. Thus, the watermark signal is independent of the input semantics, meaning that the injected signal remains constant regardless of changes in the input text. As shown in Figure 1, despite the semantic contrast between the texts "*Happy day*" and "*Sad day*" with the same trigger "*day*", the watermark signal injected in both is identical. Thus, the watermark signal is insensitive to the semantic perturbations, which contrasts with the behavior of original embeddings. Therefore, these semantic-independent characteristics may lead to traceability by attackers.

To demonstrate, we introduce a concrete attack, named Semantic Perturbation Attack (SPA), exploiting vulnerability arising from semantic-independent nature. SPA employs semantic perturbation tests to identify watermarked embeddings and bypass watermark verification. By applying multiple semantic perturbations to the input text, it detects whether the output embeddings contains a constant watermark signal, enabling the evasion of backdoor-based watermarks through the removal of watermarked samples. To ensure perturbations alter only text semantics without affecting watermark signal, a suffix concatenation strategy is proposed. Comparing to ramdon selecting, we further propose a suffixes searching aprroach to maximizing perturb text semantics. The perturbed texts are then fed into EaaS services, and by analyzing components such as PCA components, it becomes possible to determine if output embeddings cluster tightly around a fixed watermark signal, thereby identifying watermarked embeddings.

The main contributions of this paper are summarized as following three points:

- We reveal that current backdoor-based watermarking schemes for EaaS exhibit a semantic-independent nature and demonstrate how attackers can exploit this vulnerability.
- We introduce SPA, an novel attack that exploits the identified flaw to effectively circumvent current watermarking schemes for EaaS.
- Extensive experiments across various datasets demonstrate the effectiveness of SPA, achieving a TPR of over 95% in identifying watermarked embeddings.

2 Preliminary

2.1 EaaS Copyright Infringement

Publicly deployed APIs, particularly in recent EaaS services, have been shown vulnerable (Liu et al., 2022; Sha et al., 2023). We focus on EaaS services based on LLMs, defining the victim model as Θ_v, which provides the EaaS service S_v. The client's query dataset is denoted as D, with individual text as d_i. Θ_v computes the original embedding $e_{o_i} \subseteq \mathbb{R}^{dim}$, where dim is the embedding dimension. To protect EaaS copyright, a watermark is injected into e_{o_i} before delivery. Backdoor-based watermarking schemes (Adi et al., 2018; Li et al., 2022; Peng et al., 2023) are used to inject a hidden pattern into the model's output, acting as a watermark. We denote this scheme as f, producing the final watermarked embedding $e_{p_i} = f(e_{o_i})$.

2.2 EaaS Watermarks

EmbMarker (Peng et al., 2023) is the first to propose using backdoor-based watermarking to protect the copyright of EaaS services. It injects the watermark by implanting a backdoor, which the embedding of text containing triggers is linearly added with a predefined watermark vector. It can be defined as

$$e_{p_i} = Norm\Big\{(1-\lambda)\cdot e_{o_i} + \lambda \cdot e_t\Big\}, \quad (1)$$

where λ represents the strength of the watermark injection and e_t represents the watermark vector. EmbMarker (Peng et al., 2023) utilizes the difference of cosine similarity and L_2 distance (ΔCos and ΔL_2) between embedding sets with and without watermark to conduct verification. The embedding set with watermark will be more similar with e_t. Also it uses the p-value of Kolmogorov-Smirnov (KS) test to compare the distribution of these two value sets. The limitations of a single watermark vector make it vulnerable, prompting WARDEN (Shetty et al., 2024a) to propose a multi-watermark scheme. It can be defined as

$$e_{p_i} = Norm\Big\{(1-\Sigma_{r=1}^{R}\lambda_r)\cdot e_{o_i} + \Sigma_{r=1}^{R}\lambda_r \cdot e_{t_r}\Big\}, \quad (2)$$

where λ_r represents the different strengths of watermarks and e_{t_i} represents the different watermark vectors. WET (Shetty et al., 2024b) injects the watermark into all the embeddings without considering the text with triggers, which may have a great impact on the utility of the embeddings. VLPMarker (Tang et al., 2023) extends the backdoor-based watermarking to multi-modal models.

2.3 Attacks on EaaS Watermarks

Current attacks on EaaS watermarks generally fall into two categories: watermark elimination attacks

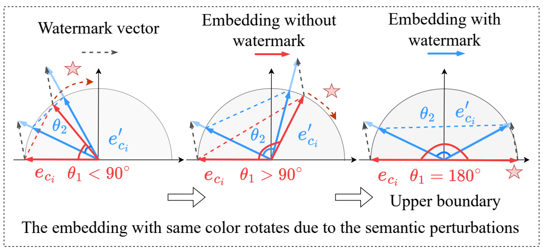

Watermark vector
Embedding without watermark
Embedding with watermark
θ_2
e'_{c_i}
e_{c_i} $\theta_1 < 90°$
e_{c_i} $\theta_1 > 90°$
e_{c_i} $\theta_1 = 180°$
Upper boundary
The embedding with same color rotates due to the semantic perturbations

Figure 2: Semantic Perturbation Demonstration in 2D Space. When the perturbed angle reaches 180°, this $\theta_1 < \theta_2$ relationship holds for any watermark vector.

and watermark identification attacks.

Watermark Elimination Attacks. They aim to bypass watermark verification by modifying the original embeddings to remove injected watermark signals. Typical methods include CSE (Shetty et al., 2024a) and PA (Shetty et al., 2024b).

Watermark Identification Attacks. They aim to bypass watermark verification by identifying and removing the watermarked embeddings. ESSA (Yang et al., 2024) is a representative method.

Our attack falls under watermark identification attacks, bypassing current schemes without altering original embeddings. In addition, our attack identifies watermarked embeddings in both single and multi-watermark scenarios while ESSA struggles with multi-watermark schemes. Detailed description of different attacks is given in Appendix A.

3 Motivation

As discussed in Section 2.2, e_t is independent of e_{o_i}, showing that the watermark siginal is semantic-independent. The watermark signal will affect watermarked samples and unwatermarked samples differently when faced with semantic perturbations. A key insight is that under semantic perturbations, the text with triggers should exhibit fewer embedding changes than the text without triggers due to the semantic-independent component.

Effective perturbations increase the likelihood of identifying watermarked embeddings as outliers, accompanied by an upper boundary that guarantees complete identification. For a text sample d_i, its perturbed form d'_i yields the embedding pair (e_i, e'_i). Both e_i and e'_i are high-dimensional vectors. To visualize perturbations, we utilize a 2D example with a fixed watermark vector vec_t. As illustrated in Figure 2, assume text d_i contains triggers, and perturbations preserve the original triggers without introducing new ones. Without injecting vec_t, the angle between (e_i, e'_i) is θ_1. And after injecting vec_t, the angle between e_i and e'_i changes to θ_2. In Figure 2, red vectors represent original ones, transforming to blue vectors after adding vec_t. Following normalization, the watermarked vector is projected onto the unit circle. The goal of constructing (d_i, d'_i) is to ensure $\theta_2 < \theta_1$, clustering watermarked embeddings tightly in vector space. This angle distribution difference can be used to identify suspicious samples. When θ_1 is small, achieving $\theta_2 < \theta_1$ requires $|vec_t|$ to be large and form an angle $< 180°$ with e_i and e'_i. For large θ_1, constraints on vec_t relax. $\theta_1 = 180°$ is the upper boundary of semantic perturbation (Figure 2). If e'_i opposes e_i, any vec_t ensures $\theta_2 < \theta_1$.

4 Semantic Perturbation Attack

In this section, we offer a detailed explanation of Semantic Perturbation Attack (SPA). The key insight of SPA lies in leveraging carefully designed semantic perturbations to amplify the divergence between watermarked and non-watermarked embeddings in their responses to semantic variations. Since existing watermarking methods are semantic-independent, the watermark signals in watermarked embeddings remain invariant under perturbations, whereas original embeddings exhibit semantically coherent variations. This fundamental discrepancy in variation patterns enables watermark identification. Thus, SPA is constructed with total three components: (1) Semantic Perturbation Strategy; (2) Embeddings Tightness Measurement; (3) Threshold Selection. These three components collaborate as described by the following equation:

$$D_{sc} = \{d_{c_i} \in D_c \mid S(d_{c_i}, G(d_{c_i})) < \varphi\}, \quad (3)$$

where G indicates how to guide the semantic perturbation, S represents the tightness measurement of embeddings before and after perturbation, and φ is the selected threshold for indentifying the watermarked samples from the query datasets. The attacker queries the victim service using a dataset D_c. And each sample in D_c is defined as d_{c_i}. D_{sc} represents the purified dataset after SPA. The overview and workflow of SPA is illustrated in Figure 3.

4.1 Threat Model

Based on real-world scenarios and previous work (Peng et al., 2023; Shetty et al., 2024a), we define the threat model, including the objective, knowledge, and capability of the attacker. Notably, the attacker can only interact with EaaS services in a

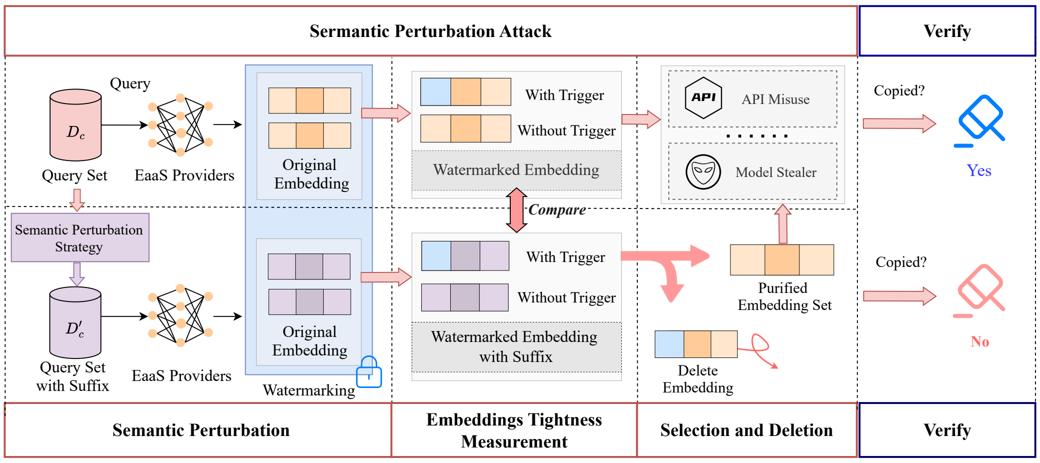


Figure 3: The Framework of Semantic Perturbation Attack. Attackers apply the semantic perturbation strategy to modify the original query dataset. The semantic-independent characteristic enables the selection and deletion of watermarked embeddings, ultimately resulting in a purified dataset that bypasses watermark verification.

black-box approach, but is capable of leveraging a general text corpus D_p for assistance (Shetty et al., 2024a). Further details of the threat model can be found in Appendix B.

4.2 Semantic Perturbation Strategy

To conduct perturbation, many text-modifying techniques (e.g., synonym replacement) are available. However, these techniques are not suitable for EaaS scenario which may invalidate the original triggers in text. Therefore, we design a method that concatenates text as a suffix to the input text to avoid invalidating the original triggers. All perturbations use suffix concatenation with $d'_{c_i} = d_{c_i} + perb$ and the corresponding embedding e'_{c_i}. We first explore a naive approach of randomly selecting text from the general text corpus as the perturbation suffix. However, this naive approach is only effective on some datasets. Details are provided in Appendix C. Therefore, we further proposed SPA.

SPA serves as an enhancement to the random perturbation approach. To search for optimal perturbation text as suffix, SPA leverages a lightweight open-source embedding model Θ_s locally without any training. By encoding $(d_{c_i}, perb)$ from Θ_s, embeddings (se_{c_i}, se_{perb}) are obtained, where $perb$ traverses the general text corpus as perturbation pool. The *top-k* perturbations with the lowest similarity between (se_{c_i}, se_{perb}) are selected, maximizing the semantic gap between d_{c_i} and $perb$. After obtaining the optimal suffixes through the guidance of Θ_s, constructing $(d_{c_i}, d_{c_i} + perb)$ can effectively perform semantic perturbation on embeddings from Θ_v to detect the presence of watermarks.

Algorithm 1 Suffix Direct Search Guidance

```
1:  Input: Perturbation Pool $P$, Dataset $D_c$,
           Standard Model $\Theta_s$, Hyperparameter $k$
3:  Output: Metric Values Set $v$
4:  Initialize $s \leftarrow \emptyset$(Suffix)
5:  Initialize $n \leftarrow |D_c|$, $m \leftarrow |P|$
6:  Set $max(s) \leftarrow 1$      {▷ Cosine similarity range: [-1, 1]}
7:  for $i = 1$ to $n$ do
8:      for $j = 1$ to $m$ do
9:          Encode: $se_{c_i} \leftarrow \Theta_s(d_{c_i})$, $se_{perb} \leftarrow \Theta_s(perb_j)$
10:         $sim \leftarrow cosine(se_{c_i}, se_{perb})$
11:         if $|s| < k$ then
12:             Append $perb_j$ to $s$
13:         else if $|s| \geq k$ and $sim < max(s)$ then
14:             Remove $max(s)$ from $s$
15:             Insert $perb_j$ into $s$
16:         end if
17:     end for
18:     Compute aggregate metric: $metric \leftarrow agg(s)$
19:     Append $metric$ to $v$
20: end for
21: return $v$
```

The core idea stems from two components: (1) **Similarity Representing Semantic Gap**: For text d_{c_i}, its embedding e_{c_i} is the feature representation of d_{c_i} in a high-dimensional space. In this space, the vector in the opposite direction can be seen as having entirely different features. Lower similarity between embeddings corresponds to greater semantic gap. (2) **Dual-Model Correlation**: Since the embeddings obtained from EaaS services may contain watermarks, SPA employs a lightweight local model Θ_s to guide semantic perturbations. Both Θ_s and the victim model Θ_v fundamentally capture textual features through their respective embeddings. Although their embeddings reside in distinct feature spaces, they exhibit consistent differential properties (the similarity between em-

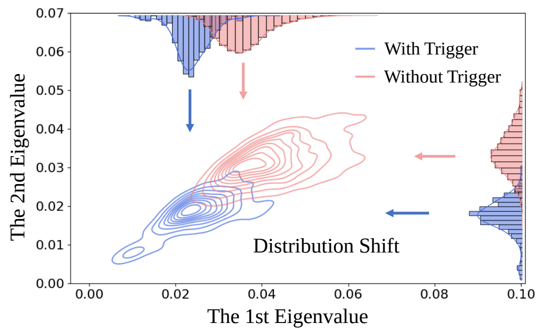


Figure 4: PCA Score Visualization.

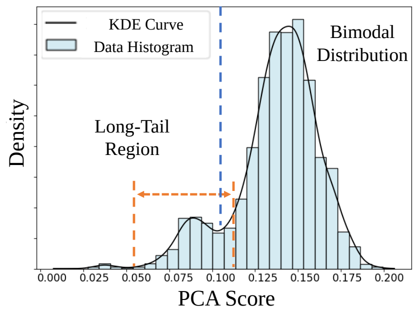


Figure 5: Threshold Selection.

beddings) in feature representation. This enables Θ_s to effectively guide suffix selection despite architectural differences, as shown in Appendix G.

SPA obtains the optimal perturbation text via Θ_s. And concatenating the text with obvious semantic gap allows for significant perturbation. Θ_s encodes dataset D_c and the perturbation pool only once, with linear time complexity of $|D_c| + |perb\ pool|$. The complete process is in Algorithm 1. We use Sentence-BERT (Reimers and Gurevych, 2019) without any training as Θ_s, which has fewer dimensions than the victim model (384 $\leftrightarrow$ 1536) and only 22.7M parameters.

4.3 Embeddings Tightness Measurement

After applying the strategies mentioned above to semantically perturb the input text, we require effective metrics to capture the "tightness" or "variation patterns" between the original embeddings and the perturbed embeddings, in order to indentify whether semantic-independent watermark signals are present. Our primary evaluation consists of three metrics represented as

$$Cosine_i = \frac{1}{k}\Sigma_{j=1}^{k}\frac{e_{c_i}\cdot e_{c_i}^{j'}}{|e_{c_i}|\cdot|e_{c_i}^{j'}|},$$
$$L_{2_i} = \frac{1}{k}\Sigma_{j=1}^{k}\left|\frac{e_{c_i}}{|e_{c_i}|} - \frac{e_{c_i}^{j'}}{|e_{c_i}^{j'}|}\right|,$$
$$PCA\ Score_i = \Sigma_{d=1}^{D_{pca}} f_{pca}(e_{c_i}^{j'} \mid j = 1, 2, 3, \dots, k)$$
$$D_{pca} : lower\ dimension, \tag{4}$$

where the three metrics are based on cosine similarity, L_2 distance, and PCA score, representing the similarity of the original embeddings and perturbed ones. However, text perturbations may rarely introduce new triggers. Thus, k perturbations are conducted for each sample, combining results from k trials to mitigate potential impacts.

Cosine Similarity Metric: Cosine similarity measures the cosine of the angle between the embeddings in the vector space. We use the average of the k trials as one of the evaluation metrics.

L_2 Distance Metric: L_2 distance represents the straight-line distance between two data points in high-dimensional space. We use the average of the k trials as one of the evaluation metrics.

PCA Score Metric: We perform k perturbations, obtaining e_{c_i} and k perturbed embeddings: $\{e_{c_i}^{j'} \mid j = 1, 2, \dots, k\}$. For each sample d_{c_i}, an embedding set of size $k+1$ is obtained. We apply PCA to compute eigenvalues for each principal component. If d_{c_i} contains triggers, the embeddings will cluster tightly in high-dimensional space because of the identical watermark signal, resulting in smaller eigenvalues after PCA. Thus, we use the sum of eigenvalues as one of the evaluation metrics, as shown in Equation 4, where D_{pca} is the reduced dimension and f_{pca} computes eigenvalues. Reducing embeddings to two dimensions and using eigenvalues as coordinates yields Figure 4.

4.4 Threshold Selection

After obtaining the metrics of tightness measurement, we need to determine a threshold to distinguish between watermarked and non-watermarked embeddings, as the attacker has no knowledge of the ground truth. The metric distributions will exhibit a long-tail phenomenon due to texts with triggers. Figure 5 elaborates on the attacker's perspective. A partially overlapping and imbalanced PCA score distribution may occur. Consequently, we select the saddle point between bimodal distributions as the decision threshold φ for watermark identification, where the first-order derivative equals zero at this critical point. Samples with metrics below φ are identified as containing watermarks and removed from D_c, yielding a purified dataset. The majority of samples with triggers are eliminated. Although some benign data might also be removed, it represents only a small proportion of D_c.

Table 1: Watermark Identification Attack Performance.

Datasets	Methods	EmbMarker				WARDEN			
		$ACC.(\%)\uparrow$	**Detection Performance**			$ACC.(\%)\uparrow$	**Detection Performance**		
			$\Delta Cos(\%)$	$\Delta L_2(\%)$	$p-value$		$\Delta Cos(\%)$	$\Delta L_2(\%)$	$p-value$
SST2	Original	91.60	$+2.37$	-4.74	10^{-5}	91.00	$+6.47$	-12.94	10^{-6}
	+ ESSA	91.00	-0.06	$+0.12$	10^{-1}	92.60	$+5.47$	-10.93	10^{-7}
	+ Random	91.20	$+0.25$	-0.51	10^{-1}	91.20	-1.75	$+3.01$	10^{-4}
	+ SPA	91.00	$+0.17$	-0.33	$\mathbf{10^{-1}}$	90.00	-1.08	$+2.16$	$\mathbf{10^{-2}}$
AG News	Original	88.80	$+1.99$	-3.99	10^{-6}	89.00	$+5.92$	-11.84	10^{-8}
	+ ESSA	89.57	$+1.14$	-2.28	10^{-2}	89.76	$+12.79$	-25.58	10^{-11}
	+ Random	89.00	$+0.65$	-1.30	10^{-2}	90.20	$+2.58$	-5.16	10^{-7}
	+ SPA	89.80	$+0.26$	-0.52	$\mathbf{10^{-1}}$	89.00	$+0.98$	-1.95	$\mathbf{10^{-2}}$
Enron Spam	Original	92.00	$+5.99$	-11.99	10^{-7}	92.20	$+5.19$	-10.39	10^{-8}
	+ ESSA	92.00	-0.51	$+1.03$	10^{-1}	92.60	$+5.47$	-10.93	10^{-7}
	+ Random	91.60	$+0.11$	-0.22	10^{-1}	92.00	$+1.56$	-3.12	10^{-4}
	+ SPA	91.40	$+0.49$	-0.98	$\mathbf{10^{-1}}$	92.40	$+1.25$	-2.50	$\mathbf{10^{-2}}$
MIND	Original	70.20	$+5.64$	-11.28	10^{-6}	71.80	$+9.26$	-18.52	10^{-6}
	+ ESSA	70.10	-0.62	$+1.24$	10^{-1}	70.18	$+4.63$	-9.26	10^{-6}
	+ Random	70.00	-0.32	$+0.66$	10^{-1}	69.20	$+2.48$	-4.96	10^{-2}
	+ SPA	70.00	-0.33	$+0.66$	$\mathbf{10^{-1}}$	70.00	$+2.80$	-5.61	$\mathbf{10^{-2}}$

5 Experiment

5.1 Experiment Setup

We evaluate SPA on EmbMarker (Peng et al., 2023) and WARDEN (Shetty et al., 2024a), with text classification as downstream tasks and OpenAI's text-embedding-ada-002 as the victim model. Experiments are conducted on four datasets: SST2 (Socher et al., 2013), AG News (Zhang et al., 2015), Enron Spam (Metsis et al., 2006), and MIND (Wu et al., 2020). Due to API costs, we sample subsets of each dataset for experiments. Our results are the average of multiple experiments. Details of the datasets are in Appendix D.

Baselines. We adopt ESSA (Yang et al., 2024), CSE (Shetty et al., 2024a) and PA (Shetty et al., 2024b)as baselines, with ESSA as watermark identification attack and (CSE, PA) classified as watermark elimination attacks. We also compare the naive approach of random perturbation with SPA.

Metrics. We employ the AUPRC to quantify the cosine similarity, L_2 distance, and PCA score. A higher AUPRC indicates better performance in watermark identification. We also utilize the TPR, FPR and Precision to assess the performance of watermark identification. The $p-value$, ΔCos, and ΔL_2 are employed to assess the verification ability of the watermark. The attack is considered successful when the $p-value$ ceases to be statistically significant (i.e., reaches 10^{-1} or 10^{-2}), with ΔCos and ΔL_2 approaching zero. The utility of embeddings is evaluated through the downstream tasks accuracy.

Settings. k perturbations are involved for each text, with $k = 10$ chosen to balance considerations of time and cost. The suffix search guidance uses the WikiText (Merity et al., 2016) dataset totaling one million entries as the candidate pool.

5.2 Main Results

5.2.1 Attack Performance

We conduct a comprehensive evaluation of SPA and other attack methods, which further highlight the performance and advancement of SPA.

Bypass Watermark Verification. In SPA, almost $95\% - 100\%$ of watermarked samples are identified and removed as shown in Table 3. Thus, SPA results in a significant increase in $p - value$ from $(10^{-6}, 10^{-7}, 10^{-8})$ to $(10^{-1}, 10^{-2})$, leading to the failure of watermark verification as shown in Table 1. For other attack methods, ESSA and Random Perturbation fails against the multi-watermark scheme WARDEN (Shetty et al., 2024a) with $p - value$ less than 10^{-4}, meaning that they cannot clearly identify the watermarked samples. However, **SPA have the ability to indentify the watermarked samples in both single-watermark and multi-watermark scheme**, as effectively bypassing different schemes across all four datasets.

The Utility of Embeddings. In SPA, attacker will finally remove the suspicious samples from original dataset. Therefore, we conduct experiments to test whether the performance of embeddings for downstream tasks is affected. Table 1 demonstrates that after applying SPA, although the dataset size is reduced, the decrease in accuracy on downstream tasks is generally below 1%. Moreover, the accuracy on some datasets even slightly improves after SPA. Thus, **the utility of embeddings is basically unaffected, remaining comparable to the performance of original dataset.**

Table 2: Performance of Combining SPA with Elimination Attack.

Datasets	Methods	EmbMarker				WARDEN			
		$ACC.(\%)\uparrow$	Detection Performance			$ACC.(\%)\uparrow$	Detection Performance		
			$\Delta Cos(\%)$	$\Delta L_2(\%)$	$p-value$		$\Delta Cos(\%)$	$\Delta L_2(\%)$	$p-value$
SST2	Original	91.60	+2.37	−4.74	10^{-5}	91.00	+6.47	−12.94	10^{-6}
	+ PA	90.57	+0.27	−0.54	10^{-1}	90.34	+0.12	−0.24	10^{-1}
	+ CSE	90.54	+0.65	−1.31	10^{-2}	91.40	+0.05	−0.10	10^{-1}
	+ (SPA+CSE)	**91.20**	+0.20	−0.40	10^{-1}	**92.00**	−1.02	+2.18	10^{-2}
AG News	Original	88.80	+1.99	−3.99	10^{-6}	89.00	+5.92	−11.84	10^{-8}
	+ PA	88.68	+4.27	−8.54	10^{-7}	88.60	+5.80	−11.60	10^{-11}
	+ CSE	89.96	+0.35	−0.70	10^{-2}	89.75	+0.93	−1.88	10^{-1}
	+ (SPA+CSE)	**90.40**	+0.36	−0.72	10^{-1}	**90.36**	+1.06	−2.41	10^{-2}
Enron Spam	Original	92.00	+5.99	−11.99	10^{-7}	92.20	+5.19	−10.39	10^{-8}
	+ PA	90.40	−0.25	+0.50	10^{-1}	90.85	+0.02	−0.03	10^{-1}
	+ CSE	91.25	+0.40	−0.81	10^{-1}	91.90	+0.94	−1.88	10^{-1}
	+ (SPA+CSE)	**92.40**	+0.57	−1.02	10^{-1}	**92.42**	+1.32	−2.63	10^{-2}
MIND	Original	70.20	+5.64	−11.28	10^{-6}	71.80	+9.26	−18.52	10^{-6}
	+ PA	69.25	+0.22	−0.45	10^{-1}	69.26	+1.33	−2.65	10^{-1}
	+ CSE	69.62	+0.93	−1.86	10^{-2}	70.38	−0.02	+0.04	10^{-1}
	+ (SPA+CSE)	**71.00**	−0.44	+0.88	10^{-1}	**73.34**	+2.20	−4.41	10^{-2}

Table 3: Semantic Perturbation Attack Performance. '$\star$' demonstrates the most important metrics.

Datasets	Schemes	Cos AUPRC	L_2 AUPRC	PCA AUPRC*	Deletion Performance			
					Total Deletion	*TPR*$^\star\uparrow$	*FPR*$\downarrow$	*Precision*$\uparrow$
SST2	EmbMarker	0.8947	0.8888	0.9214	439/5000	95.68%	2.30%	75.63%
	WARDEN	0.6190	0.6190	0.9000	437/5000	95.68%	2.26%	75.97%
AG News	EmbMarker	0.5665	0.5398	0.7052	1478/5000	97.65%	19.62%	42.08%
	WARDEN	0.3323	0.3323	0.6791	1498/5000	96.86%	20.19%	41.19%
Enron Spam	EmbMarker	0.9284	0.9227	0.9685	572/5000	91.49%	1.26%	90.21%
	WARDEN	0.7348	0.7348	0.9530	619/5000	92.91%	2.14%	84.65%
MIND	EmbMarker	1.0	1.0	1.0	152/5000	100%	0%	100%
	WARDEN	0.4971	0.4971	0.7957	188/5000	84.21%	1.24%	68.09%

5.2.2 Orthogonal Combination

SPA serves as an effective identification method that can be orthogonally combined with watermark elimination attacks to achieve enhanced performance. Thus, SPA can be effectively combined with CSE to precisely localize the suspicious embeddings, while CSE's removal mechanism eliminates the watermark signals from these targeted embeddings. SPA+CSE modify only suspicious embeddings with TPR almost above 95% instead of large-scale modifications to the embeddings, thus enhancing the utility of embeddings as shown in Table 2. It also demonstrates comparable attack performance across different schemes, with p-values reaching magnitudes of 10^{-1} or 10^{-2}. Compared to standalone SPA, SPA+CSE avoids dataset reduction while outperforming standalone CSE in providing more precise identification of watermarked embeddings, **thereby achieving the highest accuracy in downstream tasks**.

5.2.3 Time Overhead

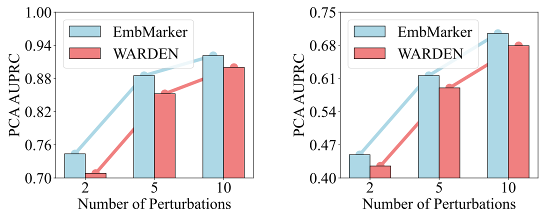
In SPA, local model encoding process constitutes the dominant time cost with the linear time complexity. Thus, we measured the time required for suffix search in SPA. In our settings, searching for the $top-10$ optimal perturbation suffixes for a single text from the pertrubation pool of size 10^6 only requires 0.15 seconds on average across all four datasets. Meanwhile, we evaluate the effectiveness of SPA under varying perturbation pool sizes. Even with a pool of size 10^3, comparable perturbation performance can be achieved. Detailed information of experiment results can be found in Appendix H.

5.3 Ablation Study

We conducted extensive experiments on SPA from multiple perspectives to validate its effectiveness and capability across various scenarios. For clarity of presentation, we conduct experiments on SST2 (Socher et al., 2013) and AG News (Zhang et al., 2015) datasets.

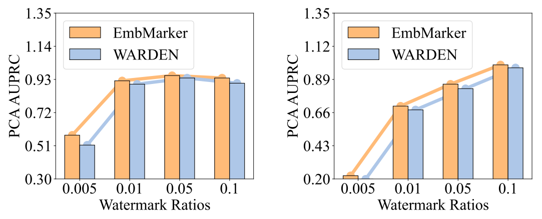
PCA Score demonstrates superior robustness compared to other metrics. Table 3 shows that the PCA score metric remains stable across different schemes. It also shows the performance of watermark identification using the PCA score metric, along with a TPR universally exceeding 90%. This is likely because the PCA algorithm extracts and preserves the watermark information in the embeddings while eliminating redundant information.

SPA performance improves as the number of semantic perturbations increases. We evaluated SPA performance under different numbers



(a) SST2 (b) AG NEWS

Figure 6: PCA AUPRC and Number of Perturbations.



(a) SST2 (b) AG NEWS

Figure 7: PCA AUPRC and Watermark Ratio.

of perturbations using PCA AUPRC as the evaluation metric. The perturbation suffixes are selected following the order determined by suffix search guidance. The results shown in Figure 6 indicate that, SPA performance increases and stabilizes as the number of perturbations grows. This further demonstrates the effectiveness of SPA, as it ensures that effective suffixes are incorporated among multiple candidates.

SPA remains effective under different watermark ratios. We evaluated SPA's performance under varying watermark ratios, with a fixed number of perturbations. Figure 7 shows that even with low watermark ratios (low-frequency triggers), SPA achieves a PCA AUPRC of 0.3-0.5, *despite the stealer model failing to learn the watermark behavior*. Performance improves as the watermark ratio increases. However, a high watermark ratio will result in excessive watermark injection and embedding modification. Nevertheless, the PCA AUPRC remains above 0.9, demonstrating SPA's robustness across varying watermark ratios.

6 Discussion of Mitigation Strategies

To counter SPA, we explore potential mitigation strategies. We suggest a deep learning-based solution with: (1) a semantic-aware injection model that dynamically injects watermarks based on semantic features, and (2) a verification model. However, this approach introduces a critical trade-off: when the adaptive watermark is learned by a stealer model, verification becomes substantially more challenging. In addition, we attempt to reproduce the adaptive EaaS watermarking schemes (Wang and Cheng, 2024; Wang et al., 2024) in contemporary studies. We discovere conclusions consistent with our exploration. We believe that the semantic-aware watermarking paradigm represents a promising direction, yet further investigation into its verification capability remains essential. Detailed results and analysis are provided in Appendix F.

7 Related Work

7.1 Model Extraction Attack

Model extraction attacks (Orekondy et al., 2019; Sanyal et al., 2022; Chandrasekaran et al., 2020) threaten Deep Neural Networks (DNNs) and cloud services by enabling adversaries to replicate models without internal access. Attackers can query APIs (Kalpesh et al., 2020) or gather physical data (Hu et al., 2020) to train the stolen models. Public APIs, especially in current EaaS services, are proved to be vulnerable (Liu et al., 2022).

7.2 Deep Watermarking

Deep watermarking can be classified into white-box, black-box, and box-free approaches based on accessible data during verification (Li et al., 2021). White-box watermarking schemes access model parameters (Yan et al., 2023; Lv et al., 2023; Pegoraro et al., 2024), while black-box schemes rely only on the model output (Leroux et al., 2024; Lv et al., 2024). Box-free watermarking schemes exploits inherent output variations without crafted queries (An et al., 2024). In EaaS, watermarking can be regarded as a form of black-box watermarking.

8 Conclusion

In this paper, we propose SPA, a novel attack on EaaS watermark exploiting the limitation that current schemes is semantic-independent. SPA conducts several semantic perturbations to input text, constructs embedding pairs using the original and perturbed embeddings, measuring the tightness of embeddings and deletes suspicious samples exceeding the threshold while preserving service utility. Our extensive experiments demonstrate the effectiveness of SPA. We also validate the importance of SPA's components and explore mitigation strategies. Our work emphasizes the critical role of text semantics in EaaS watermarking.

Limitations

In this paper, we propose SPA, a novel attack which exploits the semantic-independent vulnerabilities inherent in current EaaS watermarking schemes, successfully removing the majority of watermarked embeddings. However, an attacker requires a small local model for assistance to successfully execute SPA. Although such a scenario is realistic, we plan to explore attack schemes that do not require assistant models in our future work. Additionally, after each text perturbation, the attacker needs to re-access the original EaaS service, which increases the API cost of SPA. Furthermore, we note that as the number of suffixes increases, the effectiveness of SPA becomes more stable, while an insufficient number of suffixes may lead to failure of SPA, thereby further amplifying concerns regarding the associated API costs. In future, we believe that advanced watermarking schemes will emerge, but SPA provides a perspective that emphasizes the importance of text semantics in the design of EaaS watermarking schemes. We will continue to explore how to develop feasible attack and watermarking schemes with enhanced robustness.

Ethics Statement

We introduce a novel and effective attack targeting EaaS watermarks through the semantic perturbation. Our objective is to underscore the critical consideration of text semantics in EaaS watermark design, thereby enhancing security. We believe that the first step toward enhancing security is to expose potential vulnerabilities. All our experiments are conducted under control, with no attempts made to launch actual attacks on EaaS service providers. We have further explored potential mitigation strategies to address SPA.

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 symposium (USENIX Security)*, pages 1615–1631.

Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. 2015. Evaluation of output embeddings for fine-grained image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2927–2936.

Haonan An, Guang Hua, Zhiping Lin, and Yuguang Fang. 2024. Box-free model watermarks are prone to black-box removal attacks. *arXiv preprint arXiv:2405.09863*.

Varun Chandrasekaran, Kamalika Chaudhuri, Irene Giacomelli, Somesh Jha, and Songbai Yan. 2020. Exploring connections between active learning and model extraction. In *USENIX Security Symposium (USENIX Security)*, pages 1309–1326.

Chengyuan Deng, Yiqun Duan, Xin Jin, Heng Chang, Yijun Tian, Han Liu, Henry Peng Zou, Yiqiao Jin, Yijia Xiao, Yichen Wang, et al. 2024. Deconstructing the ethics of large language models from long-standing issues to new-emerging dilemmas. *arXiv preprint arXiv:2406.05392*.

Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth JF Jones. 2015. Word embedding based generalized language model for information retrieval. In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 795–798.

Xing Hu, Ling Liang, Shuangchen Li, Lei Deng, Pengfei Zuo, Yu Ji, Xinfeng Xie, Yufei Ding, Chang Liu, Timothy Sherwood, et al. 2020. Deepsniffer: A dnn model extraction framework based on learning architectural hints. In *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 385–399.

Jui Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 2553–2561.

Krishna Kalpesh, Tomar Gaurav Singh, P Parikh Ankur, Papernot Nicolas, and Iyyer Mohit. 2020. Thieves on sesame street! model extraction of bert-based apis. In *Proceedings of International Conference on Learning Representations (ICLR)*, pages 1–19.

Sam Leroux, Stijn Vanassche, and Pieter Simoens. 2024. Multi-bit black-box watermarking of deep neural networks in embedded applications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2121–2130.

Yiming Li, Yang Bai, Yong Jiang, Yong Yang, Shu-Tao Xia, and Bo Li. 2022. Untargeted backdoor watermark: Towards harmless and stealthy dataset copyright protection. *Advances in Neural Information Processing Systems (NIPS)*, 35:13238–13250.

Yue Li, Hongxia Wang, and Mauro Barni. 2021. A survey of deep neural network watermarking techniques. *Neurocomputing*, 461:171–193.

Yupei Liu, Jinyuan Jia, Hongbin Liu, and Neil Zhenqiang Gong. 2022. Stolenencoder: stealing pre-trained encoders in self-supervised learning. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2115–2128.

Peizhuo Lv, Pan Li, Shengzhi Zhang, Kai Chen, Ruigang Liang, Hualong Ma, Yue Zhao, and Yingjiu Li. 2023. A robustness-assured white-box watermark in neural networks. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 20(6):5214–5229.

Peizhuo Lv, Pan Li, Shenchen Zhu, Shengzhi Zhang, Kai Chen, Ruigang Liang, Chang Yue, Fan Xiang, Yuling Cai, Hualong Ma, Yingjun Zhang, and Guozhu Meng. 2024. Ssl-wm: A black-box watermarking approach for encoders pre-trained by self-supervised learning. *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

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

Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. 2017. Embedding-based news recommendation for millions of users. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1933–1942.

Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2019. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4954–4963.

Alessandro Pegoraro, Carlotta Segna, Kavita Kumari, and Ahmad-Reza Sadeghi. 2024. Deepeclipse: How to break white-box dnn-watermarking schemes. *arXiv preprint arXiv:2403.03590*.

Wenjun Peng, Jingwei Yi, Fangzhao Wu, Shangxi Wu, Bin Bin Zhu, Lingjuan Lyu, Binxing Jiao, Tong Xu, Guangzhong Sun, and Xing Xie. 2023. Are you copying my model? protecting the copyright of large language models for eaas via backdoor watermark. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 7653–7668.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3980–3990.

Sunandini Sanyal, Sravanti Addepalli, and R Venkatesh Babu. 2022. Towards data-free model stealing in a hard label setting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15284–15293.

Zeyang Sha, Xinlei He, Ning Yu, Michael Backes, and Yang Zhang. 2023. Can't steal? cont-steal! contrastive stealing attacks against image encoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16373–16383.

Anudeex Shetty, Yue Teng, Ke He, and Qiongkai Xu. 2024a. Warden: Multi-directional backdoor watermarks for embedding-as-a-service copyright protection. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 13430–13444.

Anudeex Shetty, Qiongkai Xu, and Jey Han Lau. 2024b. Wet: Overcoming paraphrasing vulnerabilities in embeddings-as-a-service with linear transformation watermarks. *arXiv preprint arXiv:2409.04459*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642.

Yuanmin Tang, Jing Yu, Keke Gai, Xiangyan Qu, Yue Hu, Gang Xiong, and Qi Wu. 2023. Watermarking vision-language pre-trained models for multi-modal embedding as a service. *arXiv preprint arXiv:2311.05863*.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2321–2331.

Liaoyaqi Wang and Minhao Cheng. 2024. Guardemb: Dynamic watermark for safeguarding large language model embedding service against model stealing attack. In *Findings of the Association for Computational Linguistics (EMNLP)*, pages 7518–7534.

Zongqi Wang, Baoyuan Wu, Jingyuan Deng, and Yujiu Yang. 2024. Espew: Robust copyright protection for llm-based eaas via embedding-specific watermark. *arXiv preprint arXiv:2410.17552*.

Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. 2020. Mind: A large-scale dataset for news recommendation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3597–3606.

Yifan Yan, Xudong Pan, Mi Zhang, and Min Yang. 2023. Rethinking White-Box watermarks on deep learning models under neural structural obfuscation.

In *USENIX Security Symposium (USENIX Security)*, pages 2347–2364.

Zuopeng Yang, Pengyu Chen, Tao Li, Kangjun Liu, Yuan Huang, and Xin Lin. 2024. Defending against similarity shift attack for eaas via adaptive multi-target watermarking. *Information Sciences*, page 120893.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in Neural Information Processing Systems (NIPS)*, 28.

Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. 2024. Adapting large language models by integrating collaborative semantics for recommendation. In *IEEE International Conference on Data Engineering (ICDE)*, pages 1435–1448.

Appendix

A Overview of Different Attack Methods

In Appendix A, we provide a comprehensive and detailed introduction to various attack methods, including CSE, PA, and ESSA.

- **ESSA** (Yang et al., 2024) is a kind of watermark identification attack. ESSA appends a token to the input text and evaluating whether the token functions as a trigger by analyzing the divergence between embeddings before and after token addition.
- **CSE** (Shetty et al., 2024a) is a kind of watermark elimination attack. CSE uses clustering to identify embedding pairs, selects potential watermarked embeddings by analyzing discrepancies between a standard model and the victim model, and eliminates principal components to erase watermark signals.
- **PA** (Shetty et al., 2024b) is a kind of watermark elimination attack. PA employs a language model to rewrite input texts multiple times, retaining semantics but potentially losing trigger tokens. Averaging embeddings from these iterations dilutes the watermark signals. This attack paradigm modifies original embeddings, inevitably compromising the utility of embeddings.

B Definition of the Threat Model

In Appendix B, we clearly define the threat model, detailing the objective, knowledge, and capability of the attacker.

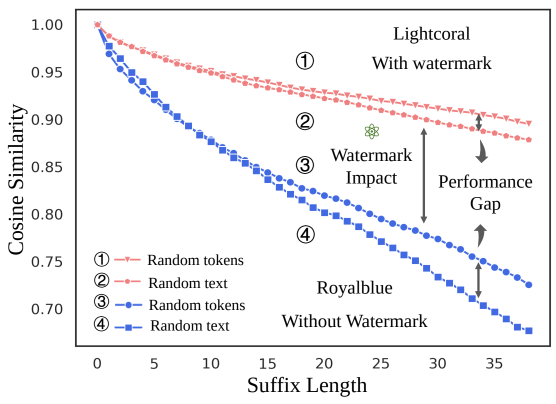


Figure 8: Different Approaches of Semantic Perturbations: Length and Semantics. Regardless of whether watermarked or not, random text preforms better than random tokens. The injection of the watermark has led to a significant gap between the curves.

Attacker's Objective. TThe attacker aims to use embeddings from the victim model Θ_v without watermark verification. The attacker can then efficiently provide a competitive alternative instead of pre-training a new model.

Attacker's Knowledge. The EaaS service operates as a black box. The attacker queries the victim service S_v using a dataset D_c, where each sample is d_{c_i}. While unaware any information of Θ_v, the attacker can reasonably access a general text corpus D_p and a small local embedding model Θ_s to design the attack algorithm.

Attacker's Capability. With sufficient budget, the attacker can query S_v to obtain the embedding set E_c for D_c. They can then employ various attack strategies to bypass watermark verification.

C Exploration of Perturbations

C.1 Exploration of Suffix

In Appendix C.1, we provide the detailed exploration of semantic perturbation. The text perturbation denoted as $perb$ can only be constructed as prefix or suffix. The potential construction space for the suffix can be classified from two perspectives: the length of the suffix and its semantics. We use EmbMarker (Peng et al., 2023) as an example.

***Random tokens without semantics*:** We first explore a simple construction method by the adding random tokens as the suffix without semantics. Specifically, we tokenize each sentence in a general text corpus and compile all tokens into a total token vocabulary. We randomly add tokens to the suffix. At this stage, we explore the relationship between suffix length and perturbation performance before and after the watermark injection, measured by (e_{c_i}, e'_{c_i}). The results in Figure 8 indicate that as the suffix length increases, the embeddings similarity gradually decreases. After the watermark injection to (e_{c_i}, e'_{c_i}), the rate of decrease significantly slows and remains notably higher than the curve without the watermark injection.

***Random text with semantics*:** We randomly selected long texts from a general text corpus, tokenize it to obtain a sequence of tokens and sequentially add each token to the suffix. We explored the effects both with and without watermark injection. The results are illustrated in Figure 8. It is evident that semantic suffix lead to a faster enhancement of perturbation performance, with the curve with watermark injection also significantly exceeding that without injection. Interestingly, for the same suffix length, the performance of perturbations using text with semantics is generally higher than that achieved with random tokens. The finding suggests that using the suffix with semantics is more cost-effective and produces better results. Therefore, we will consistently utilize the semantic suffix during the perturbation process.

***Text with & without semantics*:** For suffix, the construction space can be categorized from two perspectives: length and semantics. A series of experiments demonstrate that using random text with semantics is more cost-effective and produces better results compared to random tokens without semantics. Based on this, we propose a heuristic perturbation scheme.

C.2 Heuristic Perturbation Scheme

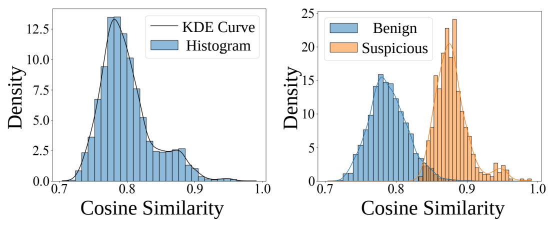


Figure 9: Cosine similarity metric distribution and KDE curve of the Enron Spam dataset in Heuristic Perturbation Scheme.

In Appendix C.2, we introduce heuristic semantic perturbation scheme. Semantic suffixes improve perturbation performance at lower costs, making suspicious samples easier to detect. Based on this, we propose a heuristic perturbation scheme. Following previous works, we focus on text classification tasks. In the context of text classification, heuristic perturbation scheme randomly selects samples with different labels from original as suffixes, leveraging semantic differences to enhance the perturbation. We randomly select k samples for perturbation and calculate the average cosine similarity of k embedding pairs, to reduce the influence of potential triggers in the suffixes. We conducted experiments on four classic datasets: Enron Spam (Metsis et al., 2006), SST2 (Socher et al., 2013), MIND (Wu et al., 2020) and AG News (Zhang et al., 2015). From the perspectives of the attacker and ground truth, the cosine similarity distribution of Enron Spam dataset is shown in Figure 9. The distribution results indicate observable differences for the Enron Spam and MIND datasets, while such differences are less pronounced for the SST2 and AG News datasets. Thus, we need to further explore a more effective approach.

Algorithm 2 Suffix Perturbation Guidance

```
 1: Input: Perturbation Pool $P$, Dataset $D_c$
 2:        Standard Model $\Theta_s$, Hyperparameter $k$
 3: Output: Metric Values $v$
 4: Initialize $s \leftarrow \emptyset$(Suffix)
 5: Initialize $n \leftarrow |D_c|$, $m \leftarrow |P|$
 6: Set $max(s) \leftarrow 1$    {▷ Cosine similarity range: [-1, 1]}
 7: for $i = 1$ to $n$ do
 8:   for $j = 1$ to $m$ do
 9:     $d'_{c_i} \leftarrow d_{c_i} + perb_j$
10:     Encode: $se_{c_i} \leftarrow \Theta_s(d_{c_i})$, $se'_{c_i} \leftarrow \Theta_s(d'_{c_i})$
11:     $sim \leftarrow cosine(se_{c_i}, se_{perb})$
12:     if $|s| < k$ then
13:       Append $perb_j$ to $s$
14:     else if $|s| \geq k$ and $sim < max(s)$ then
15:       Remove $max(s)$ from $s$
16:       Insert $perb_j$ into $s$
17:     else
18:       Skip $perb_j$
19:     end if
20:   end for
21:   Compute aggregate metric: $metric \leftarrow agg(s)$
22:   Append $metric$ to $v$
23: end for
24: return $v$
```

C.3 Semantic Perturbation Guidance

In Appendix C.3, we introduce another small local model suffix perturbation guidance approach. The results in Figure 9 indicate that the effectiveness of the simple heuristic perturbation scheme needs further improvement. Although the embedding spaces of Θ_v and Θ_s differ, the variations between (e_{c_i}, e'_{c_i}) under the same perturbation show similar patterns across all these spaces. Specifically, we input the text pair $(d_{c_i}, d_{c_i}+perb)$ into Θ_s to obtain the corresponding embedding pair (se_{c_i}, se'_{c_i}).

Table 4: Training Settings.

Datasets	Train	Test	Class	Metrics	Schemes	Original	Subset	Epoch Adjustment
SST2	$67,349 \rightarrow 5,000$	$872 \rightarrow 500$	2	ACC.(%)	EmbMarker	93.46%	91.60%	$3 \rightarrow 30$
					WARDEN	93.46%	92.20%	$3 \rightarrow 50$
AG News	$120,000 \rightarrow 5,000$	$7,600 \rightarrow 500$	4	ACC.(%)	EmbMarker	93.57%	88.80%	$3 \rightarrow 20$
					WARDEN	93.76%	89.00%	$3 \rightarrow 20$
Enron Spam	$31,716 \rightarrow 5,000$	$2,000 \rightarrow 500$	2	ACC.(%)	EmbMarker	94.85%	92.00%	$3 \rightarrow 20$
					WARDEN	94.60%	92.20%	$3 \rightarrow 10$
MIND	$97,791 \rightarrow 5,000$	$32,592 \rightarrow 500$	18	ACC.(%)	EmbMarker	77.23%	69.20%	$3 \rightarrow 75$
					WARDEN	77.18%	71.80%	$3 \rightarrow 75$

The perturbation $perb$ traverses through all candidates in the perturbation pool. The *top-k* $perb$ texts that minimize the similarity of (se_{c_i}, se'_{c_i}) are selected as candidate suffixes. Since the embeddings output by Θ_s are not watermarked, it is feasible to use this small local model to guide the perturbations for Θ_v. We similarly take the aggregate metric over k perturbed samples for evaluation. Θ_s captures the differential features between $(d_{c_i}, d_{c_i} + perb)$. Such differential features are consistent across models. However, suffix perturbation guidance is less efficient since each text have to traverse all the candidates in the perturbation pool. It results in the time complexity of $|D_c| \cdot |perb\ pool|$, requiring Θ_s to encode $|D_c| \cdot |perb\ pool|$ perturbation processes. The entire process of the algorithm is shown in Algorithm 2.

D Dataset Introduction

In Appendix D, we will provide a comprehensive description of the specific details of the datasets utilized, including their structure, preprocessing steps, and relevant statistics. The datasets selected for our experiments—SST2 (Socher et al., 2013), AG News (Zhang et al., 2015), Enron Spam (Metsis et al., 2006), and MIND (Wu et al., 2020)—are widely recognized as benchmark datasets in the field of Natural Language Processing (NLP). We apply the four datasets to the text classification task, with a primary focus on investigating the potential impact of watermarks on this downstream task.

- **SST2:** The SST2 dataset is a collection of movie reviews labeled with binary sentiment (positive or negative), commonly used for training and evaluating models in sentiment classification tasks.

- **AG News:** The AG News dataset is a collection of news articles categorized into four topics, commonly used for text classification and NLP tasks.

- **Enron Spam:** The Enron Spam dataset consists of the emails collection labeled as either “spam" or “non-spam" (ham), making it a valuable resource for studying spam filtering, email classification.

- **MIND:** The MIND dataset is a large-scale dataset designed for news recommendation. It can also used for news classification tasks.

E Experiment Settings

In Appendix E, we will provide a detailed description of the training configurations employed in our experiments. Furthermore, we demonstrate that our experimental setup is both rational and effective in conducting various evaluation tests.

Table 4 provides detailed information about the datasets used in our study. It also highlights the adjustments made to the number of training epochs in order to ensure performance on the respective subsets of each dataset. Specifically, the smallest dataset contains more than 30,000 data items, while the largest dataset includes over 12,000 data items. For our experiments, we sampled a subset of 5,000 examples from the training set and 500 examples from the test set. This sampling strategy was carefully chosen to balance the need for the cost of the experiment with the goal of maintaining representative data coverage. Table 4 indicates that, despite using subsets, the accuracy of downstream tasks has not significantly decreased in different watermarking schemes. On certain specific datasets, the accuracy achieved using the subset for training has even shown a slight improvement. This may be attributed to the inherent randomness in training process. Since the focus is on a relatively simple text classification task, the model appears to perform well even on the subset, maintaining favorable results. The results of the experiments demonstrate that conducting tests on these subsets not only produces valid and meaningful outcomes but also confirms the practicality.

Table 5: Semantic Aware Watermarking Performance

Datasets	Cos AUPRC	L_2 AUPRC	PCA AUPRC*	Deletion Performance				Verification
				Total Deletion	*TPR** ↑	*FPR* ↓	*Precision* ↑	$p - value$
SST2	0.5348	0.5296	0.6946	87/5000	20.75%	0.32%	82.76%	$10^{-7} \rightarrow 10^{-1}$
AG News	0.2731	0.2669	0.3295	216/5000	13.97%	2.97%	39.81%	$10^{-9} \rightarrow 10^{-1}$
Enron Spam	0.4133	0.4088	0.5574	133/5000	15.78%	0.99%	66.92%	$10^{-6} \rightarrow 10^{-2}$
MIND	0.9999	0.9999	0.9999	80/5000	52.63%	0%	100%	$10^{-7} \rightarrow 10^{-1}$

Table 6: Reproduction of GuardEmb

Method	Watermark Detection Performance		
	Recall (%)	Accuracy (%)	F1 Score (%)
Original	0.00	96.94	*N/A*
EmbMarker	5.91	8.67	11.14
GuardEmb	0.00	97.63	0.00

F Adaptive Watermark Analysis

In Appendix F, we will present experimental reproduction results of current adaptive watermarking schemes (Wang and Cheng, 2024; Wang et al., 2024) alongside our own extended investigations.

GuardEmb (Wang and Cheng, 2024) uses DNNs to enable dynamic watermark injection and verification, ensuring distinct watermark signals for different text embeddings. However, as shown in our reproduction results in Table 6 on SST2 dataset, GuardEmb achieves F1 Score = 0 and Recall = 0 during watermark verification, indicating that it misclassifies all watermarked embeddings as non-watermarked. Due to the use of mid-frequency trigger tokens, watermarked embeddings constitute an extreme minority class. Consequently, while GuardEmb attains high accuracy (biased by class imbalance), its F1 and Recall metrics collapse to zero. This demonstrates GuardEmb's failure to sustain verifiability against model extraction attacks.

ESpeW (Wang et al., 2024) is the same as EmbMarker (Peng et al., 2023) in the selection of the watermark, only selecting a single watermark vector. During the watermark injection process, ESpeW selects the positions with the $top - k$ smallest absolute values in the text embeddings for watermark injection, employing a strategy reminiscent of the least significant bit (LSB) approach. At the selected positions, ESpeW applies a linear combination scheme similar to EmbMarker, while the watermark verification process remains identical to that of EmbMarker. Consequently, ESpeW qualifies as an adaptive watermarking scheme, injecting watermark information at positions dynamically determined by the embedding characteristics. However, in practice, ESpeW does not rely on straightforward linear combinations. Instead, it overwrites the values at the chosen positions in the text embeddings with the corresponding watermark vector entries with high probability. This strong perturbation ensures reliable watermark verification, but it also deviates from the original paper's method. This approach is susceptible to statistical analysis attacks, as the values of watermark vector may exhibit abnormally high frequency in the embeddings, potentially undermining stealthiness.

We also try to implement an adaptive (semantic-aware) watermarking scheme and conduct extensive experiments. We implement the adaptive watermarking scheme by training the watermark injection model and the watermark verification model. Our experimental results in Table 5 demonstrate that end-to-end training of both watermark injection and verification models can significantly enhance watermark stealthiness and resistance against SPA. However, this approach introduces a critical trade-off: when the watermark is learned by a stealer model, verification becomes substantially more challenging. This represents a fundamental trade-off between watermark stealthiness and verifiability. As demonstrated in Table 5, our exploration reveals that when dynamically injecting watermarks via the watermark injection model, the stolen model struggles to learn watermarking signals, resulting in higher p-values than the fixed watermark during watermark verification. Adaptive watermarking approach generates distinct watermark signals for each instance, significantly increasing the difficulty for stolen models to learn and inherit these patterns compared to fixed watermark vectors. Our code is also publicly available in our repository.

G Differential Properties

In Appendix G, we will demonstrate that text embeddings derived from different models exhibit similarities in their differential properties. Meanwhile, we demonstrate that this similarity is preserved in

Table 7: Spearman Correlation Coefficient of Differential Properties

Comparison Type	Architecture	Model Size	Spearman Correlation Coefficient			
			Enron	SST2	MIND	AG News
$(victim, random)$	–	–	0.000	-0.002	0.001	0.009
$(victim, local_1)$	paraphrase-MiniLM-L6-v2	80 MB	0.403	0.516	0.537	0.504
$(victim, local_2)$	paraphrase-MiniLM-L12-v2	120 MB	0.431	0.557	0.534	0.501
$(victim, local_3)$	all-mpnet-base-v2	420 MB	0.484	0.625	0.635	0.668
$(victim, local_4)$	sentence-t5-large	640 MB	0.562	0.576	0.638	0.609

Table 8: Perturbation with Different Pool Size

Pool Size	AUPRC			Deletion	
	Cosine	L2	PCA	TPR	Precision
10^6	0.9982	0.9979	0.9979	0.57	100%
10^5	0.9666	0.9656	0.9802	0.56	100%
10^4	0.9536	0.9518	0.9762	0.53	100%
10^3	0.9767	0.9690	0.9517	0.51	100%

models with different parameters.

To systematically validate the consistency of differential properties across different embedding models, we conduct a controlled experiment by randomly sampling 100 text embeddings from four distinct datasets. For each dataset, pairwise cosine similarities between embeddings are calculated to construct a 100×100 similarity matrix, which is subsequently transformed into a rank matrix by sorting similarity values per embedding. Using the rank matrix derived from OpenAI API embeddings (serving as the victim model in our paper) as the reference benchmark, we calculate the Spearman Correlation Coefficient with: (1) the rank matrix obtained using the local model (such as paraphrase-MiniLM-L6-v2), and (2) the randomly selected rank matrix. The Spearman Correlation Coefficient ranges from [-1, 1], where values closer to 1 indicate a stronger positive correlation.

As demonstrated in Table 7, our experimental results reveal:

- A strong positive correlation between similarity gap in embeddings from OpenAI API and those from the local model.
- No significant correlation with randomly selected embeddings.

H Perturbation Pool Size

In Appendix H, we will demonstrate that under varying sizes of the text perturbation pool, the perturbation suffixes selected through guidance from a local small model consistently achieve effective semantic perturbation performance. Furthermore, as the pool size decreases, the semantic perturbation effectiveness exhibits only a marginal decline.

In our experiments, we evaluate the effectiveness of semantic perturbations under varying perturbation pool sizes. Specifically, we randomly sample 100 watermarked embeddings and 100 non-watermarked embeddings, conducting experiments across perturbation pools of sizes 10^3, 10^4, 10^5, and 10^6. We randomly sample the pools from Wikitext-103-raw-v1 (Merity et al., 2016) dataset to construct the perturbation pool. As shown in Table 8, the effectiveness of semantic perturbations exhibits only minimal degradation across different perturbation pool sizes. Even with a pool size as small as 1,000 samples, comparable perturbation performance can be achieved.