Semantic-Oriented Robust Text Watermark for Large Language Models

Anonymous ACL submission

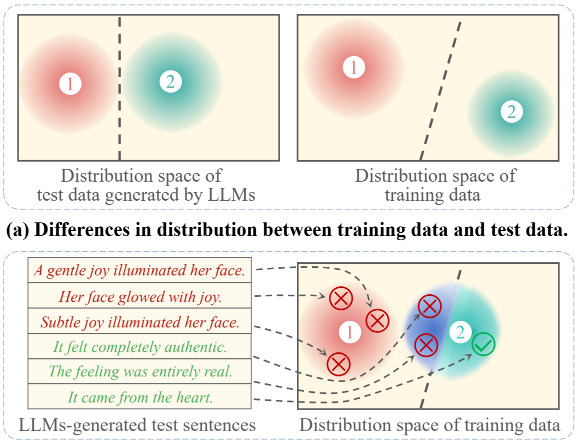
Abstract

Text watermark focuses on injecting identifiable information into the generated content, which has become increasingly important with the rapid development of Large Language Models (LLMs). Existing watermarking works either divide the vocabulary of LLMs into "green" and "red" tokens for the watermark generation (i.e., token-level watermark), or use the distance of generated sentence embeddings to distinguish the "green" and "red" partitions (i.e., sentence-level watermark). Despite the achieved progress, existing methods are still vulnerable when dealing with attacking or Out-Of-Distribution (OOD) generalization. To this end, we focus on sentence-level watermark and propose a novel *Semantic-oriented Robust Text Watermark for LLMs* (*SoTW*). Specifically, we first employ a pre-trained embedding model to obtain representations of generated sentences. Then, different from existing sentence-level works, we design a novel Semantic Quantization AutoEncoder (*SQAE*) to generate discrete representations for the partitions. Moreover, a semantic loss and a consistency loss are developed to ensure the generalization and robustness of generated watermarks. Furthermore, we develop an easy-to-use detection method for our proposed *SoTW*. Extensive experiments with two LLMs over two publicly available datasets demonstrated the robustness of *SoTW* in different attack methods and OOD settings. As a bypass, we release the code to facilitate the community[1].

1 Introduction

Large Language Models (LLMs) have demonstrated impressive generation capabilities and have been widely used in various applications, such as ChatBot (OpenAI, 2023), Copilot (Microsoft, 2023), Claude (Anthropic, 2023), etc. With the deep collaboration of LLMs in content generation,

[1] http://anonymous.4open.science/r/SoTW/



(a) Differences in distribution between training data and test data.

(b) Test sentences under the training data distribution space.

Figure 1: Illustration of sentence distributions divided by existing sentence-level watermark algorithms, where similar sentence embeddings are represented by the same color, and the blue partition indicates that the test data is misclassified based on the distribution space partitioned by training data.

potential risks (e.g., misleading information, copyright issues) also become essential when using generated content (Rillig et al., 2023). For example, lawyers use LLMs to fabricate a legal brief filled with fictitious case references[2], and the New York Times sues OpenAI for generating content that infringes on its copyrights[3].

Among all of them, text watermark technology can be used in both information identification and copyright tracing, which has become a hot topic in LLMs application (Kirchenbauer et al., 2023; Chen et al., 2024b). Specifically, text watermark for LLMs aims to embed implicit identifiable information into generated content, in which designing "red" and "green" groups is the common paradigm. For LLMs, this technology can be classified into token-level (e.g., KGW (Kirchenbauer et al., 2023), SIR (Liu et al., 2024)) and sentence-

[2] https://www.nytimes.com/2023/06/22/nyregion/lawyers-chatgpt-schwartz-loduca.html

[3] https://nytco-assets.nytimes.com/2023/12/NYT_Complaint_Dec2023.pdf

Algorithm	Model	Similar	Dissimilar	Avg.
SEMSTAMP	LSH	42.30%	86.25%	64.28%
k-SEMSTAMP	k-means	73.76%	53.32%	63.54%

Table 1: Results ($\underline{Accuracy}$ ↑) of partitioning sentences by constructing sentence partitions with size 64 using LSH and k-means employed in SEMSTAMP and k-SEMSTAMP, respectively. *Similar*: sentences with a similarity score above 4.0 fall into the same partition; *Dissimilar*: sentences with a similarity score below 0.7 fall into different partitions. Note that the training data is MultiNLI (Williams et al., 2018) and the test data is STS (Cer et al., 2017).

level (e.g., SEMSTAMP (Hou et al., 2024a), k-SEMSTAMP (Hou et al., 2024b)) watermark algorithms based on the granularity of the watermark. Token-level methods usually focus on how to divide the vocabulary into "red" and "green" tokens. E.g., KGW utilizes the hash value of a previous token as the random seed to divide the vocabulary of LLMs into "red" and "green" tokens, favoring "green" tokens by increasing their logits during sampling. Sentence-level methods focus on projecting the sentence representations into "red" and "green" sentence partitions. For example, k-SEMSTAMP uses k-means clustering (Lloyd, 1982) based on sentence embedding distances to determine cluster centers as sentence partition anchors, then divides these sentence partitions into "green" and "red" partitions using the sentence partition number of a previous sentence as the random seed. During sampling, sentences that fall into "green" partitions are preferentially selected.

Despite the progress, existing methods still suffer from the vulnerable watermarking capability and poor generalization. For token-level methods, sentence-level attacks (e.g., translation) can easily break pre-defined "red" and "green" groups, rendering the watermark. For sentence-level methods, existing methods usually construct sentence partitions based on the distances between the generated sentences and the partition anchors, suffering from weak Out-Of-Distribution (OOD) generalizations. Taking Figure 1 (a) as an example, existing methods usually employ cluster methods to decide the partition anchors (i.e., the red and green points in the figure). Then, they use the distance calculations to realize the partition. However, when dealing with OOD scenarios (i.e., Figure 1 (a)), existing methods will inevitably conduct incorrect partitions. Moreover, these incorrect partitions will cause a large number of generated sentences to be clustered into the same partition (i.e., 1 partition in Figure 1 (b)), thus significantly reducing the quality of watermarks. To support our opinion, we conduct experiments over advanced k-SEMSTAMP and report results in Table 1. From the results, we observe that k-SEMSTAMP tends to divide sentences with different meanings into the same partition, resulting in higher sentence concentration and lower distinguishability (i.e., lower value on *Dissimilar*). Therefore, one important question remains unresolved "**How to construct robust sentence partitions for sentence-level text watermarking**?"

To this end, in this paper, we propose a novel *Semantic-oriented Robust Text Watermark for LLMs* (*SoTW*) for robust sentence-level watermarking. Different from existing sentence-level methods, we innovatively propose to use a learnable discrete representation to directly represent different partitions. Specifically, we first leverage pre-trained embedding models (e.g., BGE-M3 (Chen et al., 2024a)) to generate sentence embeddings. Then, instead of finding the partition anchors and calculating the distance, we design a novel Semantic Quantization AutoEncoder (*SQAE*) to generate the discrete representations by taking the sentence embeddings as the input. Since the discrete representation will lose important semantic information, we develop a semantic loss to ensure that discrete representations can maintain as much information as possible. Moreover, considering that sentences with the same semantics can be expressed in multiple different ways, we design a consistency loss to improve the robustness of the partition boundaries. Along this line, LLMs generated sentences can be accurately assigned to corresponding partitions, thus improving the generalization and robustness of sentence-level watermarking. Finally, extensive experiments on two advanced LLMs against eight state-of-the-art algorithms demonstrate the superiority and effectiveness of *SoTW*. Compared with existing token-level and sentence-level watermark algorithms, *SoTW* improves watermark embedding success rate and resistance to translation attacks by 76% and 94%, respectively.

2 Related Work

In this section, we group the related work into two categories based on the granularity of watermark: *Token-level watermark algorithms* and *Sentence-level watermark algorithms*.

Token-level watermark algorithms aim to mark some tokens as "green" before generating

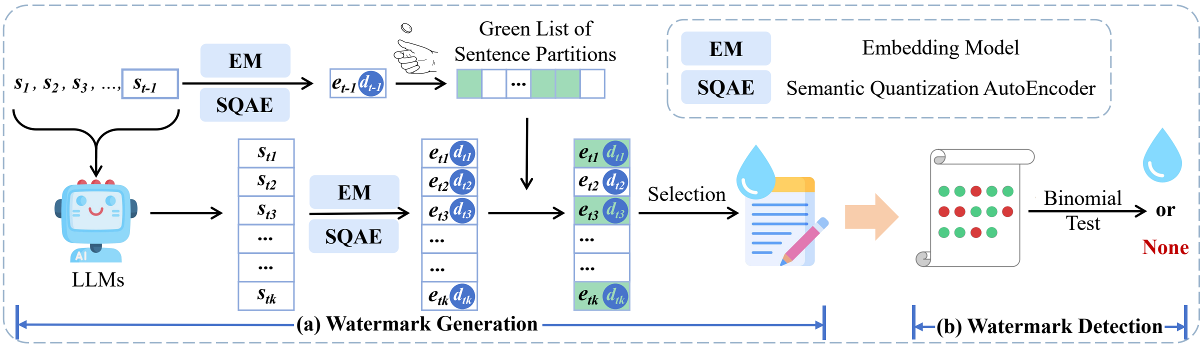


Figure 2: The overall process of watermark generation and detection.

a token, and then select these green-marked tokens during sampling. KGW (Kirchenbauer et al., 2023) divides the vocabulary of LLMs into "red" and "green" tokens based on the hash value of the previous token, and increases the probability of green-marked tokens being selected by increasing logits of these tokens. To improve the robustness of KGW, UNIGRAM (Zhao et al., 2024) fixes the vocabulary of LLMs globally to avoid the impact of text changes on "red" and "green" tokens. However, due to the lack of diversity in "red" and "green" tokens, they can be easily inferred, and non-watermarked texts may still contain a large number of green-marked tokens, leading to more false positives. Semantically Invariant Robust watermark algorithm (SIR) (Liu et al., 2024) divides the LLMs vocabulary according to text semantics, improving the robustness of the embedded watermark against attacks. To further resist translation attacks, X-SIR (He et al., 2024) marks tokens that mutually translate in the vocabulary with the same color. Moreover, Robust and Imperceptible Watermark algorithm (RIW) (Ren et al., 2024) leverages token prior probabilities to divide the vocabulary, improving detectability and maintaining watermark imperceptibility. Besides, to improve the quality of the generated text, Watermarking with Mutual Exclusion (WatME) (Chen et al., 2024b) clusters synonyms and divides them into "red" and "green" synonyms. However, paraphrase attacks can remove watermarks by replacing words or word order without changing the semantics. Therefore, paraphrase attacks may eliminate watermarks generated by token-level watermark algorithms.

Sentence-level watermark algorithms aim to mark some sentences as "green" before generating sentences, and then select these green-marked sentences during sampling to embed watermarks. These algorithms embed watermarks based on sentence semantics, ensuring that the embedded watermarks will not be eliminated when the sentence semantics are unchanged. SEMSTAMP (Hou et al., 2024a) uses locality-sensitive hashing (LSH) (Indyk and Motwani, 1998; Charikar, 2002) randomly partitioning the hash space to construct sentence partitions and dividing them into "green" and "red" partitions. Subsequently, sentences that fall into green-marked partitions are selected during sampling. However, randomly constructing sentence partitions does not guarantee that semantically similar sentences fall into the same partition. Further, k-SEMSTAMP (Hou et al., 2024b) determines sentence partition centers using cluster centers derived from k-means clustering (Lloyd, 1982), based on the distances between sentence embeddings.

Our Distinction. Different from existing methods, we propose to directly learn the discrete representations for sentence partitions, avoiding the sensitivity of partition anchor searching and distance calculation methods in sentence-level methods. Moreover, we leverage semantic loss and consistency loss to enhance the generalization and robustness of learned partition boundaries. Along this line, *SoTW* can generate robust text watermark against various OOD and attack scenarios.

3 *Semantic-oriented Robust Text Watermark for LLMs*

In this section, we describe *SoTW* in detail. First, we introduce the overall pipeline of watermark generation. Then, we introduce the technical details of *SQAE* and the watermark detection process.

3.1 Overall Process

The overall process of watermark generation is illustrated in Figure 2 (a). Specifically, given the previous sentences $\boldsymbol{S}_{:t-1} = [s_1, s_2, ..., s_{t-1}]$, we need to select the t^{th} sentence from a newly generated sentence set $\boldsymbol{S}_t = [s_{t1}, s_{t2}, ..., s_{tk}]$, so that the watermark can be properly inserted. Thus, we

Algorithm 1 Pseudocode of Watermark Generation

Input: $\mathbb{M}$: LLMs;
  s_1: a prompt;
  T: the number of generated sentences;
  $\mathbb{E}$: a pre-trained embedding model;
  $\mathbb{Q}$: a trained *SQAE*;
Output: $\boldsymbol{S}_{:T}$: watermark text $[s_1, s_2, ..., s_T]$;

for $t = 2, 3, ..., T$ **do**
  $\boldsymbol{e}_{t-1} = \mathbb{E}(s_{t-1})$; //sentence embedding
  $\boldsymbol{d}_{t-1} = \mathbb{Q}(\boldsymbol{e}_{t-1})$; //discrete representation
  Use $\boldsymbol{d}_{t-1}$ as the seed to divide sentence partitions into "green" and "red" partitions;
  repeat
    $s_t = \mathbb{M}(\boldsymbol{S}_{:t-1})$; //next sentence
    $\boldsymbol{e}_t = \mathbb{E}(s_t)$; //sentence embedding
    $\boldsymbol{d}_t = \mathbb{Q}(\boldsymbol{e}_t)$; //discrete representation
  until $\boldsymbol{d}_t$ in "green" partitions;
end for

first leverage a pre-trained embedding model to generate embeddings $\boldsymbol{E}_t = [\boldsymbol{e}_{t1}, \boldsymbol{e}_{t2}, ..., \boldsymbol{e}_{tk}]$ for $\boldsymbol{S}_t$. Then, we design a novel *SQAE* to generate discrete representations $\boldsymbol{D}_t = [\boldsymbol{d}_{t1}, \boldsymbol{d}_{t2}, ..., \boldsymbol{d}_{tk}]$. Next, we utilize the discrete representation $\boldsymbol{d}_{t-1}$ of the previous sentence s_{t-1} as the random seed to divide sentence partitions into "green" and "red" partitions, and select a sentence s_t that falls into "green" partition. By iterating the process, we can realize the watermark generation. We also provide the pseudocode in Algorithm 1 and the detailed notation explanation in Table 5 in Appendix A.

3.2 *SQAE*

Structure of *SQAE*. As mentioned in Section 3.1, we first leverage pre-trained embedding models (e.g., BGE-M3) to obtain sentence embeddings $\boldsymbol{E}_t$. Then we need to construct the "green" and "red" partitions for watermarking. In general, we can define some partition anchors and use distance calculation to divide $\boldsymbol{S}_t$, which is also the main strategy of existing methods. However, this strategy suffers from weak OOD generalizations. Thus, how to construct robust partitions with sentence embeddings remains challenging. In response, we propose to directly learn the partition representation, which should be able to project the sentence embeddings to different partitions directly. Therefore, we can alleviate the negative impact of partition anchor selections and distance calculations.

Specifically, as illustrated in Figure 3, we design a novel *SQAE* to achieve this goal, which consists

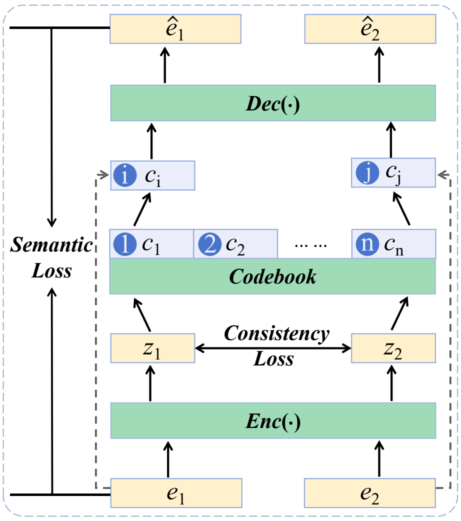


Figure 3: The architecture of our proposed *SQAE*.

of three main components: *Encoder*, *Codebook*, and *Decoder*. For simplicity, we omit the subscript t for better description. Note that the following process focuses on the t^{th} green sentence selection (watermarking process).

For *Encoder*, we intend the sentence embeddings to be separated from the discrete representations. Therefore, we take sentence embeddings $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ as the input and use encoder $Enc(\cdot)$ to generate latent representations, formulating as $\boldsymbol{z}_1 = Enc(\boldsymbol{e}_1)$, $\boldsymbol{z}_2 = Enc(\boldsymbol{e}_2)$.

For *Codebook*, we aim to use the obtained presentations to directly generate discrete representations for the partitions. Thus, following previous work (Van Den Oord et al., 2017), we first design a codebook to represent the partition space, formulating as $\boldsymbol{R} \in \mathcal{R}^{n \times d}$. Then, we use the following operation to obtain the discrete representations (i.e., i for $\boldsymbol{z}_1$ and j for $\boldsymbol{z}_2$):

$$\begin{aligned} i &= \arg\min_k \|\boldsymbol{z}_1 - \boldsymbol{c}_k\|^2, \\ j &= \arg\min_k \|\boldsymbol{z}_2 - \boldsymbol{c}_k\|^2, \end{aligned} \tag{1}$$

where $\boldsymbol{c}_k$ is the k^{th} partition representation. $\|\cdot\|^2$ is the L_2-norm. By using Eq.(1), we can select the most suitable partition for each input sentence.

For *Decoder*, since there is no supervised signal for the learning process of discrete partition representations, we intend to incorporate an autoencoder to ensure the learning quality. Thus, we use decoder $Dec(\cdot)$ to reconstruct the original embeddings based on latent representations, formulating as: $\hat{\boldsymbol{e}_1} = Dec(\boldsymbol{c}_i)$, $\hat{\boldsymbol{e}_2} = Dec(\boldsymbol{c}_j)$.

Learning Strategy of *SQAE*. In the above module, we intend to project the generated sentences into different partitions for watermarking.

However, one important issue remains unresolved: "*There is no supervised signal for the learning process*". Moreover, how to ensure the robustness of learned discrete representations is still unclear. To tackle the above problems, we design two optimization targets for our proposed *SQAE*: *semantic loss* and *consistency loss*.

1) Semantic Loss. Since *SQAE* projects the sentence embeddings into discrete representations, it will inevitably lose useful information, hurting the model performance. Therefore, we design a novel semantic loss to alleviate this problem. Specifically, we leverage the original embedding as the guidance to reconstruct the original embedding via the latent representation, ensuring that the latent representation learns more semantic information about the original embedding. Thus, we utilize the original embedding as the prediction target and formulate the optimization objective as follows:

$$\begin{aligned}\mathcal{L}_{rec} &= ||\boldsymbol{e}_1 - \hat{\boldsymbol{e}_1}||_2^2 \\ &= ||\boldsymbol{e}_1 - Dec(\boldsymbol{z}_1 + sg(\boldsymbol{c}_i - \boldsymbol{z}_1))||_2^2,\end{aligned} \tag{2}$$

where $sg(\cdot)$ denotes the stop-gradient operator that has zero partial derivatives. Meanwhile, since the minimum operation in Eq.(1) has no gradient, the above optimization objective cannot be used to update the codebook. In response, we utilize the latent embedding as the guidance to enable the embedding in the codebook learning the semantic information of the latent representation, and formulate the optimization objective as follows:

$$\begin{aligned}\mathcal{L}_{co} &= ||\boldsymbol{z}_1 - \boldsymbol{c}_i||_2^2 \\ &= ||\boldsymbol{c}_i - sg(\boldsymbol{z}_1)]||_2^2 + \delta||\boldsymbol{z}_1 - sg(\boldsymbol{c}_i)]||_2^2,\end{aligned} \tag{3}$$

where the former optimizes the embedding in the codebook, the latter optimizes the latent representation, and δ is generally set to a smaller value for reducing the update of the latent representation. Thus, the total *semantic loss* is

$$\mathcal{L}_s = \mathcal{L}_{rec} + \mathcal{L}_{co}. \tag{4}$$

2) Consistency Loss. Besides, LLMs can generate multiple different sentences to express the same semantics. Therefore, *SQAE* should be able to project sentences with the same semantics into similar discrete representations, while projecting sentences with different semantics into different partitions as far as possible. To this end, we maximize the divergence in their latent representations $\{\boldsymbol{z}_1, \boldsymbol{z}_2\}$, which can be formulated as follows:

$$\begin{aligned}\mathcal{L}_c = \frac{1}{2N}\sum_{i=1}^{N}[&\mathbb{1}_{sim(\boldsymbol{e}_1,\boldsymbol{e}_2)>\alpha} sim(\boldsymbol{z}_1, \boldsymbol{z}_2)^2 + \\ &\mathbb{1}_{sim(\boldsymbol{e}_1,\boldsymbol{e}_2)\leq\alpha} \max(sim(\boldsymbol{z}_1, \boldsymbol{z}_2) - \alpha, 0)^2],\end{aligned} \tag{5}$$

where $sim(\cdot,\cdot)$ denotes the similarity function (e.g., cosine similarity), α is a threshold, and $\mathbb{1}$ is the indicator function.

Finally, we use a weighted summarization of Eq.(4) and Eq.(5) to construct the optimization target for our proposed *SoTW*, where the former ensures the learned quality of discrete representations and the latter improve the robustness of the watermarking. The overall optimization can be formulated with a hyper-parameter λ as follows:

$$\mathcal{L} = \lambda\mathcal{L}_s + (1-\lambda)\mathcal{L}_c. \tag{6}$$

3.3 Watermark Detection

As illustrated in Figure 2 (b), after generating watermarks, it is also essential to develop a convenient detection method. For our proposed *SoTW*, a third party just needs to use the embedding model and trained *SQAE* to obtain the partition of each sentence and count the number of sentences that fall into "green" partitions. Then they can detect watermarks by testing the following null hypothesis:

H0: The rules for dividing sentence partitions are unknown when generating text.

Since half of the sentence partitions are randomly selected as "green" partitions, approximately half of sentences in the non-watermarked text fall into "green" partitions, while all sentences in the watermarked text fall into "green" partitions. Therefore, we use the binomial test (Howell, 1992) to evaluate the null hypothesis, as follows:

$$p\text{-value} = \sum_{i=N_G}^{N_T}\binom{N_T}{i}\left(\frac{1}{2}\right)^{N_T}, \tag{7}$$

where N_G refers to the number of green-marked sentences, and N_T is the total number of sentences. Note that our detection method requires no LLMs and can work efficiently.

4 Experiment

In this section, we first introduce the experimental setup. Then, we provide a detailed analysis of the experimental results of *SoTW* and its rivals. Moreover, we conduct detailed experiments to verify the effectiveness of each component in *SQAE*.

Setting	Algorithm	C4 dataset						LFQA dataset						Avg.↑
		LLaMA2-7B-Chat			Baichuan2-7B-Chat			LLaMA2-7B-Chat			Baichuan2-7B-Chat			
		F1↑ 1%FPR	F1↑ 5%FPR	AUC↑	F1↑ 1%FPR	F1↑ 5%FPR	AUC↑	F1↑ 1%FPR	F1↑ 5%FPR	AUC↑	F1↑ 1%FPR	F1↑ 5%FPR	AUC↑	
No Attack	KGW$_{\text{ICML'23}}$	97.14	96.96	99.66	99.30	97.74	99.94	93.68	95.41	99.14	**98.70**	**98.13**	**99.90**	**97.98**
	UNIGRAM$_{\text{ICLR'24}}$	46.97	68.11	91.07	61.26	85.56	94.43	34.43	60.16	87.07	40.44	65.21	86.01	68.39
	SIR$_{\text{ICLR'24}}$	89.13	92.20	98.37	93.91	95.52	99.04	81.59	91.62	96.98	96.52	96.44	99.36	94.22
	X-SIR$_{\text{ACL'24}}$	67.45	83.72	96.30	96.20	96.35	99.46	77.87	88.63	96.60	96.62	97.15	99.73	91.34
	RIW$_{\text{ACL'24 (Findings)}}$	14.34	59.25	85.54	24.35	73.34	90.33	22.81	46.49	79.79	29.39	59.60	86.30	55.96
	WatME$_{\text{ACL'24}}$	91.51	94.25	98.94	**99.40**	**98.04**	**99.97**	92.32	95.21	**99.25**	98.08	97.23	99.73	96.99
	SEMSTAMP$_{\text{NAACL'24}}$	0.40	6.72	48.07	8.71	12.66	60.59	0.40	1.89	49.14	5.02	16.11	58.77	22.37
	k-SEMSTAMP$_{\text{ACL'24 (Findings)}}$	2.73	28.15	67.51	43.17	86.73	95.28	3.12	9.16	54.19	39.43	78.84	91.68	50.00
	SoTW (Ours)	**97.77**	**97.16**	**99.81**	98.70	97.36	99.84	**94.12**	**95.42**	98.52	94.89	96.25	99.32	97.43
Rewrite	KGW$_{\text{ICML'23}}$	34.75	60.00	82.98	47.20	62.20	84.68	25.00	45.88	77.00	26.76	49.79	79.86	56.34
	UNIGRAM$_{\text{ICLR'24}}$	14.00	37.71	78.49	6.51	22.97	67.74	9.79	32.54	77.44	3.88	11.15	54.42	34.72
	SIR$_{\text{ICLR'24}}$	39.17	54.77	84.56	48.58	70.37	88.90	30.54	**64.95**	**84.90**	39.17	61.84	88.54	63.02
	X-SIR$_{\text{ACL'24}}$	20.28	40.43	83.22	**69.00**	**83.07**	**94.61**	**36.30**	63.10	86.32	**58.94**	**75.15**	**92.89**	**66.94**
	RIW$_{\text{ACL'24 (Findings)}}$	0.79	13.83	65.10	1.19	20.27	71.80	2.35	9.44	51.46	-	3.38	54.20	24.48
	WatME$_{\text{ACL'24}}$	19.96	33.12	73.17	49.70	68.09	88.95	24.96	47.61	78.05	31.39	47.61	78.08	53.39
	SEMSTAMP$_{\text{NAACL'24}}$	1.96	3.04	48.99	5.39	10.26	55.80	0.40	3.74	51.57	1.57	8.04	54.48	20.44
	k-SEMSTAMP$_{\text{ACL'24 (Findings)}}$	0.79	20.82	70.67	12.96	51.49	83.51	3.50	7.75	63.44	15.69	52.62	77.12	38.36
	SoTW (Ours)	**40.94**	**63.28**	**85.48**	52.98	70.69	90.20	34.21	58.68	83.45	36.36	64.52	86.15	63.91
Translate Chinese	KGW$_{\text{ICML'23}}$	3.12	10.47	47.87	6.90	18.97	56.99	1.18	9.11	48.07	5.02	13.52	57.07	23.19
	UNIGRAM$_{\text{ICLR'24}}$	3.89	8.39	49.33	4.26	20.24	64.00	3.88	10.49	50.34	3.88	13.17	55.41	23.94
	SIR$_{\text{ICLR'24}}$	0.40	0.78	7.51	10.51	27.59	66.10	0.40	3.01	8.65	12.27	24.41	68.83	19.21
	X-SIR$_{\text{ACL'24}}$	2.73	8.39	70.64	3.88	10.47	34.07	9.07	36.19	**80.81**	5.41	10.85	37.67	25.85
	RIW$_{\text{ACL'24 (Findings)}}$	-	-	-	-	-	-	-	-	-	-	-	-	0
	WatME$_{\text{ACL'24}}$	1.57	2.26	21.95	-	0.38	16.51	1.58	4.47	31.25	2.75	3.42	16.91	8.59
	SEMSTAMP$_{\text{NAACL'24}}$	2.73	4.90	51.01	5.02	8.50	53.55	0.79	13.50	54.74	3.12	12.83	54.16	22.07
	k-SEMSTAMP$_{\text{ACL'24 (Findings)}}$	0.79	17.71	69.70	12.27	42.58	77.84	3.12	8.10	63.52	21.55	**58.81**	**81.27**	38.11
	SoTW (Ours)	**19.32**	**41.39**	**73.02**	**22.50**	**43.52**	**78.63**	**13.33**	**37.96**	71.68	**21.91**	48.92	76.71	**45.74**

Table 2: Model performance against various attacks. **Boldface** and underline denote the best and second best results.

4.1 Experiment Setup

Dataset and Prompt: We utilize two datasets: C4 (Raffel et al., 2020) and LFQA (Krishna et al., 2024) to evaluate watermark algorithms. For C4, we take the first sentence as a prompt and generate the next 200 tokens. For LFQA, we use questions as prompts and generate 200 token responses. For model training, we utilize the MultiNLI dataset (Williams et al., 2018) (different from C4 and LFQA) to generate embeddings.

Baseline and Language Model: We select six token-level baselines (i.e., KGW (Kirchenbauer et al., 2023), UNIGRAM (Zhao et al., 2024), SIR (Liu et al., 2024), X-SIR (He et al., 2024), RIW (Ren et al., 2024), and WatME (Chen et al., 2024b)) and two sentence-level baselines (i.e., SEMSTAMP (Hou et al., 2024a) and k-SEMSTAMP (Hou et al., 2024b)). For a fair comparison, we use BGE-M3 (Chen et al., 2024a) with cross-lingual capabilities as the embedding model. We select LLaMA2-7B-Chat (Touvron et al., 2023) and Baichuan2-7B-Chat (Yang et al., 2023) to generate sentences for watermarking.

Evaluation: Similar to (Liu et al., 2024; Chen et al., 2024b), to avoid the impact of detection thresholds, we set False Positive Rate (FPR) at 1% and 5%, and adjusted the thresholds of detector accordingly to calculate the F1 score. We also calculate the Area Under the Curve (AUC) to evaluate performance. Furthermore, we evaluate the quality of the generated watermarked text by calculating its perplexity using the superior LLaMA2-13B (Touvron et al., 2023) model.

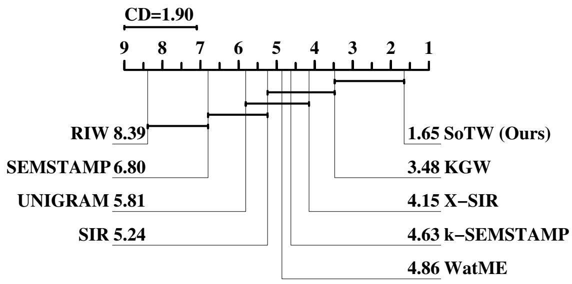


Figure 4: The crucial difference diagram of the Nemenyi test for our proposed *SoTW* and its rivals.

Hyper-parameters: For *SQAE* training, we use Adam optimizer ($lr = 1 * 10^{-5}$), the batch size is 64, the latent representation size (i.e., dimension of z_1 in Eq.(1)) is $1,000$, the size of the codebook n is 64, the δ in Eq.(3) is 0.25, the α in Eq.(5) is 0.7, and the λ in Eq.(6) is 0.5. Moreover, all experiments are conducted on NVIDIA A100 GPU.

4.2 Watermark Robustness

Table 2 presents the watermark detection results, including scenarios without attacks, as well as rewriting and translation to Chinese using *GPT4o-mini*. We also illustrate results of watermark text translated into French and Japanese in the Appendix B. For *Rewrite* and *Translate* attacks, we use prompts

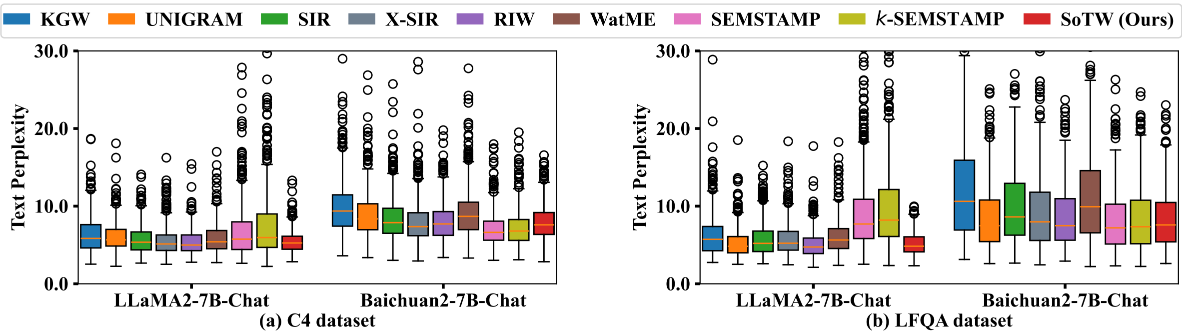


Figure 5: Text quality generated by LLMs with different watermark algorithms.

"*Rewrite the following paragraph*" and "*Translate the following English into Chinese*", respectively.

From these results, we can draw the following conclusions. Firstly, *SoTW* utilizes *SQAE* to replace cluster method for constructing sentence partitions, effectively improving the success rate of embedded watermarks (94% improvement in no attack setting), proving the effectiveness of *SQAE*. Moreover, when facing sentence-level attacks, especially the more destructive translation attacks, *SoTW* has significant advantages. By using semantic loss to capture the semantic information of sentence embeddings as much as possible, and using consistency loss to preserve the similarity information between sentence embeddings, *SQAE* can improve the generalization and robustness of constructing sentence partitions in an unsupervised manner, boosting the performance of *SoTW*.

Meanwhile, we observe that SIR and X-SIR have better performance when handling rewrite attacks. Since they tried to fix the token partitions as much as possible (e.g., marking synonym tokens into the same partition), they can successfully handle rewrite attacks. However, this operation will incorrectly mark those generated texts that have not been watermarked, resulting in incorrect detection (e.g., 3.2% decrease in no attack setting). Moreover, they still cannot deal with translation attacks. Meanwhile, k-SEMSTAMP projects sentences with different semantics into the same partition, which may improve its ability to resist attacks, but higher aggregation and lower distinguishability significantly reduce the quality of the embedded watermark. In contrast, *SoTW* generates discrete representations to directly project the sentence into different partitions and use two optimization targets to ensure the generalization of the partitions, thus achieving better performance over different scenarios.

To further evaluate model performance, we perform Friedman test (Friedman, 1937) at 5% significance level. The null hypothesis that all algorithms perform equally is rejected. The average ranks of KWG, UNIGRAM, SIR, X-SIR, RIW, WatME, SEMSTAMP, k-SEMSTAMP, and *SoTW* are 3.48, 5.81, 5.24, 4.15, 8.39, 4.86, 6.80, 4.63, and 1.65, respectively (*the lower rank, the better*). Then, Nemenyi test (Nemenyi, 1963) is performed as a post-hoc test. Figure 4 provides a Critical Difference (CD) diagram illustrating the average ranks of each algorithm marked along the axis. The results indicate that *SoTW* is significantly better than its rivals when the critical difference is 1.90.

4.3 Watermark Text Quality

We also conduct experiments to evaluate the impact of text watermark on text quality and summarize results in Figure 5. Here low perplexity denotes better performance. We observe that *SoTW* achieves comparable or lower perplexity than advanced sentence-level baselines, and even has better performance than token-level baselines on LLaMA2-7B-Chat, demonstrating the superiority of *SoTW*. In contrast, advanced sentence-level baselines struggle to generate green-marked partition sentences due to their partition method and distance calculation. We also provide examples of watermark text in the Appendix C for a more intuitive understanding.

4.4 Detailed Analysis

To figure out which part plays a more important role in *SQAE*, we conduct detailed analyses on the learning strategies, hyperparameter λ in Eq.(6), and embedding backbones. We also verify the impact of different codebook sizes in the Appendix D.

Learning Strategy. Since semantic loss in Eq.(4) is the core target of *SoTW*, here we just verify the impact of consistency loss in Eq.(6). According to Table 3, the existence of the consistency loss significantly enhances *SQAE* to group similar sentences within the same partition and to separate sentences with different meanings into distinct spaces (7.2% and 12.9% improvements on *Similar*

Setting	Similar	Dissimilar	Avg.
w/o consistency loss	70.63%	56.09%	63.36%
w consistency loss	75.76%	63.38%	69.57%

Table 3: Results (<u>*Acc*</u>↑) with/without consistency loss.

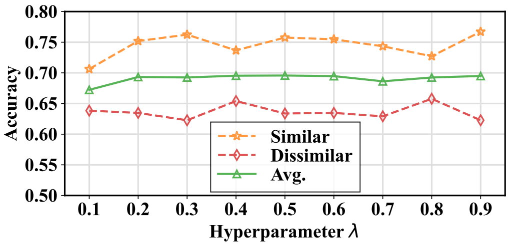


Figure 6: Impact of Hyperparameter λ on *SQAE*.

and *Dissimilar*). This indicates that the consistency loss effectively helps the model build partition boundaries, improving the robustness of the learned discrete representations.

Hyperparameter λ. In Eq.(6), we employ λ to balance the semantic loss and consistency loss. To evaluate its impact, we conduct parameter sensitive test and report results in Figure 6. From the figure, we observe that model performance first increases and then decreases slightly. The best results are achieved when $\lambda \in [0.4, 0.6]$, which is in line with our expectations. For the increasing part, *SoTW* focuses more on the boundaries of sentence partitions, which is essential for watermarking, thus model performance increasing. For the latter slightly fluctuating part, *SoTW* pays more attention to the semantic loss, which may cause the partition boundaries to not be so clear, leading to the incorrect projection of sentences.

Embedding Models. To further investigate the impact of different embedding models, we select four advanced pre-trained embedding models for comparison, whose results are summarized in Table 4. From the table, we have the following observations. Since BERT is not specially designed to obtain sentence embeddings, it tends to generate embeddings with high similarity (Liu et al., 2024), resulting in difficulty for *SQAE* to construct robust partition boundaries. In contrast, sentence BERT is specifically designed for sentence embeddings. We can observe that *SQAE* with sentence BERT can better deal with semantically similar sentences and dissimilar sentences. This phenomenon is also in line with our intuition. Better sentence embeddings can provide more information for *SoTW* to generate robust discrete representations, thus generating better watermarking. Although Sentence-BERT achieves comparable results, its applicability is limited to approximately 50 languages. Therefore, we opt for BGE-M3 as the embedding model.

Model	Similar	Dissimilar	Avg.
BERT	41.63%	70.29%	55.96%
Sentence-BERT	75.57%	69.83%	72.70%
Compositional-BERT	64.64%	71.13%	67.88%
BGE-M3	75.76%	63.38%	69.57%

Table 4: Results (<u>*Acc*</u> ↑) on various embedding models.

5 Conclusion

In this paper, we argued that existing sentence-level watermark algorithms for LLMs suffered from the lack of generalizability in constructing sentence partitions, showing vulnerable performance in OOD scenarios and attacks. In response, we proposed to construct sentence partitions by directly generating discrete representations of sentence partitions and developed a novel *SoTW*. Specifically, we first employed a pre-trained embedding model to generate sentence embeddings. Then, we designed a novel *SQAE* to project sentence embeddings into discrete representations for the watermarking. To enhance the robustness of the learned watermark, we designed a semantic loss to help *SoTW* maintain as much information as possible for discrete representations of sentence partitions, and developed a consistency loss to improve the robustness of learned sentence partition boundaries. Finally, extensive experiments against different attacks over different datasets and backbone LLMs demonstrate the effectiveness of *SoTW*.

Limitations

To inspire future work, we summarize some limitations of our proposed *SoTW* as follows:
1) Although *SoTW* improves the robustness of text watermarks, limited by the performance of *SQAE*, the watermark detection rate is still significantly reduced after attacks. How to further improve the accuracy of constructing sentence partitions needs to be investigated.
2) *SoTW* achieves fast detection of watermarked text. However, due to multiple sampling of sentences during the watermark generation, text generation with *SoTW* is slower than that without watermarks. This represents a limitation that warrants further discussion and exploration in future work.

Despite these limitations, we believe our work serves as an important catalyst in the field, contributing positively to the advancement of more robust text watermark techniques.

References

Anthropic. 2023. Claude. https://www.anthropic.com/claude. Accessed: November 18, 2024.

Daniel Cer, Mona Diab, Eneko E Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and cross-lingual focused evaluation. In *The 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14.

Moses S Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 2318–2335.

Liang Chen, Yatao Bian, Yang Deng, Deng Cai, Shuaiyi Li, Peilin Zhao, and Kam-Fai Wong. 2024b. Watme: Towards lossless watermarking through lexical redundancy. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9166–9180.

Milton Friedman. 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200):675–701.

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 *Association for Computational Linguistics*, pages 4115–4129.

Abe Hou, Jingyu Zhang, Tianxing He, Yichen Wang, Yung-Sung Chuang, Hongwei Wang, Lingfeng Shen, Benjamin Van Durme, Daniel Khashabi, and Yulia Tsvetkov. 2024a. Semstamp: A semantic watermark with paraphrastic robustness for text generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4067–4082.

Abe Bohan Hou, Jingyu Zhang, Yichen Wang, Daniel Khashabi, and Tianxing He. 2024b. k-semstamp: A clustering-based semantic watermark for detection of machine-generated text. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 1706–1715.

David C Howell. 1992. *Statistical methods for psychology*. PWS-Kent Publishing Co.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613.

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.

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.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. 2024. A semantic invariant robust watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

Stuart Lloyd. 1982. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.

Microsoft. 2023. Copilot. https://www.copilot.microsoft.com. Accessed: November 18, 2024.

Peter Bjorn Nemenyi. 1963. *Distribution-free multiple comparisons*. Princeton University.

OpenAI. 2023. ChatGPT. https://openai.com/blog/chatgpt. Accessed: November 18, 2024.

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

Yubing Ren, Ping Guo, Yanan Cao, and Wei Ma. 2024. Subtle signatures, strong shields: Advancing robust and imperceptible watermarking in large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 5508–5519.

Matthias C Rillig, Marlene Ågerstrand, Mohan Bi, Kenneth A Gould, and Uli Sauerland. 2023. Risks and benefits of large language models for the environment. *Environmental Science & Technology*, 57(9):3464–3466.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems*, 30.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2018*, pages 1112–1122. Association for Computational Linguistics (ACL).

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, et al. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*.

Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang. 2024. Provable robust watermarking for ai-generated text. In *The Twelfth International Conference on Learning Representations*.

A Notations

We summarize the necessary notations in Table 5 used in this paper.

Notation	Explanation
$\mathbb{M}$	LLMs
$\mathbb{E}$	A pre-trained embedding model
$\mathbb{Q}$	A trained *SQAE*
$\boldsymbol{S}_{:t}$	The sequence of the previous t sentences
s_i	The i^{th} sentence
$\boldsymbol{e}_i$	The sentence embedding of s_i
$\boldsymbol{z}_i$	The latent representation of $\boldsymbol{e}_i$
$\mathcal{R}^{n\times d}$	A codebook containing n embeddings of dimension d
$\boldsymbol{c}_i$	The i^{th} embedding in $\mathcal{R}^{n\times d}$

Table 5: Notations and explanations in *SoTW*.

B Additional Watermark Robustness Results

We include additional experimental results on watermark robustness, i.e., translate watermark text into Japanese and French using *GPT4o-mini* with prompt "*Translate the following English into French (Japanese)*", as shown in Table 6. We observe that *SoTW* achieves almost the best results when translated into multiple languages, proving the robustness of *SoTW* against translation attacks. This observation is consistent with the results in Section 4.2. Moreover, after translating into Japanese, which is quite different from English, the detection effect of our proposed algorithm is almost doubled compared to existing token-level watermark algorithms. This provides solid proof for the superiority of *SoTW*.

C Case Study

We present some examples of *SoTW* generated watermark text.

D Analysis of Different Codebook Size

Different codebook sizes construct sentence partitions with different sizes. To evaluate the impact of codebook size on sentence partitioning, we conduct experiments summarized in Table 7. The results show that as the codebook size increases, the distribution of sentences becomes more dispersed (i.e., higher Dissimilar values). The suboptimal results are achieved with n=64. This result can be attributed to the fact that a larger number of sentence partitions allows sentences to select partitions that are more relevant to their semantics, facilitating sentences with different semantics to be assigned

Setting	Algorithm	C4 dataset						LFQA dataset						Avg.↑
		LLaMA2-7B-Chat			Baichuan2-7B-Chat			LLaMA2-7B-Chat			Baichuan2-7B-Chatl			
		F1↑ 1%FPR	F1↑ 5%FPR	AUC↑	F1↑ 1%FPR	F1↑ 5%FPR	AUC↑	F1↑ 1%FPR	F1↑ 5%FPR	AUC↑	F1↑ 1%FPR	F1↑ 5%FPR	AUC↑	
Translate French	KGW$_{\text{ICML'23}}$	4.27	15.47	55.87	6.17	16.43	58.12	2.35	9.80	51.43	4.64	14.18	55.70	24.54
	UNIGRAM$_{\text{ICLR'24}}$	33.88	**60.74**	**88.73**	0.40	3.01	32.09	**40.44**	**64.34**	**89.43**	-	1.14	30.29	37.04
	SIR$_{\text{ICLR'24}}$	0.80	1.51	30.08	0.40	2.64	29.39	0.79	3.04	30.69	0.79	1.54	19.55	10.10
	X-SIR$_{\text{ACL'24}}$	5.77	13.17	58.96	1.58	18.65	63.88	7.25	21.47	62.99	2.36	7.72	61.82	27.14
	RIW$_{\text{ACL'24 (Findings)}}$	-	-	-	-	-	-	-	-	-	-	-	-	0
	WatME$_{\text{ACL'24}}$	4.26	11.83	56.14	17.36	34.12	74.40	7.62	25.58	65.14	23.43	35.68	70.43	35.50
	SEMSTAMP$_{\text{NAACL'24}}$	1.18	4.16	49.41	4.26	8.15	53.36	0.79	4.10	50.74	2.73	9.78	52.82	20.12
	k-SEMSTAMP$_{\text{ACL'24 (Findings)}}$	1.96	25.29	74.08	23.43	60.56	**87.68**	2.73	9.85	66.06	24.96	**64.68**	**85.10**	43.87
	SoTW (Ours)	**37.62**	60.19	82.88	**45.33**	**61.84**	85.56	30.59	53.96	79.39	**29.73**	59.06	80.76	**58.91**
Translate Japanese	KGW$_{\text{ICML'23}}$	4.65	23.53	63.88	11.24	24.41	63.93	6.15	21.12	60.97	15.69	30.97	66.95	32.79
	UNIGRAM$_{\text{ICLR'24}}$	-	1.89	22.51	-	-	2.67	-	1.89	19.62	-	-	1.17	4.15
	SIR$_{\text{ICLR'24}}$	-	-	10.66	2.35	13.83	32.97	0.40	1.15	11.11	0.40	3.38	25.56	8.48
	X-SIR$_{\text{ACL'24}}$	-	-	42.25	1.57	6.63	47.06	1.96	11.15	66.12	4.64	18.34	72.57	22.69
	RIW$_{\text{ACL'24 (Findings)}}$	-	-	-	-	-	-	-	-	-	-	-	-	0
	WatME$_{\text{ACL'24}}$	0.40	0.39	18.79	-	0.76	19.34	-	3.00	28.45	0.40	0.39	12.64	7.05
	SEMSTAMP$_{\text{NAACL'24}}$	0.79	3.04	49.82	1.57	3.42	50.88	0.40	4.83	51.03	5.02	11.49	51.84	19.51
	k-SEMSTAMP$_{\text{ACL'24 (Findings)}}$	0.40	14.49	68.82	9.07	41.15	78.29	3.12	9.85	62.14	16.70	**45.86**	**74.18**	35.34
	SoTW (Ours)	**20.60**	**40.18**	**74.81**	**22.18**	**41.63**	**79.76**	**12.29**	**33.86**	**67.24**	**19.68**	43.75	73.07	**44.09**

Table 6: Results of our algorithm (*SoTW*) and its rivals against various attack methods. **Boldface** and underline denote the best and the second best results.

	Text Content	*P*-value
Prompt	A version of this article appeared in the print edition of The Daily Star on July 12, 2018, on page 16.	
Original	A version of this article appeared in the print edition of The Daily Star on July 12, 2018, on page 16.\nAssembled by Palestinian artist Said Baalbaki, the exhibition presents 50 of Abbo's sketches, etchings and object, as well as texts – all taken from Baalbaki's personal collection of work from, and publications on, the elusive sculptor. Baalbaki has no work of his own in this show.\nBorn in Safad, northern Palestine, around 1888-1890, Abbo was a farmer and fisherman who later took up carpentry and stone masonry. It was not until he moved to Berlin in 1911 to study sculpture that his artistic career took shape.\nBaalbaki has spent the last four years finding, and determining attribution for, 70 artworks and 100 texts mentioning the artist.[...truncated]	0.00
Watermarked	A version of this article appeared in the print edition of The Daily Star on July 12, 2018, on page 16. Investors are being warned about a new crypto scam that is sweeping the digital asset space. A message from our editor: Welcome to The Daily Star's news podcast, coming to you live from […]. Investors in cryptocurrency are being warmed about a new scam that involves fake celebrity endorsements and manipulated trading volume. Cryptocurrency investors beware: a new scams that involves fake endorsements by celebrities has been identified by security researchers. [...truncated]	0.99
Prompt	According to Variety, one of the movies about the 1973 tennis match between Billie Jean King and Bobby Riggs is recasting one of its leads.	
Original	According to Variety, one of the movies about the 1973 tennis match between Billie Jean King and Bobby Riggs is recasting one of its leads. No, not the serious one that HBO is doing with Elizabeth Banks and Paul Giamatti, and not the overtly comedic one that Will Ferrell is set to star in. It' s Battle Of The Sexes, the middle one that' s supposed to be a comedy with "dramatic and political overtones" —as we said in an earlier report. Battle Of The Sexes comes from Little Miss Sunshine' s Jonathan Dayton and Valerie Faris, and it was set to star Emma Stone and Steve Carell as the eponymous sexes.[...truncated]	0.01
Watermarked	According to Variety, one of the movies about the 1973 tennis match between Billie Jean King and Bobby Riggs is recasting one of its leads. According to Variety's sources, Emma Stone will no longer be playing the role of Billie Jean king in the upcoming movie \"Battle of the Sexes\". No reason was given for Stone's departure, but it's likely that her schedule won't allow her to take on the role. While Stone has not confirmed anything herself, her Representative did release a statement saying she was no longer going to be part of the film. [...truncated]	0.99

Figure 7: Comparison of original and watermarked text using the LLaMA2-7B-Chat. Green and red sentences are color-coded respectively.

n	Similar	Dissimilar	Avg.
8	75.38%	58.03%	66.70%
16	**78.33%**	53.69%	66.01%
32	72.53%	63.28%	67.91%
64	75.76%	63.38%	69.57%
128	74.43%	62.73%	68.58%
256	71.10%	**68.91%**	**70.01%**

Table 7: Results ($Accuracy$ ↑) of *SQAE* with different codebook sizes. **Boldface** and underline denote the best and the second best results.

to different partitions. However, limited by the ability of the pre-trained embedding model to capture semantic features of sentences, the increased codebook size also increases the number of misclassified semantically similar sentences.