WET: Overcoming Paraphrasing Vulnerabilities in Embeddings-as-a-Service with Linear Transformation Watermark

Anudeex Shetty[1], Qiongkai Xu[1,2], Jey Han Lau[1]

[1]School of Computing and Information System, the University of Melbourne, Australia
[2]School of Computing, FSE, Macquarie University, Australia
anudeexs@student.unimelb.edu.au
qiongkai.xu@mq.edu.au
laujh@unimelb.edu.au

Abstract

Embeddings-as-a-Service (EaaS) is a service offered by large language model (LLM) developers to supply embeddings generated by LLMs. Previous research suggests that EaaS is prone to imitation attacks—attacks that clone the underlying EaaS model by training another model on the queried embeddings. As a result, EaaS watermarks are introduced to protect the intellectual property of EaaS providers. In this paper, we first show that existing EaaS watermarks can be removed by paraphrasing when attackers clone the model. Subsequently, we propose a novel watermarking technique that involves linearly transforming the embeddings, and show that it is empirically and theoretically robust against paraphrasing.[1]

1 Introduction

Large language models (LLMs) represent the state-of-the-art in natural language processing (NLP) due to their remarkable ability to understand languages and generate texts (Zhao et al., 2023). To make LLMs more accessible, LLM developers such as OpenAI and Google provide Machine-Learning-as-a-Service (MLaaS) to assess their models.

Embeddings-as-a-Service (EaaS) is a variant of MLaaS that offers feature extraction capabilities by delivering embeddings generated by LLMs (OpenAI, 2022). Alarmingly, Liu et al. (2022) demonstrated successful imitation attacks on these services. Specifically, they showed that it is possible to clone the underlying EaaS model by training a different model (with a different architecture) using queried embeddings, thereby violating the intellectual property (IP) of LLM developers.

Watermarking techniques have been proposed to defend against these EaaS imitation attacks. EmbMarker (Peng et al., 2023) introduces a method that integrates a *target embedding* into the original embedding based on the presence of *trigger words*—a pre-defined set of words—in the input text. Such techniques implant verifiable statistical signals, *i.e.,* watermarks, for the service provider to verify if their model has been copied. However, Shetty et al. (2024) demonstrated that an attacker could circumvent EmbMarker by using a contrastive method to identify and remove the single target embedding from the embedding space. To counter this, they introduced WARDEN, which strengthens the defence by incorporating *multiple* target embeddings instead of just one, making it more challenging for an attacker to eliminate the watermarks.

Nonetheless, these methods rely on words to trigger watermark injection, which we suspect could be circumvented by paraphrasing the input texts and using their queried embeddings during imitation attacks. To this end, we show that paraphrasing does dilute the watermark and thereby reveals a new form of vulnerability in these watermarking techniques. To address this vulnerability, we introduce a new defence technique, *WET* (**W**atermarking **E**aaS with Linear **T**ransformation), which applies linear transformations to the original embeddings to implant watermarks that can be verified later through reverse transformation. We analyse *WET* both theoretically and empirically to show it is robust against the new paraphrasing attack. Extensive experiments demonstrate near-perfect verifiability, even with one sample. Additionally, the utility of embeddings is mostly preserved due to the use of simple linear transformations.

The contributions of our work are as follows:

- We introduce and validate paraphrasing attack to bypass current EaaS watermarking techniques.

- We design a novel EaaS watermarking method, *WET*, and show that it is robust against paraphrasing attacks.

---

[1]The code can be found at https://github.com/anudeex/WET.git.

2 Related Work

2.1 Imitation Attacks

An imitation attack, also known as "model stealing" or "model extraction" (Tramèr et al., 2016; Orekondy et al., 2019; Krishna et al., 2020; Wallace et al., 2020), involves an imitator querying an MLaaS (or EaaS) to construct a surrogate model without the authorisation of the victim service providers. The primary motivation is to bypass service charges or even offer competitive services (Xu and He, 2023). Imitation attacks extend beyond IP violations; they can also be used to craft adversarial examples (He et al., 2021) and conduct privacy breaches like attribute inference (He et al., 2022a). Notably, Xu et al. (2022) demonstrated that a copied model can outperform the victim model through ensemble and domain adaptation. Recent successful imitation attacks on EaaS (Liu et al., 2022) not only compromise the confidentiality of embeddings but also violate the copyright of EaaS providers. These attacks constitute the threat model we explore in our research.

2.2 Text Watermarks

He et al. (2022b) and He et al. (2022c) introduced early text watermarking techniques by selectively replacing words in LLM-generated text with synonyms. A more recent work by Kirchenbauer et al. (2023) advanced text watermarks by biasing LLMs towards a set of preferred words—verifiable later—using a pseudo-random list based on the most recent tokens. Building on this approach, several works (Kuditipudi et al., 2024; Christ et al., 2024; Aaronson, 2023) have applied cryptographic methods to watermarking, using a secret key to minimise the gap between original and watermarked distributions, thereby making the watermark unbiased and stealthy.

Recent studies (Sadasivan et al., 2023; Krishna et al., 2024) demonstrated that these watermarks are vulnerable to paraphrasing-based attacks, where paraphrasing the generated text disrupts the token sequences, thereby evading watermark detection. Similarly, He et al. (2024) demonstrated that round-trip translation, another form of paraphrasing, can diminish watermark detection. These observations motivate our attack, in which we explore the use of paraphrasing and round-trip translation to remove watermarks from *embeddings*.

2.3 Embedding Watermarks

Peng et al. (2023) proposed EmbMarker, the first watermark algorithm designed to protect EaaS against imitation attacks. This algorithm uses a set of trigger words and a fixed target embedding as a watermark, where the target embedding is proportionally added to the original embedding based on the number of trigger words present in the input text. In other words, the number of trigger words determines the *watermark weight*. However, EmbMarker has only been tested against a narrow range of attacks and relies on the secrecy of the target embedding.

Shetty et al. (2024) showed that it is possible to recover the target embedding used in EmbMarker and subsequently eliminate it from the embeddings. To counter this, they proposed WARDEN, an improved watermarking technique that incorporates multiple target embeddings, making the watermark more difficult to recover. WARDEN, however, still relies on trigger words and, therefore, might remain susceptible to paraphrasing during imitation attacks.

3 Methodology

We first provide an overview of the existing EaaS watermark techniques and their benefits in defending against imitation attacks. We then introduce our paraphrasing attack and subsequently propose a new watermarking technique, *WET*.

3.1 Preliminary Background

We assume that a malicious attacker conducts an imitation attack on a victim EaaS service $\mathbb{S}_v$ based on model Θ_v. The attacker queries $\mathbb{S}_v$ to collect the embeddings (which are watermarked, unbeknownst to the attacker) for a set of input texts D_a, which will then be used for training an attack/surrogate model Θ_a. The goal of the attacker is to provide a competitive EaaS service $\mathbb{S}_a$ and they may actively employ strategies to remove or bypass the watermark. For the victim, *i.e.,* EaaS provider, it is crucial that the watermarked embeddings perform similarly to the original non-watermarked embeddings on downstream tasks. To determine whether their model has been copied, the victim will query suspicious services $\mathbb{S}_a$ to check if the returned embeddings contain the injected watermarks.
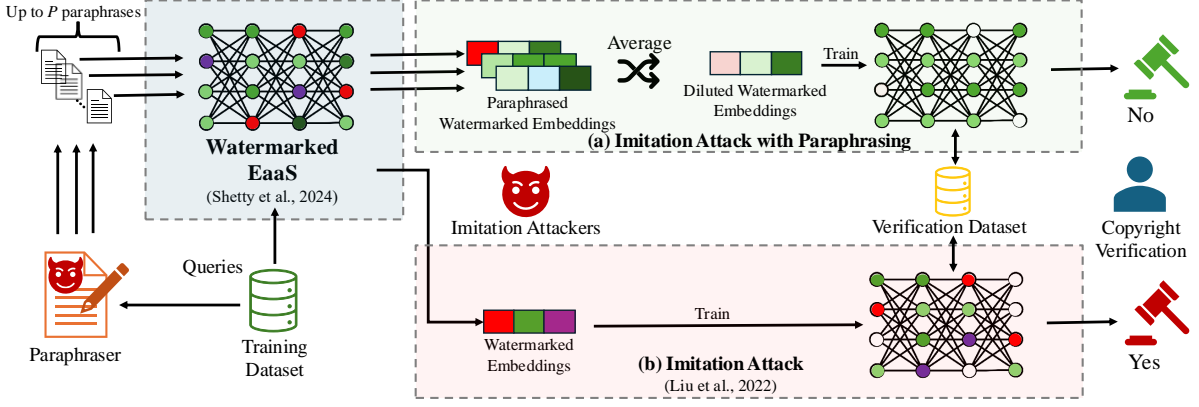
Figure 1: An overview of our paraphrasing attack, where the (a) Green area shows the EaaS watermarks (presented as the elements in Red) getting diluted due to paraphrasing and potentially bypassed. On the contrary, the (b) Red area denotes a traditional imitation attack without paraphrasing, leading to copyright infringement.

3.2 Paraphrasing Attack

We propose generating multiple paraphrases and using their averaged embedding to train the surrogate model so as to bypass the detection of embedding watermark; see Figure 1 for an illustration.

Formally, we generate P paraphrased texts $S_P = \{s^1, \cdots, s^P\}$ given an input text s. Next, we query the EaaS $\mathbb{S}_v$ to get their embeddings and aggregate them into a single embedding through averaging ($\text{avg}(\cdot)$):

$$E_a = \left\{\mathbb{S}_v(s^i)\right\}_{i=1}^P, \ \text{avg}(E_a) = \sum_{e \in E_a} e/|E_a|.$$

We will then use the aggregated embeddings $\text{avg}(E_a)$ for training the surrogate model in an imitation attack (illustrated as the "Diluted Watermarked Embeddings" in Figure 1). To measure the success of this paraphrasing attack, we will evaluate verification accuracy (*verifiability*) and downstream task performance (*utility*), as detailed in Section 4.1. The attack would be considered successful if the downstream task performance is high and verification accuracy is low.

We provide a theoretical validation in the Appendix B.1 to show that averaging paraphrase embedding reduces the possibility of observing embedding samples with high watermark weights. We validate this hypothesis empirically in Section 4.4.

3.3 *WET* Defence

Next, we introduce **W**atermarking **E**aaS with Linear **T**ransformation (*WET*), a new embedding watermarking protocol (shown in Figure 2) that is designed to be robust against paraphrasing attacks.

The core idea is to use a preset linear transformation matrix $\mathbf{T}$ (unknown to the attacker) to transform an original embedding e_o into a watermarked embedding e_p (Figure 2 left part). Our watermarking technique discards the original elements and retains only the transformed ones, which makes the watermark more difficult to be detected.[2] To check for the watermark in a copied embedding e'_p (produced by the surrogate model), we apply the *inverse* of the linear transformation matrix $\mathbf{T}^+$ to it and assess whether the recovered embedding e'_o is similar to the original embedding e_o (Figure 2 right part). An important consideration is constructing the transformation matrix in a way that balances the trade-off between utility and verifiability.

Watermark Injection. Given a transformation matrix $\mathbf{T}$, we (i) multiply it with the original embedding e_o and (ii) normalise it to a unit vector,

$$e_p = \text{Norm}(\mathbf{T} \cdot e_o) = \frac{\mathbf{T} \cdot e_o}{||\mathbf{T} \cdot e_o||}. \quad (1)$$

Note that, unlike previous EaaS watermarks, our approach does not rely on trigger words for watermark injection. Instead, we watermark all the output embeddings, leading to denser signals and making it more difficult to bypass while maintaining the same level of utility for EaaS users.

Matrix Construction. One challenge for *WET* is in designing the transformation matrix. In the watermark verification process, we perform a reverse

---

[2]We initially explored embedding dimension obfuscation by adding new dimensions mixed with the original ones, inspired by Yan et al. (2023), but found that these obfuscated dimensions could be easily identified using feature correlation and feature importance techniques; details in Appendix C.5.
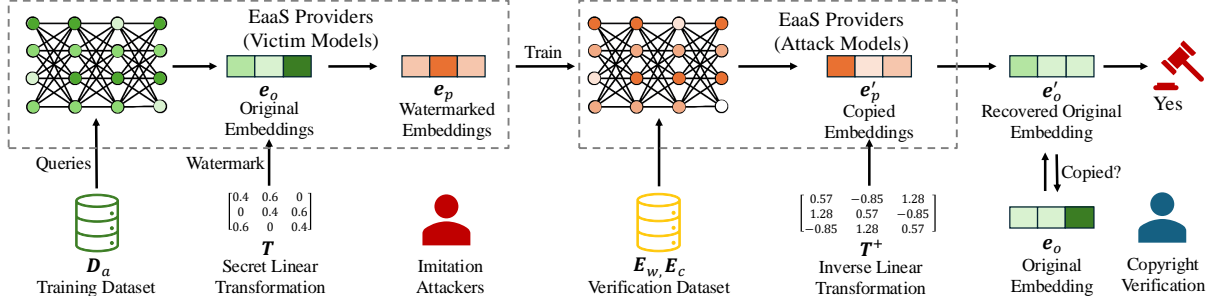
Figure 2: An overview of the workflow for *WET*. The left block illustrates the watermarking process using a secret transformation matrix $\mathbf{T}$. The right block follows the watermark verification process, employing the pseudoinverse of the transformation matrix $\mathbf{T}^+$. The recovered embedding e_o' and the original embedding e_o are compared in copyright verification.

transformation (Equation 3) to recover the original embeddings from the watermarked ones. There, it is crucial that the transformation matrix is both full-rank and well-conditioned to allow for accurate pseudoinverse computation (Strang, 2016). To meet the requirement, we adopt circulant matrices (Gray et al., 2006) to ensure these properties. The first row is generated randomly, and subsequent rows are circulations of the initial row. The positions and values of non-zero entries in the first row are selected randomly (see "Secret Linear Transformation" matrix in Figure 2). The circulant matrix is full-rank if the first row has non-zero fast Fourier transform (FFT) values (corresponding to eigenvalues of circulant matrix), which is more probable by our row construction (Gray et al., 2006). Moreover, full-rank guarantees a lower condition number, which is beneficial for computing well-conditioned pseudoinverses (Strang, 2016). Additionally, cycle shifts ensure that all dimensions in the original embedding contribute equally to the watermark. Algorithm 1 details the generation of the transformation matrix. The two hyperparameters to be considered are w and k. w represents the number of dimensions of the watermarked embeddings. k represents the number of original dimensions used to compute a dimension in the watermarked embeddings. We explore and discuss alternative matrix constructions by relaxing various properties (like circularity, randomness, and others) in Appendix C.3.

Robustness to Paraphrasing Attacks. We now show theoretically how the linear transformation used in *WET* is robust against paraphrasing during imitation attacks and the watermark is still learned by the surrogate model.

Theorem 1 (Robustness of *WET*) *Given P wa-*

termarked embeddings, $e_p^i = f(e_o^i)$, where f is a linear transformation function, as defined in Equation 1 and $i \in [1..P]$. The average of these paraphrased embeddings is equivalent to a linear transformation of a pseudo-aggregation of the original embeddings, $\hat{e}_o^i$, i.e.,

$$avg(f(\{e_p^i\}_{i=1}^P)) = f(avg(\{\hat{e}_o^i\}_{i=1}^P)). \quad (2)$$

Proof

$$avg(f(\{e_p^i\}_{i=1}^P)) = avg(\{\underbrace{Norm(\mathbf{T} \cdot e_o^i)}_{\triangleq \alpha_i \cdot \mathbf{T} \cdot e_o^i}\})$$

$$= \mathbf{T} \cdot \frac{1}{P} \sum_{i=1}^P \underbrace{\alpha_i \cdot e_o^i}_{\triangleq \hat{e}_o^i} = \mathbf{T} \cdot avg(\{\hat{e}_o^i\}_{i=1}^P)$$

$$= f(avg(\{\hat{e}_o^i\}_{i=1}^P)).$$

The transformation $\mathbf{T}$ should be consistent regarding the aggregation on the pseudo embedding $\hat{e}$ though distorted by $\alpha_i = 1/||\mathbf{T} \cdot e_o^i||$. Given Theorem 1, the *WET* watermark key (*i.e.,* $\mathbf{T}$) will not be removed through the aggregation of paraphrased embeddings.

Watermark Verification. The verification process attempts to decode the watermarked embedding using the authentic $\mathbf{T}$ and verify whether it matches the original embedding. That is, we first apply the pseudoinverse of the transformation matrix $\mathbf{T}^+$ to the copied embedding e_p' to produce recovered original embedding e_o':

$$e_o' = \mathbf{T}^+ \cdot e_p', \quad (3)$$

where $\mathbf{T}^+$ is Moore-Penrose inverse (*a.k.a* pseudoinverse) (Strang, 2016). When $\mathbf{T}$ has linearly independent rows (guaranteed by the circulant matrix), then $\mathbf{T}^+$ is a right inverse, *i.e.,* $\mathbf{T} \cdot \mathbf{T}^+ = \mathbf{I}_w$.

Algorithm 1 Transformation Matrix Generation.

Require:
    n: # original dimensions
    k: # original dimensions used in transformation
    w: # watermarked embedding dimensions

1: **function** MATRIX_GEN(n, k)
2:     Initialise $\mathbf{T} \leftarrow \phi$
3:     row $\leftarrow$ ROW_GEN(n, k)   ▷ $\mathbb{R}^{1 \times n}$
4:     cnt $\leftarrow 0$
5:     **for** each $i = 1, 2, \cdots, w$ **do**   ▷ Circular
6:         $\mathbf{T}[i] \leftarrow$ row
7:         row $\leftarrow$ Roll(row)
8:         cnt $+= 1$
9:         **if** cnt $== n$ **then**   ▷ Re-generate
10:             row $\leftarrow$ ROW_GEN(n, k)
11:             cnt $\leftarrow 0$
12:         **end if**
13:     **end for**
14:     **return T**   ▷ $\mathbb{R}^{w \times n}$
15: **end function**

16: **function** ROW_GEN(n, k)
17:     Initialise row $\leftarrow$ Zeroes(n)
18:     positions $\leftarrow$ Sample(n, k)   ▷ Correlations
19:     **for** p in positions **do**
20:         row[p] $\sim U(0, 1)$ ▷ Random
21:     **end for**
22:     row $\leftarrow$ Norm(row)
23:     **return** row
24: **end function**

To check the transformation aligns with the authentic watermark process, we measure the similarity between the recovered embedding e'_o by the attack model and the original embedding e_o by the victim model. If the attacker has copied the victim model, then the similarity score should be high. In our experiments, we use cosine similarity for measuring similarity.[3]

4 Experiments

4.1 Metrics

To evaluate the effectiveness of the paraphrasing attack and our new watermarking method, we use the following metrics to assess downstream task utility and watermark verifiability.

Downstream Task Utility. Using the EaaS embeddings as input, we build multi-layer perceptron classifiers for a range of classification tasks and evaluate the accuracy (ACC) and F_1-score (F1) performance. This evaluation serves as an indicator of whether watermarking degrades the quality of the original embeddings: ideally, there should be minimal performance difference between the watermarked and original embeddings.

Watermark Verifiability. To quantify verification performance, we create a verification dataset containing two sets of embeddings: (i) watermark set E_w (which contains watermarked embeddings) and (ii) contrast set E_c (which contains watermarked embeddings generated with a different transformation matrix).[4] The goal is that the verification process should have a high accuracy in identifying E_w without confusing it with E_c.

Given the two sets, we compute the average cosine similarity between the recovered embeddings (e'^i_o from Equation 3) and original embeddings (e^i_o) and then take their difference:

$$\Delta_{cos} = cos_{\text{avg}}(S_w) - cos_{\text{avg}}(S_c),$$
$$cos_{\text{avg}}(S) = \frac{1}{|S|} \sum_{i=1}^{|S|} cos(e'^i_o, e^i_o), \quad (4)$$

where the sets of recovered and original embedding pairs are constructed by:

$$S_w = \left\{ (e'^i_o, e^i_o) | e'^i_p \in E_w \right\}_{i=1}^{|E_w|},$$
$$S_c = \left\{ (c'^i_o, c^i_o) | c'^i_p \in E_c \right\}_{i=1}^{|E_c|}. \quad (5)$$

Based on the cosine similarity scores, we also compute the area under the receiver operating characteristic curve (AUC) (Mitchell et al., 2023), which gives us a more intuitive interpretation of verifiability: an AUC of 100% means the watermark set and contrast set are perfectly separable. Additional details regarding the evaluation dataset are provided in Appendix A.5.

4.2 Datasets

We use AG News (Zhang et al., 2015), MIND (Wu et al., 2020), SST2 (Socher et al., 2013), and Enron (Metsis et al., 2006) in our experiments. Table 1 provides the statistics for these datasets. These

---

[3]Although l_2 distance is also used as a similarity metric conventionally, we found similar performances in our experiments and have omitted its results for brevity.

[4]Note that for EmbMarker and WARDEN, we follow the original studies where the contrast set are non-watermarked embeddings (*i.e.,* embeddings where their input text have no trigger words).

(a) Enron   (b) SST2
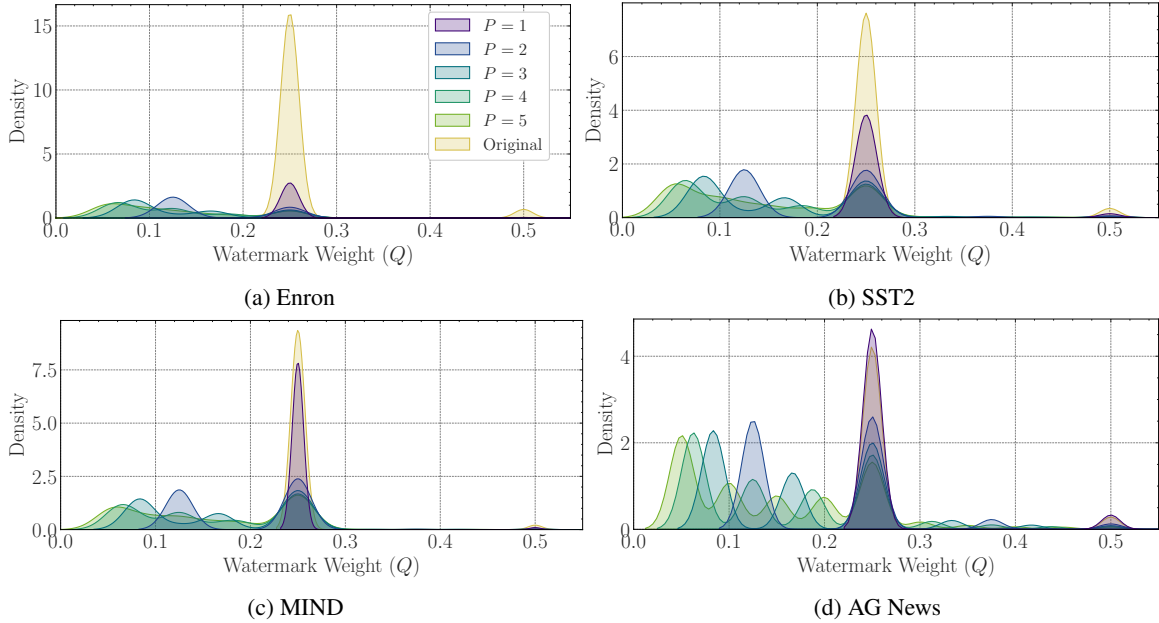
(c) MIND   (d) AG News

Figure 3: Watermark weight analysis for different datasets (in subcaption) based on GPT-3.5 paraphrases. In general, aggregating watermarked embeddings from more paraphrases (larger P) reduces the watermark weights.

datasets are used to evaluate a variety of downstream classification performances, covering tasks from spam classification (Enron) to sentiment classification (SST2) to news recommendation and classification (AG News and MIND).

Dataset	# Class	# Train	# Test	Avg. Len.
Enron	2	31,716	2,000	34.57
SST2	2	67,349	872	54.17
MIND	18	97,791	32,592	66.14
AG News	4	120,000	7,600	236.41

Table 1: The statistics of datasets.

4.3 Experimental Settings

In terms of model configurations and hyperparameters, we largely follow the experimental settings by Shetty et al. (2024). To simulate the imitation attacks, we use GPT-3 text-embedding-002 (OpenAI, 2022) as the victim EaaS to retrieve the original (non-watermarked) embeddings and BERT (Devlin et al., 2019) as backbone model for the attacker's surrogate model.[5] We experiment with three paraphrase methods: (i) prompting GPT-3.5-turbo (prompts are given in Appendix A.1); (ii) using an specialised paraphrasing model, DIPPER (Krishna et al., 2024) (configuration is detailed in

Appendix A.2); and (iii) round-trip translation using NLLB (Costa-jussà et al., 2022), a multi-lingual translation model. We present more details for these models (*e.g.,* pivot languages and translation setups) in Appendix A.3. When paraphrasing, we first generate P unique paraphrases for each input text and then filter out bad paraphrases based on their cosine similarity with the original input text (details in the Appendix B.7). Consequently, on average, we have 2.68, 3.30, 3.41, and 4.89 paraphrases (using GPT-3.5) across Enron, SST2, MIND, and AG News, respectively. Appendix B.8 presents more analyses on the quality of the paraphrases. For our main experiments, we set $w = n$ (recall that n and w are the number of dimensions in the original and watermarked embeddings) to avoid compressing the embeddings. We investigate different values of w in Section 4.5.

4.4 Attack Experiments

We now present the results of our paraphrasing attack against EmbMarker and WARDEN.

Watermark Weight Analysis. Figure 3 shows the watermark weight distribution with varying numbers of paraphrases P for EmbMarker and WARDEN.[6] As P increases, we observe that the watermark weight reduces, suggesting that the more paraphrases incorporated, the more diluted

---

[5]All watermarking techniques (EmbMarker, WARDEN and *WET*) inject watermarks post-hoc into the embeddings produced by the API calls.

[6]EmbMarker and WARDEN use the same watermark weight so these results apply to both methods.

Method	ACC ↑	F1 ↑	Δ_{cos} ↓	AUC ↓
WARDEN	94.50±0.34	94.50±0.34	5.20±0.34	97.40±0.54
+GPT-3.5 Attack	92.81±0.21	92.81±0.21	0.70±0.22	68.90±7.79
+DIPPER Attack	91.34±0.52	91.33±0.52	0.46±0.11	67.50±5.56
+NLLB Attack	93.35±0.23	93.35±0.23	0.65±0.12	71.95±4.04
(a) Enron				
WARDEN	93.10±0.12	93.10±0.12	2.57±1.19	86.75±6.20
+GPT-3.5 Attack	92.75±0.15	92.75±0.15	0.93±0.09	75.90±2.91
+DIPPER Attack	91.70±0.27	91.66±0.27	0.90±0.17	71.95±2.69
+NLLB Attack	92.57±0.09	92.55±0.08	1.06±0.19	69.35±2.94
(b) SST2				
WARDEN	77.31±0.08	51.47±0.23	5.27±0.17	98.10±0.51
+GPT-3.5 Attack	77.01±0.05	51.24±0.22	1.85±0.21	79.40±3.08
+DIPPER Attack	76.86±0.07	50.54±0.17	3.47±0.12	96.70±0.51
+NLLB Attack	76.64±0.10	50.36±0.11	3.89±0.06	97.80±0.33
(c) MIND				
WARDEN	93.51±0.13	93.50±0.13	14.46±0.68	100.00±0.00
+GPT-3.5 Attack	92.28±0.12	92.26±0.13	7.23±0.34	100.00±0.00
+DIPPER Attack	92.50±0.11	92.48±0.11	11.04±0.40	100.00±0.00
+NLLB Attack	92.70±0.10	92.69±0.10	10.56±0.44	100.00±0.00
(d) AG News				

Table 2: The performance of paraphrasing attack against WARDEN on SST2, MIND, AG News, and Enron. From an attacker's perspective, ↑ means higher metrics are better and ↓ means lower metrics are better.

Method	ACC ↑	F1 ↑	Δ_{cos} ↑	AUC ↑
WET	94.58±0.21	94.58±0.21	85.67±6.92	100.00±0.00
+GPT-3.5 Attack	92.73±0.25	92.73±0.25	83.58±6.43	100.00±0.00
+DIPPER Attack	91.37±0.10	91.36±0.10	83.11±6.48	100.00±0.00
+NLLB Attack	93.24±0.24	93.24±0.24	84.28±6.04	100.00±0.00
(a) Enron				
WET	93.07±0.40	93.07±0.40	88.97±6.62	100.00±0.00
+GPT-3.5 Attack	92.38±0.34	92.38±0.34	87.02±6.32	100.00±0.00
+DIPPER Attack	91.77±0.66	91.74±0.67	86.59±6.33	100.00±0.00
+NLLB Attack	92.75±0.34	92.74±0.34	87.78±6.27	100.00±0.00
(b) SST2				
WET	77.11±0.08	51.03±0.26	87.74±6.17	100.00±0.00
+GPT-3.5 Attack	76.72±0.05	50.62±0.25	87.44±6.17	100.00±0.00
+DIPPER Attack	76.58±0.08	49.99±0.23	86.81±5.90	100.00±0.00
+NLLB Attack	76.47±0.14	49.85±0.26	87.54±5.91	100.00±0.00
(c) MIND				
WET	93.15±0.08	93.14±0.08	88.35±6.6	100.00±0.00
+GPT-3.5 Attack	92.22±0.10	92.20±0.10	88.02±6.14	100.00±0.00
+DIPPER Attack	92.46±0.18	92.45±0.18	87.79±6.14	100.00±0.00
+NLLB Attack	92.43±0.08	92.42±0.08	88.44±5.91	100.00±0.00
(d) AG News				

Table 3: The performance of *WET* watermark for different scenarios on SST2, MIND, AG News, and Enron datasets. From a defender's perspective, ↑ means higher metrics are better. All the metrics are in %.

the watermark. In other words, these results paraphrasing might be able to circumvent the watermark detection for an imitation attack. We present other attack setups in Appendix Figures 6 and 7.

Utility and Verifiability Evaluation. Table 2 presents the utility and verifiability of WARDEN[7] under paraphrasing attack.[8] In terms of utility, the paraphrasing attack only has a small negative impact on downstream performance. In terms of verifiability, for Δ_{cos} we see the numbers drop significantly after paraphrasing, showing that it is now harder to detect the watermark. AUC tells a similar story, with one exception: watermarks for AG News are still verifiable, suggesting the paraphrasing attack is less effective for this dataset. We suspect this is because AG News has much longer texts (see Table 1), which means paraphrasing has the possibility of introducing new trigger words not in the original text. This is supported by our theoretical analyses (Section B.1), which showed that although with paraphrasing we reduce the probability of higher watermark weights, at the same time this effect diminishes with longer text. As an attacker has the freedom to select their training strategy, this

means they can technically still exploit this paraphrasing vulnerability by using shorter texts when cloning the victim model.

Ablation Study. In the Appendix B, we present additional studies to examine the impact of various factors, such as the number of watermarks (Appendix B.2), numbers of paraphrases (Appendix B.3), non-watermark case (Appendix B.4), attack model size (Appendix B.5), and training data size (Appendix B.6).

4.5 Defence Experiments

Watermark Performance. We present the utility and verifiability performance of *WET* against paraphrasing attacks in Table 3. If we compare *WET* to WARDEN in Table 2, their downstream performance is about the same, suggesting they are all competitive in terms of maintaining utility. However, *WET* is better when it comes to verifiability, as its AUC is 100% in all cases. Examining the impact of the paraphrasing attack, *WET* is a clear winner here, as all verifiability metrics see minimal changes (most importantly, AUC is still 100%). These results empirically validate that *WET* is not susceptible to paraphrasing attacks. We now present additional analyses to understand the impact of hyper-parameters k and w. For these, we only look at utility performance as verifiability does not change based on these hyper-parameters.

---

[7]The number of watermarks, R, is 4 for this experiment. Results of the impact of different R are in Appendix B.2.

[8]We omit the results for EmbMarker here as they show similar observations; but that results are included in Appendix Table 5.
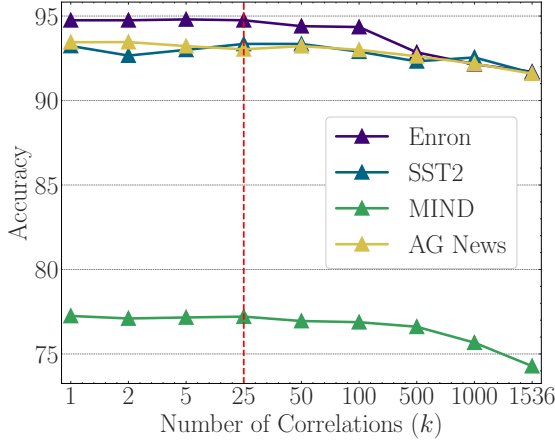
Figure 4: Impact of different values of correlations (k) on watermark utility. We ignore verifiability as they are always perfect (*i.e.,* 100%). The red vertical dashed line represents our chosen value ($k = 25$).

Number of correlations (k). In Figure 4, we can see that for higher values of k (> 100), we start seeing degradation in the watermarked embedding utility. When we consider more original embedding dimensions for calculating watermarked embedding, the increased complexity introduces confusion, making it harder for the surrogate model to learn the underlying semantic properties of the embeddings. Hence, we chose $k = 25$ in our experiments. A more comprehensive table with full results is provided in Appendix Table 13.
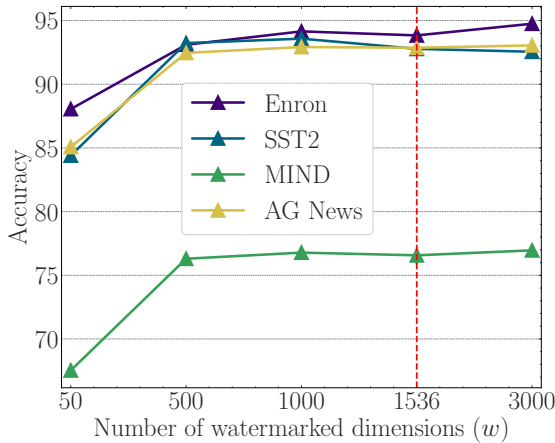

Figure 5: Impact of different values of watermark dimensions (w) on watermark utility. The red vertical dashed line represents our chosen value ($w = 1536$).

Number of watermarked dimensions (w). We can observe from Figure 5 that smaller values might also work. This demonstrates another benefit of our *WET* technique: it can also be used for compression. That said, utility is only measured using

simple classification tasks (following prior studies) and as such these results may be different if the embeddings are used for more complex tasks. As such, we use $w = n$ in our experiments. For more results on different values of w, see Appendix Table 14.

Ablation Study. In our watermark verification process, which includes reverse transformation, we evaluate the resilience of *WET* to perturbations. Our findings show that *WET* remains verifiable even under significant utility loss, highlighting its robustness; see Appendix C.1 for results. In Appendix C.2, we demonstrate that *WET* requires very few samples (even one) for watermark verification; a contrast to EmbMarker and WARDEN which require multiple samples for verification. In Appendix C.3 we present additional results with different configurations of the transformation matrix, a critical component of *WET*. Lastly, in Appendix C.4 we show that *WET* is not affected by the attack model size.

5 Conclusion

We highlight the vulnerabilities of existing EaaS watermarks against paraphrasing in an imitation attack. Our approach involves generating multiple paraphrases and combining their embeddings, which effectively reduces the impact of trigger words and thereby removes the watermark. To address this shortcoming, we devise a simple watermarking technique, *WET*, which applies linear transformations to the original embeddings to generate watermarked embeddings. Our experiments demonstrate that *WET* is robust against paraphrasing attacks and has a much stronger verifiability performance. Additionally, we conduct ablation studies to assess the contribution of each component in the paraphrasing attack and *WET*.

Limitations

With the current design of the circulant transformation matrix, the matrix is compromised if an attacker manages to recover any single row in the matrix. A better approach could be to use different weights (more in Appendix C.3) for each row in the circulant matrix, but this means we would lose crucial properties such as invertibility and full rank. Therefore, we opted to retain the current design, though we acknowledge that the design can be potentially further improved.

For utility, we focus on simple classification tasks in line with existing studies; however, these

tasks may not be sufficient to fully validate embedding quality. Moving forward, we believe it is important we start exploring other more complex NLP tasks, such as retrieval and generation, to gain a deeper understanding of the true impact of introducing watermarks into embeddings.

Ethics Statement

We introduce paraphrasing as a new form of attack against EaaS watermarks. We want to clarify that our intention here is to raise awareness about this new form of attack, as we believe the first step in improving security is by exposing vulnerabilities. As a countermeasure, we therefore also introduce a new watermarking technique, *WET*, that is resilient against paraphrasing attacks.

Acknowledgement

References

Scott Aaronson. 2023. Watermarking of large language models.

Yiyi Chen, Heather Lent, and Johannes Bjerva. 2024. Text embedding inversion security for multilingual language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7808–7827, Bangkok, Thailand. Association for Computational Linguistics.

Miranda Christ, Sam Gunn, and Or Zamir. 2024. Undetectable watermarks for language models. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 1125–1139. PMLR.

Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, and 1 others. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Robert M Gray and 1 others. 2006. Toeplitz and circulant matrices: A review. *Foundations and Trends® in Communications and Information Theory*, 2(3):155–239.

Xuanli He, Lingjuan Lyu, Chen Chen, and Qiongkai Xu. 2022a. Extracted BERT model leaks more information than you think! In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1530–1537, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xuanli He, Lingjuan Lyu, Lichao Sun, and Qiongkai Xu. 2021. Model extraction and adversarial transferability, your BERT is vulnerable! In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2006–2012, Online. Association for Computational Linguistics.

Xuanli He, Qiongkai Xu, Lingjuan Lyu, Fangzhao Wu, and Chenguang Wang. 2022b. Protecting intellectual property of language generation apis with lexical watermark. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10758–10766.

Xuanli He, Qiongkai Xu, Yi Zeng, Lingjuan Lyu, Fangzhao Wu, Jiwei Li, and Ruoxi Jia. 2022c. Cater: Intellectual property protection on text generation apis via conditional watermarks. *Advances in Neural Information Processing Systems*, 35:5431–5445.

Zhiwei He, Binglin Zhou, Hongkun Hao, Aiwei Liu, Xing Wang, Zhaopeng Tu, Zhuosheng Zhang, and Rui Wang. 2024. Can watermarks survive translation? on the cross-lingual consistency of text watermark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4115–4129, Bangkok, Thailand. Association for Computational Linguistics.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *International Conference on Machine Learning*, pages 17061–17084. PMLR.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. 2024. On the reliability of watermarks for large language models. In *The Twelfth International Conference on Learning Representations*.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2024. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Advances in Neural Information Processing Systems*, 36.

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 *International Conference on Learning Representations*.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2024. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, and 13 others. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yupei Liu, Jinyuan Jia, Hongbin Liu, and Neil Zhenqiang Gong. 2022. Stolenencoder: Stealing pretrained encoders in self-supervised learning. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 2115–2128, New York, NY, USA. Association for Computing Machinery.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

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

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *International Conference on Machine Learning*, pages 24950–24962. PMLR.

John Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander Rush. 2023. Text embeddings reveal (almost) as much as text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12448–12460, Singapore. Association for Computational Linguistics.

OpenAI. 2022. New and improved embedding model.

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*, pages 4954–4963.

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 the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7653–7668.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2023. Can ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*.

Philip Sedgwick. 2012. Pearson's correlation coefficient. *Bmj*, 345.

Anudeex Shetty, Yue Teng, Ke He, and Qiongkai Xu. 2024. WARDEN: Multi-directional backdoor watermarks for embedding-as-a-service copyright protection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13430–13444, Bangkok, Thailand. Association for Computational Linguistics.

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 *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Harold Somers. 2005. Round-trip translation: What is it good for? In *Proceedings of the Australasian Language Technology Workshop 2005*, pages 127–133, Sydney, Australia.

G. Strang. 2016. *Introduction to Linear Algebra*. Wellesley.

Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*, pages 601–618.

Eric Wallace, Mitchell Stern, and Dawn Song. 2020. Imitation attacks and defenses for black-box machine translation systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5531–5546, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven

Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3597–3606, Online. Association for Computational Linguistics.

Qiongkai Xu and Xuanli He. 2023. Security challenges in natural language processing models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*, pages 7–12, Singapore. Association for Computational Linguistics.

Qiongkai Xu, Xuanli He, Lingjuan Lyu, Lizhen Qu, and Gholamreza Haffari. 2022. Student surpasses teacher: Imitation attack for black-box NLP APIs. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2849–2860, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Yifan Yan, Xudong Pan, Mi Zhang, and Min Yang. 2023. Rethinking {White-Box} watermarks on deep learning models under neural structural obfuscation. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 2347–2364.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Shujian Huang, Lingpeng Kong, Jiajun Chen, and Lei Li. 2024. Multilingual machine translation with large language models: Empirical results and analysis. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2765–2781, Mexico City, Mexico. Association for Computational Linguistics.

Appendix

A Experimental Settings

A.1 Prompt-Based Paraphrasing Attacks

We utilise the prompts chosen through the below analysis and query the `gpt-3.5-turbo-0125` model using the maximum token length of 1000 and temperature of 0.7. One must note that for all paraphrasing attack setups, it is up to P paraphrases. We query the models to generate P paraphrases; however, we do not make repeated queries to ensure we have P paraphrases for cases where it generates fewer paraphrases.

Prompts. We evaluated two prompts:

PROMPT 1 (Kirchenbauer et al., 2024): *"As an expert copy-editor, please rewrite the following text in your own voice while ensuring that the final output contains the same information as the original text and has roughly the same length. Please paraphrase all sentences and do not omit any crucial details. Additionally, please take care to provide any relevant information about public figures, organisations, or other entities mentioned in the text to avoid any potential misunderstandings or biases."*

PROMPT 2 (He et al., 2024): *"You are a helpful assistant to rewrite the text. Rewrite the following text:"*

We use PROMPT 2 in our experiments unless stated. Performance of PROMPT 1 was evaluated for Enron, it was subpar compared to PROMPT 2. It was because PROMPT 1 could explain and expand on short (few or single words) input text. This leads to a significant deviation from the original text. Moreover, increases the chances of incorporating the trigger words.

Attack Cost. The approximate total number of tokens for all the datasets is Enron (377K), SST2 (1M), MIND (2M), and AG News (7M). Considering $P = 5$ and assuming similar tokens in the output, the expected cost of generating paraphrases using GPT-3.5 (input - \$0.50 / 1M tokens and output - \$1.50 / 1M tokens) would be just under \$105.

A.2 DIPPER Paraphrasing Attacks

We employ DIPPER (Krishna et al., 2024), an explicitly trained paraphraser with hyperparameters (`lex` and `div`), to control the paraphrasing quality. As per findings in Krishna et al. (2024), DIPPER performs at par with GPT-3.5 models in terms of controlling the diversity and quality of paraphrases. We adopt a moderate setting for all our experiments: `lex = 40` and `div = 40`. It still ensures significant changes to the text but, at the same time, maintains a high quality of paraphrases.

A.3 Round-Trip Translation Paraphrasing Attacks

Language	IDO 639-I	ISO 639-2/T	Language family
Chinese (Simpl)	zh	zho_simpl	Sino-Tibetan
Japanese	ja	jpn	Other
French	fr	fra	Indo-European-Romance
German	de	deu	Indo-European-Germanic
Hindi	hi	hin	Indo-European-Indo-Aryan

Table 4: For each language we use in RTT, we list its language name, ISO code and language family (Zhu et al., 2024).

Round-trip translation (RTT) involves translating text to another language and then back-translating to the original language (*e.g.,* English → German → English). It is commonly used for evaluating machine translation systems because the original and resulting text could vary significantly (Somers, 2005). We explore translations (represented in Table 4) for languages considerably different from English (our original language), such as Chinese, German, and others, covering a diverse group such that the translated text will probably have more modifications. GPT-3.5 is still not SOTA for multilingual translations, as found in Zhu et al. (2024). Hence, we use the 1.3B NLLB model variant, an open-source multilingual model.

A.4 Baseline Method Details and Hyperparameters

For fair comparisons, we use the original default settings of the baseline methods unless specified otherwise.

EmbMarker. The size of the trigger word set is 20, and the maximum number of trigger words m is 4, with a frequency interval for trigger words of [0.5%, 1%]. We use BERT (Devlin et al., 2019) as the backbone, with a two-layer feed-forward network for imitation attacks and a mean squared error (MSE) loss for training.

WARDEN. The settings remain the same as described for EmbMarker above, with the number of watermarks (R) set to 4.

A.5 Definition of Positive and Negative Test Samples for Watermark Verification

These are used for the calculation of AUC metrics.

"Positive" Samples. In all the *WET* experiments, these are (copied) embeddings E_w returned by the copied model $\mathbb{S}_v$ that should be classified as watermarked embeddings. Whereas, for paraphrasing attack experiments, these are embeddings from the backdoor (all trigger words) verification dataset.

"Negative" Samples. In all the *WET* experiments, these are contrast (copied) embeddings E_c returned by another copied model $\mathbb{S}_v^*$. We train another copied model following a similar process using a different transformation matrix and use this model's embeddings as the non-watermarked embeddings to make it more challenging. Similarly, in paraphrasing attack experiments, these are embeddings from the benign (no trigger words) verification dataset.

A.6 Code and Compute Details

We expand on the watermarking implementation by Shetty et al. (2024). We make extensive use of the Huggingface Transformers (Wolf et al., 2020) framework and AdamW (Loshchilov and Hutter, 2019) for models and datasets library (Lhoest et al., 2021) for data assessed in this work. To spur future research in this area, we intend to make the embeddings and code available post-acceptance.

All experiments were conducted using a single A100 GPU with CUDA 11.7 and PyTorch 2.1.2. To ensure that the impact of the watermarking technique is isolated from other variables, we assume that both the victim model and imitators utilize the same datasets. Additionally, we presume that the extracted model is trained solely on the watermarked outputs of the victim model.

B Paraphrasing Attack Analyses

In this section, we perform detailed analysis and ablation studies for paraphrasing attack.

B.1 Analysis of Watermarking Weight Distribution after Paraphrasing

We analyse the impact of paraphrasing on watermarking weights in a simplified setting as follows:

- Each token has low probability $\mathbb{P}_t$ of being in the trigger words set t.

Method	ACC ↑	F1 ↑	Δ_{cos} ↓	AUC ↓
EmbMarker	94.58±0.09	94.58±0.09	5.44±0.13	93.50±0.97
+GPT-3.5 Attack	92.80±0.19	92.80±0.19	-0.03±0.07	49.80±1.35
+DIPPER Attack	92.35±0.48	92.35±0.49	0.63±0.16	61.85±4.52
+NLLB Attack	93.38±0.20	93.38±0.20	0.69±0.20	65.25±3.68

(a) Enron

Method	ACC ↑	F1 ↑	Δ_{cos} ↓	AUC ↓
EmbMarker	92.89±0.25	92.89±0.25	4.05±2.70	95.04±2.30
+GPT-3.5 Attack	92.86±0.17	92.86±0.17	0.68±0.10	68.20±2.94
+DIPPER Attack	91.31±0.24	91.27±0.25	0.94±0.12	79.95±3.89
+NLLB Attack	92.66±0.55	92.64±0.55	0.76±0.11	78.20±3.60

(b) SST2

Method	ACC ↑	F1 ↑	Δ_{cos} ↓	AUC ↓
EmbMarker	77.34±0.06	51.63±0.16	3.93±0.11	93.10±0.94
+GPT-3.5 Attack	77.01±0.07	51.23±0.13	1.04±0.08	67.75±1.66
+DIPPER Attack	76.83±0.09	50.56±0.11	2.22±0.09	90.15±1.68
+NLLB Attack	76.59±0.14	50.32±0.26	2.11±0.07	85.80±1.34

(c) MIND

Method	ACC ↑	F1 ↑	Δ_{cos} ↓	AUC ↓
EmbMarker	93.47±0.12	93.47±0.12	12.53±0.67	100.00±0.00
+GPT-3.5 Attack	92.17±0.04	92.15±0.04	4.66±0.36	99.15±0.34
+DIPPER Attack	92.47±0.10	92.45±0.10	6.68±0.40	100.00±0.00
+NLLB Attack	92.76±0.13	92.74±0.13	6.3±0.35	100.00±0.00

(d) AG News

Table 5: The performance of paraphrasing attack against EmbMarker for different scenarios, similar to Table 2.

- Sentences with equal or more than one trigger acquire the same watermark weight $\lambda > 0$.

- Average of P paraphrase sentences gives watermark weight $\lambda \cdot Q_P$ and single sentence gives watermark weight $\lambda \cdot Q_S$.

As per the above assumptions, the probability of a sentence S having trigger words is

$$\mathbb{P}_S = 1 - (1 - \mathbb{P}_t)^{|S|}.$$

The weight by a single sentence is $\lambda \cdot Q_S$, where

$$Q_S \sim \text{Bernoulli}(\mathbb{P}_S). \tag{6}$$

The weight by averaged paraphrasing embeddings are equivalent to $\lambda \cdot Q_P$, where

$$Q_P = \frac{X_P}{P}, X_P = \sum_{i=1}^{P} Q_S^i, \tag{7}$$

$$X_P \sim \text{Binomial}(P, \mathbb{P}_S). \tag{8}$$

As per WARDEN setting used, trigger word frequency is $[0.5\%, 1\%]$. Therefore, assuming a generic case, $\mathbb{P}_t = 0.005$ and $|S| = 50$ (refer to Table 1), $\mathbb{P}_S = 0.222$.

When $P = 10$, $\mathbb{P}(Q_S > a) > \mathbb{P}(Q_P > a)$ for all $a > 0.3$. Similarly, when $P = 5$, $\mathbb{P}(Q_S > a) > \mathbb{P}(Q_P > a)$ for all $a > 0.4$. This demonstrates that paraphrasing will give the attackers a higher
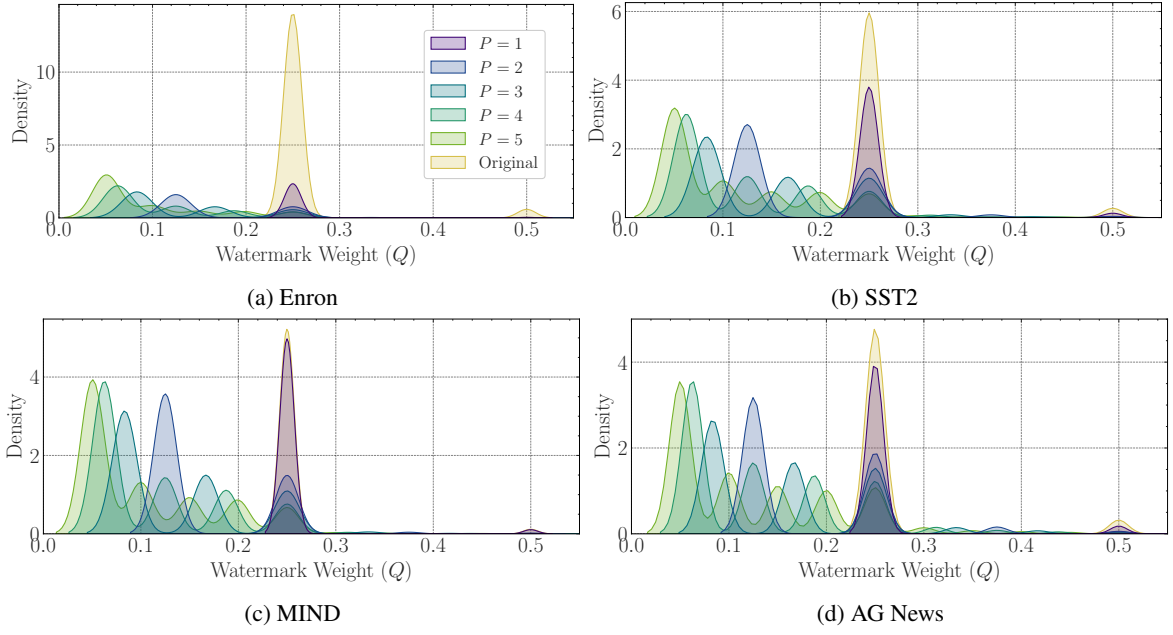
Figure 6: The analysis of watermark weight on different datasets using NLLB paraphrases. We queried NLLB up to $P = 5$ times to produce paraphrases.
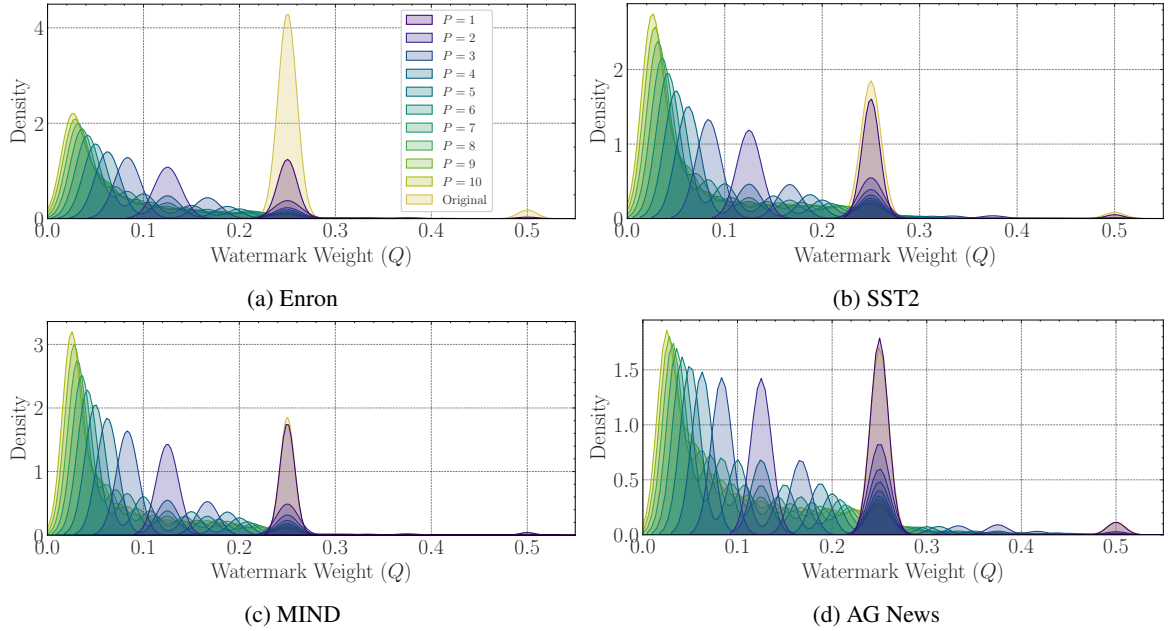

Figure 7: The analysis of watermark weight on different datasets using DIPPER paraphrases. We queried DIPPER up to $P = 10$ times to produce paraphrases.
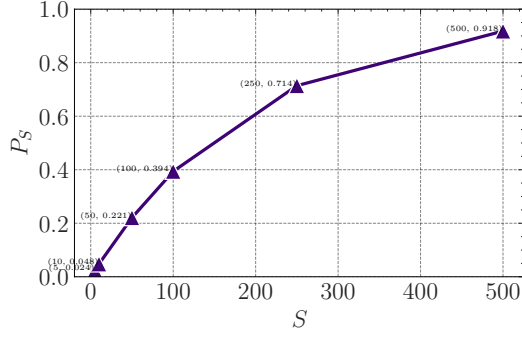
Figure 8: $\mathbb{P}_S$ for different values of sentence length $|S|$. With increasing text length (S), there are higher chances of trigger words in the text $\mathbb{P}_S$.

chance of getting watermark weights lower than a low threshold. As a result, watermarks will be diluted and neglected in the training for imitation attacks. However, this diminishes with increasing text length ($|S|$), as observed in AG News dataset in Section 4.4.

B.2 Number of Watermarks (R) in WARDEN

As expected, watermark verification performance (green and yellow lines) shows an upward trend with stable watermark utility (blue line) as shown in Figure 9.

B.3 Number of Paraphrases (P)

In Figures 10 and 11, we study the influence of the number of paraphrases (P) in paraphrasing attack. We observe there are no significant changes in attack performance with $\uparrow P$. This shows lower number of paraphrases might suffice in the attack.

B.4 Non-Watermark Case

Dataset	Paraphraser	Utility		Verifiability	
		ACC ↑	F1 ↑	Δ_{cos} ↓	AUC ↓
Enron	GPT-3.5	92.85	92.85	0.32	57.25
	DIPPER	92.00	91.99	-0.18	42.25
	NLLB	93.25	93.25	0.27	58.00
SST2	GPT-3.5	92.43	92.43	0.65	70.75
	DIPPER	91.17	91.13	0.54	67.50
	NLLB	92.20	92.18	0.22	59.50
MIND	GPT-3.5	76.97	51.29	1.09	70.75
	DIPPER	76.98	50.68	1.23	76.25
	NLLB	76.72	50.47	1.17	75.00
AG News	GPT-3.5	92.22	92.21	1.87	88.50
	DIPPER	92.51	92.51	1.28	84.50
	NLLB	92.61	92.59	1.35	85.25

Table 6: Paraphrasing attack on a non-watermarked victim model.

It will be unknown to an attacker whether the model they are attempting to copy has watermarks. Table 6 demonstrates the suitability of paraphrasing attack by causing minimal degradation in the utility and verifiability metrics.

B.5 Impact of Attack Model Size

Dataset	Size	Utility		Verifiability	
		ACC ↑	F1 ↑	Δ_{cos} ↓	AUC ↓
Enron	Small	92.60	92.60	0.12	57.75
SST2		92.55	92.55	0.37	63.75
MIND		76.93	51.17	2.18	83.50
AG News		92.42	92.40	8.61	100.00
Enron	Base	92.45	92.45	0.57	66.50
SST2		92.78	92.77	0.82	75.25
MIND		76.99	51.48	2.15	82.25
AG News		92.39	92.38	7.54	100.00
Enron	Large	92.70	92.70	0.39	61.50
SST2		93.12	93.12	0.31	62.50
MIND		76.95	51.34	2.16	78.75
AG News		92.51	92.49	7.29	100.00

Table 7: The impact of extracted model size on paraphrasing attack performance.

We assess whether our attack's performance varies with the attacker model's size. We conducted experiments for GPT-3.5 paraphrasing attack using small, base, and large variants of the BERT (Devlin et al., 2019) model to test this. The results, summarised in the Table 7, demonstrate that the attack consistently circumvents the watermark, regardless of the model size.

B.6 Impact of Scaling Train Dataset

Dataset	Type	Utility		Verifiability	
		ACC ↑	F1 ↑	Δ_{cos} ↓	AUC ↓
Enron	GPT-3.5	95.30	95.30	6.65	98.50
	DIPPER	95.30	95.30	8.47	99.50
	NLLB	94.95	94.95	8.64	99.25
SST2	GPT-3.5	93.35	93.34	6.69	96.25
	DIPPER	92.66	92.65	8.73	99.50
	NLLB	93.23	93.23	7.36	98.25
MIND	GPT-3.5	77.06	52.07	12.74	100.00
	DIPPER	77.23	55.46	15.58	100.00
	NLLB	77.12	56.89	14.97	100.00
AG News	GPT-3.5	93.11	93.10	19.68	100.00
	DIPPER	93.59	93.58	19.26	100.00
	NLLB	93.39	93.39	18.97	100.00

Table 8: The impact of scaling up the dataset with paraphrases instead of averaging the paraphrased embeddings in paraphrasing attack.
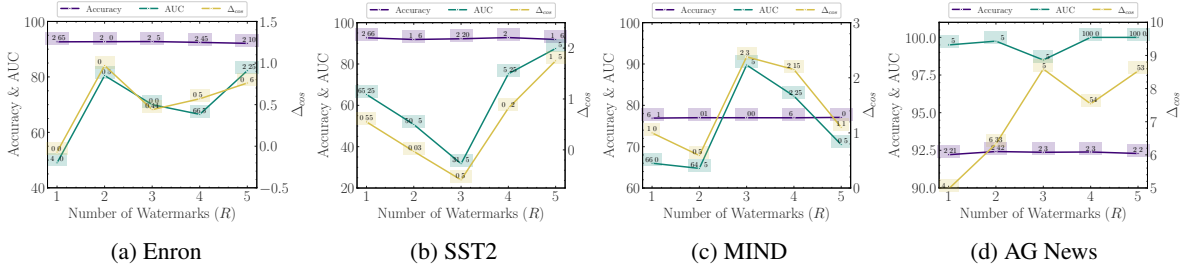
Figure 9: GPT-3.5 paraphrase attack performance different number of watermarks (R) for all the datasets.
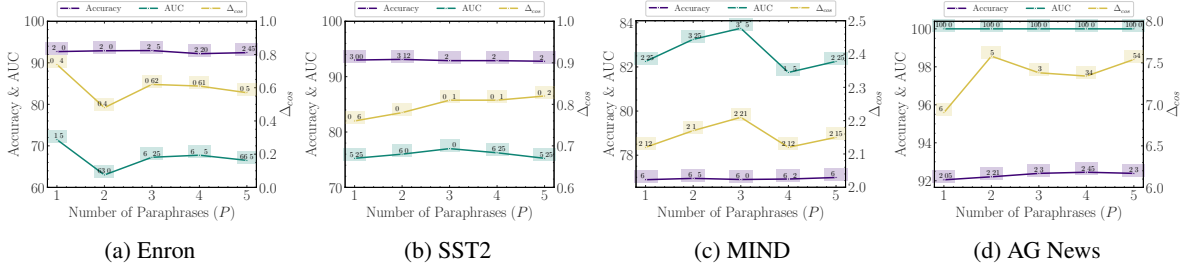
Figure 10: GPT-3.5 paraphrase attack performance different number of paraphrases (P) for all the datasets.
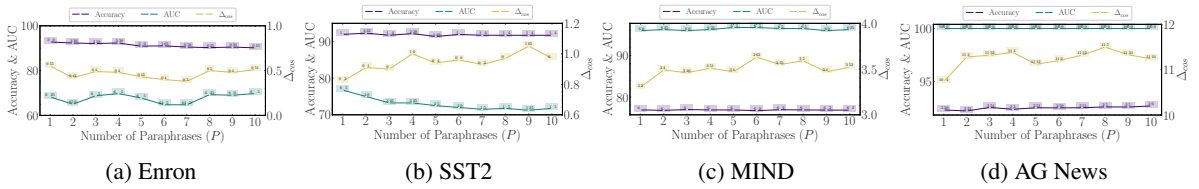
Figure 11: DIPPER paraphrase attack performance different number of paraphrases (P) for all the datasets.
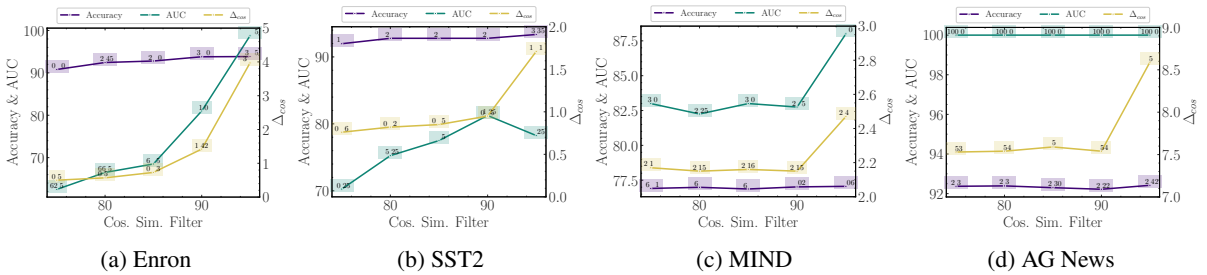
Figure 12: GPT-3.5 paraphrase attack performance using different cosine similarity filters for all the datasets.
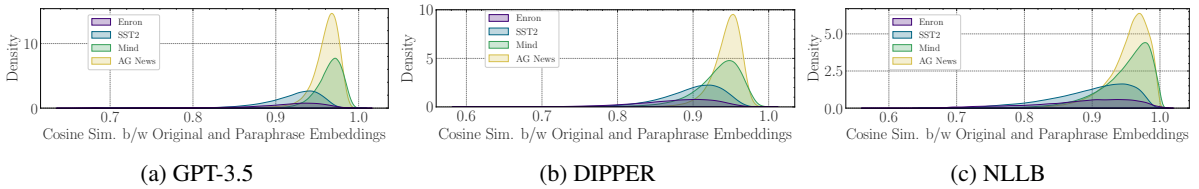
Figure 13: Cosine similarity between original and paraphrased embeddings using different paraphrases (as denoted in the captions).

A potential confound is that creating multiple paraphrases effectively increases the training data size during the imitation attack. To tease out this effect, we run another experiment where we scale up the training data size to match the size used in the paraphrasing experiment; results in Table 8. Interestingly, we found that watermark detection performance goes up, showing that the success of paraphrasing in evading detection is not due to increased training data size.

B.7 Filtering with Different Cosine Similarity

We observed that the Enron dataset contains derogatory or ambiguous text to which GPT-3.5 responds with a general disclaimer or refusal to answer. We implemented a filtering process to ensure that only relevant content is used. If no valid paraphrases are found after filtering, we revert to the default response. We also conducted an ablation study (more in Figure 12) to determine the optimal cosine similarity threshold. We settle on 80% for filtering providing a good tradeoff between quality and attack performance. This process effectively filters out texts with low paraphrase similarity, such as derogatory content or extremely vague and short texts like "swill" or "free" in the SST2 dataset.

B.8 Quality of Paraphrases

Although the quality of paraphrases is not crucial for the attack, poor-quality paraphrases can result in utility loss. We did not conduct a human evaluation, as all the paraphrasers we use have already been evaluated for the same. Hence, we check the cosine similarity between original and paraphrased text embeddings. We can note from Figure 13 that most paraphrases are similar to the original text demonstrating good-quality paraphrases. Furthermore, with the implemented cosine similarity filter (as discussed in Appendix B.7) we will remove low-quality paraphrases corresponding to left-side entries of the distribution plots.

C WET Analyses

We perform detailed ablation studies for *WET*.

C.1 Impact of Gaussian Noise

We want to evaluate the effect of adding Gaussian noise to embeddings on watermark verification and downstream utility. Following (Morris et al., 2023; Chen et al., 2024), we consider different noise levels (λ) and add noise as follows:

$$\phi_{noisy}(x) = \text{Norm}(\phi(x) + \lambda \cdot \epsilon), \epsilon \sim \mathcal{N}(0, 1).$$

How much perturbation can be handled by *WET*? From Table 9, we can see that from $\phi = 0.05$, we start seeing significant utility loss; however, we have a perfect AUC for this case for all the datasets. It demonstrates that *WET* has more capacity to handle perturbations and is more robust.

Dataset	ϕ	Utility		Verifiability	
		ACC ↑	F1 ↑	Δ_{cos} ↑	AUC ↑
Enron	0.01	93.45	93.45	62.29	100.00
	0.05	84.00	84.00	17.39	100.00
	0.10	73.60	73.59	8.84	99.00
	0.50	52.30	51.30	1.77	70.62
	1.00	50.95	49.52	0.88	61.71
SST2	0.01	91.40	91.39	64.78	100.00
	0.05	84.29	84.26	18.09	100.00
	0.10	73.74	73.65	9.17	99.55
	0.50	53.67	49.87	1.79	69.03
	1.00	51.72	45.76	0.86	59.32
MIND	0.01	70.37	44.18	63.61	100.00
	0.05	63.20	33.34	17.76	100.00
	0.10	49.83	15.87	9.03	99.26
	0.50	31.60	4.85	1.82	69.78
	1.00	29.34	4.85	0.92	60.60
AG News	0.01	92.28	92.25	64.09	100.00
	0.05	83.92	83.84	17.73	100.00
	0.10	65.58	65.52	9.00	99.33
	0.50	30.00	29.88	1.79	69.59
	1.00	25.16	25.00	0.89	59.92

Table 9: Impact of different Gaussian noise (ϕ) in *WET* for all the datasets.

C.2 Impact of Size of Verification Dataset

In this, we investigate the number of samples (V) we need in the verification dataset. From Table 10, we can observe that *WET*'s verification technique is not dependent on the number of verification samples, even just a single verification sample might suffice. This is another advantage of our technique, do note in our experiment we use $v = 250$ unless specified otherwise.

C.3 Different Transformation Matrices Construction

We investigate different alterations to our construction of the transformation matrix $\mathbf{T}$, discussed in Section 3.3.

New Wts. Circulant Matrix. In this, we construct new weights for each row in the circulant matrix. However, with this, we also lose the full-rank and well-conditioned properties.

Random Matrix. This is a pure random generation process where we randomly pick k non-zero positions and assign random values to them, for each row.

Dataset	V	Verifiability	
		$\Delta_{cos}\uparrow$	AUC $\uparrow$
Enron	1	90.00	100.00
	5	88.26	100.00
	20	89.90	100.00
	100	89.14	100.00
	500	89.67	100.00
	1000	89.34	100.00
SST2	1	90.29	100.00
	5	93.75	100.00
	20	93.67	100.00
	100	93.77	100.00
	436	93.89	100.00
MIND	1	92.28	100.00
	5	89.90	100.00
	20	90.74	100.00
	100	90.54	100.00
	500	90.97	100.00
	1000	90.87	100.00
AG News	1	96.13	100.00
	5	93.62	100.00
	20	92.63	100.00
	100	91.77	100.00
	500	92.10	100.00
	1000	92.00	100.00

Table 10: Impact of different size of verification dataset sizes (V) in *WET* verifiability for all the datasets. Note: SST2 has only 872 test samples (see Table 1).

Eq. Wts. Circulant. We set the $1/k$ as the value to non-zero positions in the row.

Seq. Pos. Circulant. We pick the first k dimensions in the row as the non-zero positions.

Seq. Pos. and Eq. Wts. Circulant. This is the combination of the previous two matrix constructions.

Discussion. We present the performance of such matrices in Table 11. We see that equal weights and sequential position-based matrices have strong verifiability. However, such matrix constructions are not stealthy with equal weights or sequential positions. At the same time, the matrix combining the above methods is much worse in terms of verifiability. The other two constructions of new row weights every time in circulant matrix and pure ran-

Dataset	Type	Utility		Verifiability	
		ACC $\uparrow$	F1 $\uparrow$	$\Delta_{cos}\uparrow$	AUC $\uparrow$
Enron	Circulant	94.75	94.75	89.22	100.00
	New Wts. Circulant	94.60	94.60	21.60	99.96
	Random	94.30	94.30	22.95	99.96
	Eq. Wts. Circulant	94.40	94.40	92.81	100.00
	Seq. Pos. Circulant	93.40	93.40	69.91	100.00
	Seq. Pos. and Eq. Wts. Circulant	92.45	92.45	-0.23	47.69
SST2	Circulant	93.35	93.34	93.70	100.00
	New Wts. Circulant	93.00	93.00	23.60	99.99
	Random	92.55	92.54	25.02	100.00
	Eq. Wts. Circulant	92.78	92.77	96.13	100.00
	Seq. Pos. Circulant	91.97	91.97	74.31	100.00
	Seq. Pos. and Eq. Wts. Circulant	90.60	90.59	0.72	53.88
MIND	Circulant	77.21	51.36	91.12	100.00
	New Wts. Circulant	76.97	50.83	23.56	100.00
	Random	77.00	50.94	24.82	100.00
	Eq. Wts. Circulant	77.04	51.03	95.34	100.00
	Seq. Pos. Circulant	76.61	50.06	71.93	100.00
	Seq. Pos. and Eq. Wts. Circulant	75.21	47.22	0.02	50.28
AG News	Circulant	93.03	93.02	92.05	100.00
	New Wts. Circulant	93.20	93.19	26.60	100.00
	Random	92.95	92.94	27.55	100.00
	Eq. Wts. Circulant	93.07	93.06	96.47	100.00
	Seq. Pos. Circulant	92.41	92.40	73.79	100.00
	Seq. Pos. and Eq. Wts. Circulant	91.89	91.88	-0.27	50.37

Table 11: *WET* performance using different variation of transformation matrix $\mathbf{T}$ as defined in the Section C.3.

dom matrix construction have low Δ_{cos} metric even though it has perfect AUC. The reason is such matrices are not full-rank and are not well-conditions leading to poorer reverse transformation.

C.4 Impact of Attack Model Size

Dataset	Size	Utility		Verifiability	
		ACC $\uparrow$	F1 $\uparrow$	$\Delta_{cos}\uparrow$	AUC $\uparrow$
Enron	Small	94.55	94.55	88.74	100.00
SST2		93.23	93.23	93.12	100.00
MIND		77.15	51.00	90.25	100.00
AG News		92.92	92.91	91.30	100.00
Enron	Base	94.75	94.75	89.22	100.00
SST2		93.35	93.34	93.70	100.00
MIND		77.21	51.36	91.12	100.00
AG News		93.03	93.02	92.05	100.00
Enron	Large	94.40	94.40	88.32	100.00
SST2		93.00	93.00	93.60	100.00
MIND		76.95	50.77	90.94	100.00
AG News		93.29	93.28	92.43	100.00

Table 12: The impact of extracted model size on *WET* performance.

We assess whether our defence's performance varies with the attacker model's size. We conducted experiments for *WET* using small, base, and large variants of the BERT model (Devlin et al., 2019). The results, summarised in the Table 12, demonstrate that the defence works effectively with similar utility and verifiability.
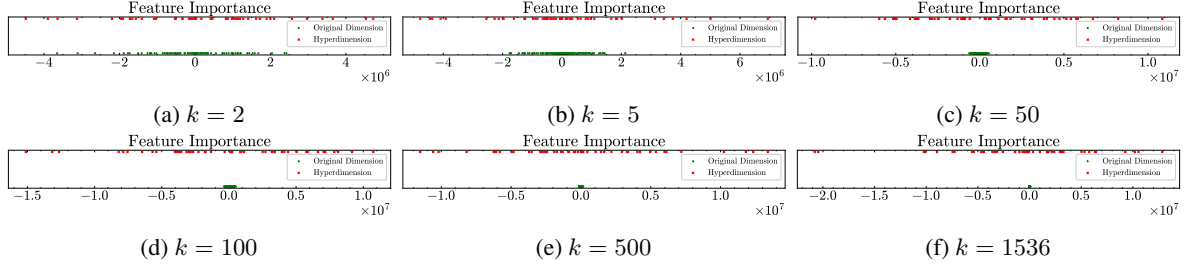
Figure 14: Visualisation plots for feature importance of watermarked embedding dimensions in SST2.
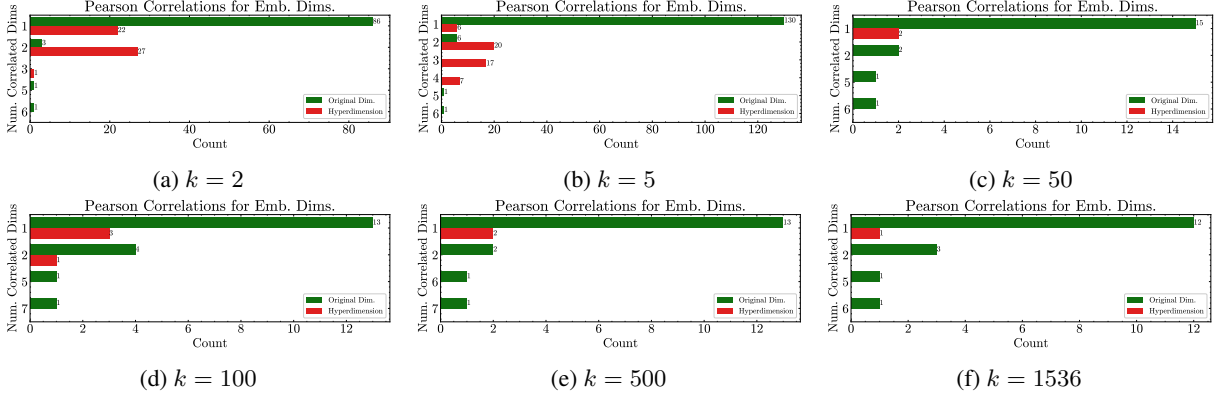

Figure 15: Visualisation plots for feature correlations of watermarked embedding dimensions in SST2.

C.5 Hyperdimension Obfuscation

In this section, we focus on the case where we add extra dimensions (*a.k.a* hyperdimensions; $w = 50$) among the original embeddings. The positions of these hyperdimensions are randomly decided, and the value is a linear transformation of some k existing original dimensions (similar to Algorithm 1 but used as additional dimensions). For verification, we use the same ideas as in *WET*, with the only difference being that we work only on these obfuscated hyperdimensions. The utility and verifiability were comparable to *WET*. To evaluate the stealthiness of these hyperdimensions, we investigated feature correlation and feature importance techniques between hyperdimension and original dimensions. Properly mixed feature importance weights illustrate that hyperdimensions are indistinguishable. Similarly, uncorrelated hyperdimensions are appreciated, or else they are redundant. Note that these stealthiness techniques are not applicable for *WET* as we discard the original embedding dimensions.

C.5.1 Feature Importance

We train a linear regression (since we are working with linear transformations) with all the watermarked embeddings (original and hyperdimensions) for the downstream task. We use the weights of the linear regression as the feature importance

weights. In Figure 14, we represent these plots for different values of k. From this, we can conclude that we need $k < 5$, as for higher values, hyperdimensions are discernible from the original embedding dimensions. For higher values of k, we have hyperdimensions that have more feature importance, which is logical considering linear combinations used in hyperdimension will represent the whole embedding with a higher value of k.

C.5.2 Feature Correlations

In this analysis, we use Pearson's coefficient (Sedgwick, 2012) with a threshold of 0.4. The plots in Figure 15 indicate that a k value between 5 and 50 is required. However, this range conflicts with the values necessary ($k < 5$) to bypass feature importance evaluation. Consequently, these plots (Figures 14 and 15) lead us to conclude that hyperdimension obfuscation will not work as they are easily detectable.

Dataset	k	Utility		Verifiability	
		ACC ↑	F1 ↑	Δ_{cos} ↑	AUC ↑
	1	94.75	94.75	89.13	100.00
	2	94.75	94.75	82.92	100.00
	5	94.80	94.80	87.64	100.00
	25	94.75	94.75	89.22	100.00
Enron	50	94.40	94.40	90.86	100.00
	100	94.35	94.35	82.84	100.00
	500	92.85	92.85	81.70	100.00
	1000	92.15	92.15	82.24	100.00
	1536	91.70	91.70	85.50	100.00
	1	93.23	93.23	91.65	100.00
	2	92.66	92.66	87.75	100.00
	5	93.00	93.00	91.59	100.00
	25	93.35	93.34	93.70	100.00
SST2	50	93.35	93.34	94.45	100.00
	100	92.89	92.89	87.39	100.00
	500	92.32	92.31	85.81	100.00
	1000	92.55	92.54	86.58	100.00
	1536	91.63	91.62	89.38	100.00
	1	77.25	51.40	91.62	100.00
	2	77.10	51.19	85.70	100.00
	5	77.16	51.05	89.19	100.00
	25	77.21	51.36	91.12	100.00
MIND	50	76.95	50.71	92.41	100.00
	100	76.88	50.72	85.06	100.00
	500	76.61	49.85	85.42	100.00
	1000	75.67	48.24	84.74	100.00
	1536	74.28	42.96	85.46	100.00
	1	93.45	93.44	92.85	100.00
	2	93.46	93.46	86.68	100.00
	5	93.22	93.22	90.59	100.00
	25	93.03	93.02	92.05	100.00
AG News	50	93.22	93.22	93.30	100.00
	100	93.00	93.00	86.90	100.00
	500	92.62	92.61	86.28	100.00
	1000	92.18	92.17	86.22	100.00
	1536	91.59	91.58	86.66	100.00

Table 13: Different k for $h = 1536$ results. Expanding on Section 4.5, we provide detailed results here for completeness.

Dataset	w	Utility		Verifiability	
		ACC ↑	F1 ↑	Δ_{cos} ↑	AUC ↑
	50	88.05	88.04	10.58	100.00
	500	93.10	93.10	53.34	100.00
Enron	1000	94.15	94.15	74.95	100.00
	1536	94.75	94.75	89.22	100.00
	3000	94.75	94.75	90.49	100.00
	50	84.40	84.34	10.20	100.00
	500	93.23	93.23	54.07	100.00
SST2	1000	93.58	93.57	77.07	100.00
	1536	93.35	93.34	93.70	100.00
	3000	92.55	92.54	94.47	100.00
	50	67.53	37.08	11.23	100.00
	500	76.30	49.83	52.96	100.00
MIND	1000	76.78	50.55	75.74	100.00
	1536	77.21	51.36	91.12	100.00
	3000	76.96	50.72	93.85	100.00
	50	85.07	85.04	12.44	100.00
	500	92.46	92.45	53.55	100.00
AG News	1000	92.92	92.91	76.72	100.00
	1536	93.03	93.02	92.05	100.00
	3000	93.05	93.05	94.19	100.00

Table 14: Different w for $k = 25$ results. Expanding on Section 4.5, we provide detailed results here for completeness.