k-SemStamp : A Clustering-Based Semantic Watermark for Detection of Machine-Generated Text

Anonymous ACL submission

Abstract

Recent watermarked generation algorithms inject detectable signatures during language generation to facilitate post-hoc detection. While token-level watermarks are vulnerable to paraphrase attacks, SemStamp (Hou et al., 2023) applies watermark on the semantic representation of sentences and demonstrates promising robustness. SemStamp employs locality-sensitive hashing (LSH) to partition the semantic space with arbitrary hyperplanes, which results in a suboptimal trade-off between robustness and speed. We propose k-SemStamp, a simple yet effective enhancement of SemStamp, utilizing k-means clustering as an alternative of LSH to partition the embedding space with awareness of inherent semantic structure. Experimental results indicate that k-SemStamp saliently improve its robustness and sampling efficiency while preserving the generation quality, advancing a more effective tool for machine-generated text detection.

1 Introduction

To facilitate the detection of machine-generated text (Mitchell et al., 2019), recent watermarked generation algorithms usually inject detectable signatures (Kuditipudi et al., 2023; Yoo et al., 2023; Wang et al., 2023; Christ et al., 2023; Fu et al., 2023; Hou et al., 2023, *i.a.*). A major concern for these approaches is their robustness to potential attacks, since a malicious user could attempt to remove the watermark with text perturbations such as editing and paraphrasing (Krishna et al., 2023; Sadasivan et al., 2023; Kirchenbauer et al., 2023b; Zhao et al., 2023). Hou et al. (2023) propose SemStamp, a paraphrase-robust and sentence-level watermark which assigns signatures to each watermarked sentence according to the locality sensitive hashing (LSH) (Indyk and Motwani, 1998) partitioning of semantic space (explained in 2.1). While demonstrating promising robustness against paraphrase attacks, SemStamp arbitrarily partitions the semantic space by a set of random hyperplanes, potentially splitting groups of semantically similar sentences into different partitions (see Figure 1).

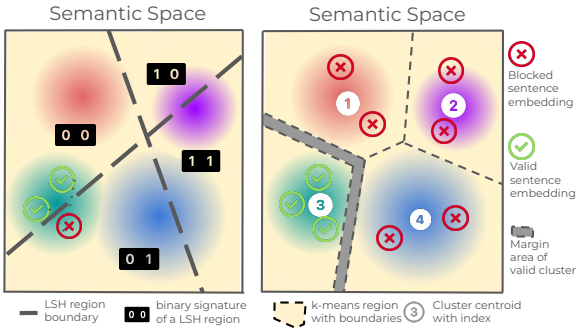

Figure 1: Illustrations of the semantic space. Sentence embeddings with close meanings share similar colors. **(Left)** Random planes from LSH arbitrarily partition the semantic space and split similar sentences into different regions. **(Right)** Margin-based rejection in k-SemStamp. Sentence embeddings which fall into the gray-shaded areas of a valid region will be rejected.

This limitation motivates our proposed method, k-SemStamp (detailed in §2.2), which partitions the space by performing k-means clustering (Lloyd, 1982) on the semantic structure of a given text domain (e.g. news, narratives, etc.). In section 3, we show that the clustering-based partitioning in k-SemStamp greatly improves its robustness against sentence-level paraphrase attacks and sampling efficiency.

2 Approach

We first review some existing watermark algorithms for machine-generated text detection, and introduce our proposed k-SemStamp watermark.

2.1 Preliminaries

Token-Level Watermark Kirchenbauer et al. (2023a) develop a popular token-level watermark algorithm. Given a token history $w_{1:t-1}$, the vocabulary V is pseudo-randomly divided into a "green list" $G^{(t)}$ and a "red list" $R^{(t)}$, where a hash of the

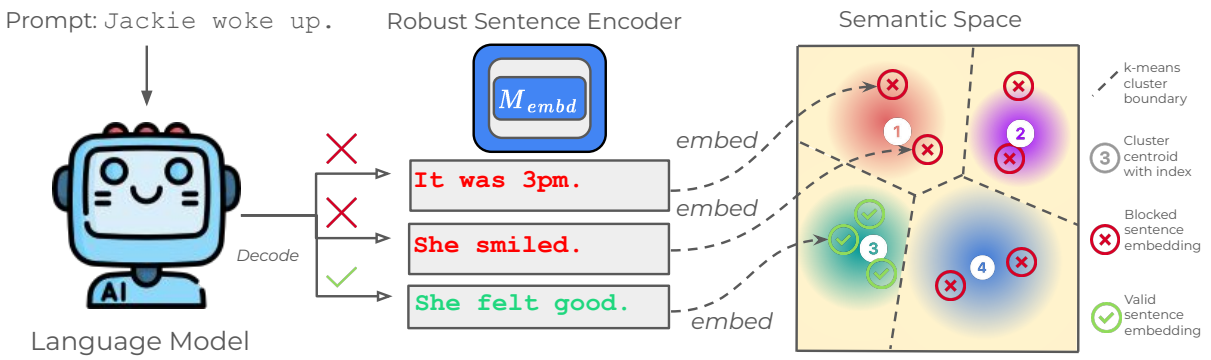

Figure 2: An overview of the proposed k-SemStamp algorithm. k-means clustering partitions the semantic space into semantically similar regions. The sentence generation is accepted if the closest cluster of its sentence embedding corresponds to a "valid" region in the semantic space.

previous token w_{t-1} is used as the seed of the partition. The algorithm then adds a bias to the logits of all tokens in the green-list and sample the next token with an increased probability from the green-list. For a given piece of text, the watermark can be detected by conducting one proportion z-test (detailed in §C) on the number of green list tokens.

SemStamp Under the intuition that common sentence-level paraphrase modifies tokens but preserves sentence meaning, Hou et al. (2023) introduce SemStamp to apply watermark on sentence semantics by partitioning the embedding space with locality sensitive hashing (LSH).

To initialize the LSH partitioning, d normal vectors are randomly sampled from a Gaussian distribution to specify d hyperplanes in the semantic space $\mathbb{R}^h$. For an embedding vector $v \in \mathbb{R}^h$, a d-bit binary LSH signature is assigned, where each digit specifies the position of v in relation to each hyperplane. Each signature $c \in \{0,1\}^d$ indexes a region consisting of all vectors with signature c.

During generation, given a sentence history denoted by $s^{(0)} \dots s^{(t-1)}$, the space of signatures is pseudorandomly partitioned into a set of "valid" regions $G^{(t)}$ and a set of "blocked" region $R^{(t)}$. The LSH signature of the last generated sentenceis used as the random seed to control randomness. A new sentence generation, $s^{(t)}$, will be accepted and if its embedding belongs to any valid region, and rejected otherwise. To detect the watermark in a given piece of text, a one-proportion z-test is performed on the number of sentences whose signatures belong to valid regions (see §C).

2.2 k-SemStamp

As discussed above, in SemStamp, the semantic space is partitioned by random planes from LSH. However, the random planes could separate two semantically similar sentences into two different regions, as depicted in Figure 1. Paraphrasing sentences near the margins of regions may shift their sentence embeddings to a nearby region, resulting in suboptimal watermark strength. This weakness motivates our proposed k-SemStamp, a simple yet effective enhancement of SemStamp that partitions the semantic space with k-means clustering (Lloyd, 1982).

To initialize k-SemStamp , we assume the language model generates text in a specific domain $\mathcal{D}$ (e.g., news articles, scientific articles, etc.). We aim to model the semantic structure of $\mathcal{D}$ and partition its semantic space into k regions. Concretely, we first randomly sample a large number of data from $\mathcal{D}$. We obtain their sentence embeddings with a robust sentence encoder fine-tuned on $\mathcal{D}$ with contrastive learning (detailed in §A). We cluster the sentence embeddings into K clusters with k-means (Lloyd, 1982) and save the cluster centroids. We index a region with $i \in \{1, ..., K\}$ representing the set of all vectors assigned to the i-th centroid.

The generation process is analogous to SemStamp (Hou et al., 2023), as illustrated in Figure 2: given a sentence history $s^{(0)} \dots s^{(t-1)}$, K regions are pseudorandomly partitioned into a set of valid regions $G^{(t)}$ of size $\gamma \cdot K$ and a set of blocked regions $R^{(t)}$ of size $(1-\gamma) \cdot K$, where $\gamma \in (0,1)$ is the ratio of valid regions. The cluster assignment of $s^{(t-1)}$, $C(s^{(t-1)})$, seeds the randomness of the partition at time step t, where $C(.)$ returns the cluster index by finding the closest cluster centroid of the input sentence embedding. We then conduct rejection sampling and only sentences whose embeddings fall into any valid regions (i.e. $C(s) \in G^{(t)}$) are accepted while the rest are rejected. If no valid

sentence is accepted after $N_{\max}$ tries, the last decoded sentence will be chosen. The full algorithm is presented in Algorithm 1.

Cluster Margin Constraint To prevent the sampled sentences from being assigned to a nearby cluster after paraphrasing, we propose a cluster margin constraint. We constrain the sentence embeddings to be sufficiently away from the cluster boundaries (visualized in Figure 1). Concretely, the cosine distance ($d_{\cos}$) of the candidate sentence embedding (v) to the closest centroid (c_q) needs to be smaller than other cluster centroids by at least a margin m:

$$d_{\cos}(v, c_q) < \min_{i \in \{1,\dots,K\} \setminus q} d_{\cos}(v, c_i) - m, \quad (1)$$

where

$$q = \arg\min_{i=1,\dots,K} d_{\cos}(v, c_i). \quad (2)$$

q is the index of the closest cluster centroid to v, and $v = M_{\text{embd}}(s^{(t)})$ is the embedding of the generated sentence at time step t by a robust sentence embedder M_{embd}.

The detection procedure of k-SemStamp is analogous to SemStamp, which is also a one-proportion z-test performed on the number of sentences belong to valid regions, explained in §C and Algorithm 2.

3 Experiments

3.1 Experimental Setup

Following Hou et al. (2023), we conduct paraphrase attack experiments and compare the detection robustness of watermarked generations.

Task and Metrics We evaluate 1000 watermarked generations after paraphrase, respectively on the RealNews subset of the C4 dataset (Raffel et al., 2020) and on the BookSum dataset (Kryściński et al., 2021). We paraphrase watermarked generations sentence-by-sentence with the Pegasus paraphraser (Zhang et al., 2020), Parrot used in Sadasivan et al. (2023), and GPT-3.5-Turbo (OpenAI, 2022). We also implement the strong bigram paraphrase attack as detailed in Hou et al. (2023). Detection robustness of paraphrased watermarked generations is measured with area under the receiver operating characteristic curve (***AUC***) and the true positive rate when the false positive rate is at 1% and 5% (***TP@1%***, ***TP@5%***).[1] Generation quality is measured with perplexity (***PPL***) (using OPT-2.7B (Zhang et al., 2022)), trigram text entropy (Zhang et al., 2018) (***Ent-3***), i.e., the entropy of the trigram frequency distribution of the generated text, and ***Sem-Ent*** (Han et al., 2022), an automatic metric for semantic diversity. Following the setup in Han et al. (2022), we perform k-means clustering ($k = 50$) with the last hidden states of OPT-2.7B on text generations, and Sem-Ent is defined as the entropy of semantic cluster assignments of test generations. We also measure the paraphrase quality with BERTScore (Zhang et al., 2019) between original generations and their paraphrases.

Generation We use OPT-1.3B (Zhang et al., 2022) as our base autoregressive LM. To obtain robust sentence encoders specific to text domains for k-SemStamp generations, we fine-tune two versions of M_{embd}, respectively on RealNews (Raffel et al., 2020) and on BookSum (Kryściński et al., 2021) datasets (See §A for specific procedure and parameter choices)

Following Hou et al. (2023) and Kirchenbauer et al. (2023a), we sample at a temperature of 0.7 and a repetition penalty of 1.05, with 32 being the prompt length and 200 being the default generation length. Results with various lengths are included in Fig. 5. For k-SemStamp , we perform k-means clustering on embeddings of sentences in 8k paragraphs, respectively on RealNews and BookSum. We keep $k = 8$ and a valid region ratio $\gamma = 0.25$, which is consistent with the number of regions in SemStamp, and we use a rejection margin $m = 0.035$.

Baselines Our baselines include popular watermarking algorithms Kirchenbauer et al. (2023a), SemStamp, Unigram-Watermark (Zhao et al., 2023), and the Semantic Invariant Robust (SIR) watermark in Liu et al. (2023), implemented with their recommended setups.

3.2 Results

Detection Detection results in Table 1 show that **k-SemStamp is more robust to paraphrase attacks than KGW (Kirchenbauer et al., 2023a) and SemStamp** across Pegasus, Parrot, and GPT-3.5-Turbo paraphrasers and their bigram attack variants, as measured by AUC, TP@1%, and TP@5%. In particular, k-SemStamp demonstrates considerable robustness against GPT-3.5, in which none of SemStamp and KGW performed strongly. While Unigram-Watermark (Zhao et al., 2023) also

[1] We denote machine-generated text as the "positive" class and human text as the "negative" class. A piece of text is classified as machine-generated when its z-score exceeds a threshold chosen based on a given false positive rate, detailed in §C.

Domain	Algorithm	No Paraphrase	Pegasus	Pegasus-bigram	Parrot	Parrot-bigram	GPT3.5	GPT3.5-bigram
		AUC ↑ / *TP@1%* ↑ / *TP@5%* ↑						
RealNews	KGW	99.6 / 98.4 / 98.9	95.9 / 82.1 / 91.0	92.1 / 42.7 / 72.9	88.5 / 31.5 / 55.4	83.0 / 15.0 / 39.9	82.8 / 17.4 / 46.7	75.1 / 5.9 / 26.3
	SIR	99.9 / 99.4 / 99.9	94.4 / 79.2 / 85.4	94.1 / 72.6 / 82.6	93.2 / 62.8 / 75.9	95.2 / 66.4 / 80.2	80.2 / 24.7 / 42.7	77.7 / 20.9 / 36.4
	SemStamp	99.2 / 93.9 / 97.1	97.8 / 83.7 / 92.0	96.5 / 76.7 / 86.8	93.3 / 56.2 / 75.5	93.1 / 54.4 / 74.0	83.3 / 33.9 / 52.9	82.2 / 31.3 / 48.7
	k-SemStamp	99.6 / 98.1 / 98.7	**99.5 / 92.7 / 96.5**	**99.0 / 88.4 / 94.3**	**97.8 / 78.7 / 89.4**	**97.5 / 78.3 / 87.3**	**90.8 / 55.5 / 71.8**	**88.9 / 50.2 / 66.1**
BookSum	KGW	99.6 / 99.0 / 99.2	97.3 / 89.7 / 95.3	96.5 / 56.6 / 85.3	94.6 / 42.0 / 75.8	93.1 / 37.4 / 71.2	87.6 / 17.2 / 52.1	77.1 / 4.4 / 27.1
	SIR	1.0 / 99.8 / 1.0	93.1 / 79.3 / 85.9	93.7 / 69.9 / 81.5	96.5 / 72.9 / 85.1	97.2 / 76.5 / 88.0	80.9 / 39.9 / 23.6	75.8 / 19.9 / 35.4
	SemStamp	99.6 / 98.3 / 98.8	99.0 / **94.3** / 97.0	98.6 / 90.6 / 95.5	98.3 / 83.0 / 91.5	98.4 / 85.7 / 92.5	89.6 / 45.6 / 62.4	86.2 / 37.4 / 53.8
	k-SemStamp	99.9 / 99.1 / 99.4	**99.3** / 94.1 / **97.3**	**99.1 / 92.5 / 96.9**	**98.4 / 86.3 / 93.9**	**98.8 / 88.9 / 94.9**	**95.6 / 65.7 / 83.0**	**95.7 / 64.5 / 81.4**

Table 1: Detection results against various paraphrase attacks. All numbers in each cell are in percentages and correspond to ***AUC***, ***TP@1%***, and ***TP@5%***, respectively. All three metrics prefer higher values. KGW and SIR refer to the watermarks in Kirchenbauer et al. (2023a) and Liu et al. (2023). **k-SemStamp is more robust than SemStamp and KGW across most paraphrasers and their bigram attack variants and both datasets.**

	PPL↓	*Ent-3*↑	*Sem-Ent*↑
No watermark	11.89	11.43	2.98
KGW	14.92	11.32	2.95
SIR	20.34	11.57	3.18
SemStamp	12.49	11.48	3.00
k-SemStamp	11.82	11.48	2.98

Table 2: Quality evaluation of generations on BookSum. ↑ and ↓ indicate the direction of preference (higher and lower). **k-SemStamp generation quality is on par with non-watermarked generations**.

Prompt: In Chapter 18, Richard begins at Kenge and Carboy's.
Non-Watermarked Generation: He goes to the inn where Mr. Kenge has been let off by the landlord. There, he meets a woman named Hannah, who is looking for him. He asks her where he is wanted.
SemStamp: He meets up with Lydgate, who is there to see if the money from the deal is still there. The lawyers are ready to go to trial, but Richard says he has a better plan. He wants to leave Middlemarch for good.
k-SemStamp : He also sees Adam for the first time since his imprisonment. They discuss the latest updates in their respective personal lives. Adam is living with Dinah and is still angry with Adam for having to leave him.

Figure 3: Generation Examples of k-SemStamp compared with SemStamp. **Both generations are contextually sensible and coherent as compared to non-watermarked generations.** Additional examples after paraphrase are presented in Figure 4 in the Appendix.

demonstrates strong robustness against paraphrase, it has a critical vulnerability to reverse-engineering attacks. We discuss its vulnerability and experimental results in §D. The BERTScores of paraphrases are presented in Table 5.

Sampling Efficiency k-SemStamp not only demonstrates stronger paraphrastic robustness, but also **generates sentences with higher sampling efficiency.** To produce the results on BookSum (Kryściński et al., 2021) in Table 1, k-SemStamp samples 13.3 sentences on average to accept one valid sentence, which is 36.2% less compared to the average 20.9 sentences sampled by SemStamp. We analyze the reasons of candidate sentences for being rejected respectively by k-SemStamp and SemStamp, discovering that around 42.0% and 80.7% of the sentences are rejected due to the margin requirements. Since k-SemStamp determines the cluster centroids by k-means clustering on the semantic structure of a given text domain, the embeddings of most candidate sentences generated in this text domain are closer to the centroids and away from the margins, and they are less likely to relocate to a blocked region after paraphrase.

Quality Table 2 shows that the perplexity, text diversity, and semantic diversity of both SemStamp and k-SemStamp generations are **on par with the base model without watermarking**, while KGW and SIR notably degrade perplexity. Qualitative examples of k-SemStamp are presented in Figure 3 and 4. Compared to non-watermarked generation, k-SemStamp convey the same level of coherence and contextual sensibility. The Ent-3 and Sem-Ent metrics also show that **k-SemStamp preserves token and semantic diversity of generation** compared to non-watermarked generation.

Generation Length As shown in Figure §5, k-SemStamp has higher AUC than Kirchenbauer et al. (2023a) and than SemStamp across most generation lengths by number of tokens.

4 Conclusion

We propose k-SemStamp, a simple but effective enhancement of SemStamp. To watermark generated sentences, k-SemStamp maps embeddings of candidate sentences to a semantic space which is partitioned by k-means clustering, and only accept sampled sentences whose embeddings fall into a valid region. This variant greatly improves the paraphrastic robustness and sampling speed.

Limitations

A core component of k-SemStamp is performing k-means clustering on a particular text domain and partitioning the semantic space according to the semantic structure of the text domain. However, this requires specifying the text domain of generation to initialize k-SemStamp . If the k-means clusters and the sentence embedder are not specific to the text domain, k-SemStamp suffers from a minor drop in paraphrastic robustness (see Table 4 for experimental results with k-SemStamp using a sentence embedder trained on RealNews).

Ethical Considerations

The proliferation of large language models capable of generating realistic texts has drastically increased the need to detect machine-generated text. By proposing k-SemStamp, we hope that practitioners will use this as a tool for governing model-generated texts. Although k-SemStamp shows promising paraphrastic robustness, it is still not perfect for all kinds of attacks and thus should not be solely relied on in all scenarios. Finally, we hope this work motivates future research interests in not only semantic watermarking but also general adversarial-robust methods for AI governance.

References

Miranda Christ, Sam Gunn, and Or Zamir. 2023. Undetectable watermarks for language models. *ArXiv*, abs/2306.09194.

Yu Fu, Deyi Xiong, and Yue Dong. 2023. Watermarking conditional text generation for ai detection: Unveiling challenges and a semantic-aware watermark remedy. *ArXiv*, abs/2307.13808.

Seungju Han, Beomsu Kim, and Buru Chang. 2022. Measuring and improving semantic diversity of dialogue generation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*.

Abe Bohan Hou, Jingyu Zhang, Tianxing He, Yichen Wang, Yung-Sung Chuang, Hongwei Wang, Lingfeng Shen, Benjamin Van Durme, Daniel Khashabi, and Yulia Tsvetkov. 2023. Semstamp: A semantic watermark with paraphrastic robustness for text generation. *arXiv preprint arXiv:2310.03991*.

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*, STOC '98, page 604–613, New York, NY, USA. Association for Computing Machinery.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023a. A watermark for large language models. *arXiv preprint arXiv:2301.10226*.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. 2023b. On the reliability of watermarks for large language models.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *arXiv preprint arXiv:2303.13408*.

Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2021. Booksum: A collection of datasets for long-form narrative summarization. *arXiv preprint arXiv:2105.08209*.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2023. Robust distortion-free watermarks for language models. *ArXiv*, abs/2307.15593.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. 2023. A semantic invariant robust watermark for large language models. *arXiv preprint arXiv:2310.06356*.

Seth Lloyd. 1982. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT*'19, page 220–229, New York, NY, USA. Association for Computing Machinery.

OpenAI. 2022. ChatGPT.

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* (JMLR).

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

Lean Wang, Wenkai Yang, Deli Chen, Haozhe Zhou, Yankai Lin, Fandong Meng, Jie Zhou, and Xu Sun. 2023. Towards codable text watermarking for large language models. *ArXiv*, abs/2307.15992.

John Wieting, Kevin Gimpel, Graham Neubig, and Taylor Berg-kirkpatrick. 2022. Paraphrastic representations at scale. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 379–388, Abu Dhabi, UAE. Association for Computational Linguistics.

Kiyoon Yoo, Wonhyuk Ahn, Jiho Jang, and No Jun Kwak. 2023. Robust multi-bit natural language watermarking through invariant features. In *Annual Meeting of the Association for Computational Linguistics*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning* (ICML).

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. OPT: Open Pre-trained Transformer Language Models. *arXiv preprint arXiv:2205.01068*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations* (ICLR).

Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and William B. Dolan. 2018. Generating informative and diverse conversational responses via adversarial information maximization. In *NeurIPS*.

Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. 2023. Provable robust watermarking for ai-generated text. *arXiv preprint arXiv:2306.17439*.

Supplemental Materials

A Contrastive Learning and Sentence Encoder Fine-tuning

To make sentence encoders robust to paraphrase, we fine-tune following the procedure in Hou et al. (2023) and Wieting et al. (2022).

First, we paraphrase 8000 paragraphs from RealNews (Raffel et al., 2020) and BookSum (Kryściński et al., 2021) using the Pegasus paraphraser (Zhang et al., 2020) through beam search with 25 beams. We then fine-tune two SBERT models[2] with an embedding dimension $h = 768$ for 3 epochs with a learning rate of 4×10^{-5}, using the contrastive learning objective with a margin $\delta = 0.8$:

$$\min_{\theta} \sum_{i} \max\Big\{\delta - f_\theta(s_i, t_i) + f_\theta(s_i, t_i'), 0\Big\}, \quad (3)$$

where f_θ measures the cosine similarity between sentence embeddings, $f_\theta(s,t) = \cos\big(M_\theta(s), M_\theta(t)\big)$, and M_θ is the sentence encoder parameterized by θ that is to be fine-tuned.

B Algorithms

The algorithms of k-SemStamp are presented in Algorithm 1.

C Watermark Detection

The detection of both SemStamp and k-SemStamp follows the one-proportion z-test framework proposed by Kirchenbauer et al. (2023a). The z-test is performed on the number of green-list tokens in Kirchenbauer et al. (2023a), assuming the following null hypothesis:

Null Hypothesis 1. *A piece of text, T, is not generated (or written by human) knowing a watermarking green-list rule.*

The green-list token z-score is computed by:

$$z = \frac{N_G - \gamma N_T}{\sqrt{\gamma(1-\gamma)N_T}}, \quad (4)$$

where N_G denotes the number of green tokens, N_T refers to the total number of tokens contained in the given piece of text T, and γ is a chosen ratio of green tokens.

The z-test rejects the null hypothesis when the green-list token z-score exceeds a given threshold M. During the detection of each piece of text, the number of the green tokens is counted. A higher ratio of detected green tokens after normalization implies a higher z-score, meaning that the text is classified as machine-generated with more confidence.

Hou et al. (2023) adapts this z-test to detect SemStamp, according to the number of valid sentences rather than green-list tokens.

Null Hypothesis 2. *A piece of text, T, is not generated (or written by human) knowing a rule of valid and blocked partitions in the semantic space.*

$$z = \frac{S_V - \gamma S_T}{\sqrt{\gamma(1-\gamma)S_T}}, \quad (5)$$

where S_V refers to the number of valid *sentences*, γ is the ratio of valid sentences out of the total number of sentences S_T in a piece of text T. To detect SemStamp, the given piece of text, T, is first broken into sentences and the number of valid sentences S_V is counted to calculate the z-score. Likewise, the null hypothesis 2 is rejected when the z-score exceeds a threshold M.

The detection procedure of k-SemStamp is analogous to SemStamp. We break a text into sentences and count the number of valid sentences to calculate the z-score, where only the determination of whether a sentence falls into a valid region is different. k-SemStamp assigns the sentence generation to its closest cluster centroid and consider if the index of the cluster centroid belongs to a valid partition. See the full detection algorithm in Algorithm 2.

For a comprehensive evaluation of detection robustness, we consider a range of possible threholds $M_f \in [0, 6.0]$, where each M_f is determined by a given false positive rate r_f, and consider machine-generated text as "positive" and human text as "negative" in a traditional classification setting. We estimate a false positive rate of corresponding M_f by computing the misclassification rate of human text samples. We let $r_f = 0.01$ and $r_f = 0.05$ to respectively measure ***TP@1%*** and ***TP@5%*** metrics in Table 1.

D Additional Experimental Results

Table 3 shows the detection results of Unigram-Watermark (Zhao et al., 2023) against paraphrase attacks, demonstrating more robustness compared to SemStamp and k-SemStamp . However, Unigram-Watermark has the key vulnerability of being readily reverse-engineered by

[2] sentence-transformers/all-mpnet-base-v1

Algorithm 1 k-SEMSTAMP text generation algorithm and subroutines

Input: language model P_{LM}, prompt $s^{(0)}$, the text domain $\mathcal{D}$, number of sentences to generate T.
Params: sentence embedding model fine-tuned on $\mathcal{D}$, $M^{\mathcal{D}}_{\text{embd}}$ with embedding dimension h, maxout number N_{max}, margin $m > 0$, valid region ratio $\gamma \in (0, 1)$, number of k-means clusters K, a large prime number p, a large integer N.
Output: generated sequence $s^{(1)} \dots s^{(T)}$.

procedure k-SEMSTAMP
$C_K \leftarrow$ INITIALIZE$(\mathcal{D}, K)$ to initialize K cluster centroids based on $\mathcal{D}$.
for $t = 1, 2, \dots, T$ **do**
1. Find the index of the closest cluster centroid of the previously generated sentence, $c_q^{(t-1)} \leftarrow$ ASSIGN$(s^{(t-1)}, C_K)$, and use $c_q^{(t-1)} \cdot p$ as the seed to randomly divide the set of clusters C_K into a "valid region set" $G^{(t)}$ of size $\gamma \cdot K$ and a "blocked region set" $R^{(t)}$ of size $(1-\gamma) \cdot K$.
2. **repeat** Sample a new sentence from LM,
   until the index of the closest cluster centroid of the new sentence is in the "valid region set", $c_q^{(t)}$ and the margin requirement
   MARGIN$(s^{(t)}, m)$ is satisfied.
   or has repeated N_{max} times
3. Append the selected sentence $s^{(t)}$ to context.

end for
return $s^{(1)} \dots s^{(T)}$
end procedure

function INITIALIZE$(\mathcal{D}, K)$
$\mathcal{D}'_N \sim \mathcal{D}$ // sample N sentences from D
$C_K \leftarrow$ K-MEANS$(\mathcal{D}'_N, K)$ // obtain k cluster centroids
return C_K
end function

function ASSIGN(s, C_K)
$c_q \leftarrow min_{i=1,\dots,K} |d_{\cos}(v, c_i)|$ // find the index of the closest cluster centroid by cosine distance
return c_q
end function

		AUC / TP@1% / TP@5%			
Algorithm	Domain	Pegasus	Pegasus-bigram	Parrot	Parrot-bigram
UNIGRAM-WATERMARK	`RealNews`	99.1 / 92.2 / 96.4	98.4 / 87.9 / 94.3	98.9 / 82.7 / 94.0	98.7 / 79.6 / 91.5
	`BookSum`	99.4 / 96.4 / 99.0	99.7 / 91.6 / 98.2	99.5 / 91.6 / 97.7	99.6 / 87.8 / 97.2

Table 3: Detection results of UNIGRAM-WATERMARK in Zhao et al. (2023)

an adversary. Since UNIGRAM-WATERMARK can be understood as a variant of the watermark in Kirchenbauer et al. (2023a) but with only one fixed greenlist initialized at the onset of generation. An adversary can reverse-engineer this greenlist by brute-force submissions to the detection API of $|V|$ times, where each submission is repetition of a token w_i, $i \in \{1, ..., |V|\}$ drawn without replacement from the vocabulary V of the tokenizer. Therefore, upon each submission to the detection API, the adversary will be able to tell if the submitted token is in the greenlist or not. After $|V|$ times of submission, the entire greenlist can be reverse-engineered. On the other hand, such hacks are not applicable to SEMSTAMP and k-SEMSTAMP , since both algorithms do not fix the list of valid regions and blocked regions during generation. In summary, despite having strong robustness against various paraphrase attacks, UNIGRAM-WATERMARK has a notable vulnerability that may limit its applicability in high-stake domains where adversaries can conduct reverse-engineering.

Computing Infrastruture and Budget We ran sampling and paraphrase attack jobs on 8 A40 and 4 A100 GPUs, taking up a total of around 200 GPU hours.

Algorithm 2 k-SEMSTAMP text detection algorithm

Input: a piece of text T, saved k-kmeans cluster centroids C_K
Params: sentence embedding model finetuned on $\mathcal{D}$, $M^{\mathcal{D}}_{\text{embd}}$, z-threshold range Z, human-written texts H, a large prime number p, valid region ratio $\gamma \in (0, 1)$, number of k-means clusters K.
Output: a z-score based on the ratio of detected sentences.
procedure DETECT(T, C_K)
  $s_1, ..., s_N \leftarrow$ SENTENCE-TOKENIZE(T)
  $c_q^{(1)} \leftarrow$ ASSIGN(s_1, C_K)
  seed $\leftarrow c_q^{(1)} \cdot p$
  $G^{(1)} \leftarrow$ RANDOM-SAMPLE(seed, K, γ) // pseudo-randomly sample a set of cluster centroid indices of size $K \cdot \gamma$, where the randomness of sampling is controlled by seed.
  for $t = 2, \ldots, N$ **do**
    $c_q^{(t)} \leftarrow$ ASSIGN(s_t, C_K)
    if $c_q^{(t)} \in G^{(t-1)}$ **then**
      S_V += 1
    end if
   SEED $\leftarrow c_q^{(t)} \cdot p$
   $G^{(t)} \leftarrow$ RANDOM-SAMPLE(seed, K, γ)
  end for
  $z \leftarrow \frac{S_V - \gamma N}{\sqrt{\gamma(1-\gamma)N}}$
  return z

Prompt: Pompey has a meeting with the pirates and they discuss the fact that Antony is in Egypt with Cleopatra.
SEMSTAMP: The pirates want to ransom him, but they aren't sure how high the price would be. They have heard that Antony is a powerful monarch who can do whatever he wants. The pirates also say that he has a lot of money. Pompey says it wouldn't matter if he had 1,000 times as much money as Antony; he would still be a tyrant. The conversation turns to the bribes Antony was accused of taking. One pirate says that if Antony were a man of honor, he would have given up his rights to the Sea-Sir by now.
Paraphrased SEMSTAMP: The pirates don't know how much they would get for him. They have heard that Antony can do whatever he wants. He has a lot of money according to the pirates.He would still be a tyrant if he had more money than Antony. Antony was accused of taking bribes. According to a pirate, Antony would have given up his rights to the Sea-Sir by now if he were a man of honor.

k-SEMSTAMP : They also discuss Caesar Augustus. The captain invites Pompey to share his thoughts on Caesar. He says that he doesn't like him very much. In fact, he thinks he's probably a bad leader. He goes on to say that Brutus should be his guest. Brutus replies that he can't because he's already got a guest for that matter - the Senate's spaghetti-spilling friend, Publius Cornelius.
Paraphrased k-SEMSTAMP : They talked about Caesar Augustus. Pompey was invited by the captain to share his thoughts on Caesar. He doesn't like him very much. He thinks he's a bad leader. He said that he should be his guest. Publius Cornelius is the Senate's spaghetti-spilling friend and he can't because he's already there.

Figure 4: Examples of k-SEMSTAMP after being paraphrased by Pegasus Paraphraser (Zhang et al., 2020). Green and plain sentences are detected, while red and underlined sentences are not. **k-SEMSTAMP generations are more robust to paraphrase, having a higher detection z-score than SEMSTAMP.**

Algorithm	Train Domain	Test Domain	*AUC / TP@1% / TP@5%* Pegasus	Pegasus-bigram	Parrot	Parrot-bigram
k-SemStamp	RealNews	BookSum	98.2 / 78.2 / 94.9	97.3 / 70.7 / 93.8	96.8 / 65.5 / 90.9	96.4 / 61.9 / 89.2
	BookSum	BookSum	**99.3 / 94.1 / 97.3**	**99.1 / 92.5 / 96.9**	**98.4 / 86.3 / 93.9**	**98.8 / 88.9 / 94.9**

Table 4: Detection results of k-SemStamp with a sentence encoder only fine-tuned on RealNews and tested on BookSum. k-SemStamp is able to generalize some level of paraphrastic robustness across domains.

	RealNews			BookSum		
***Algorithm*↓ *Paraphraser*→**	Pegasus	Parrot	GPT3.5	Pegasus	Parrot	GPT3.5
KGW	71.0 / 66.6	57.1 / 58.4	54.8 / 53.3	71.8 / 69.3	62.0 / 61.8	60.3 / 56.7
SStamp	72.2 / 69.7	57.2 / 57.4	55.1 / 53.8	73.0 / 71.3	64.4 / 67.1	55.4 / 50.0
k-SStamp	71.9 / 67.8	55.8 / 56.1	54.8 / 53.3	73.5 / 71.5	64.2 / 67.1	35.7 / 33.4

Table 5: BERTScore (Zhang et al., 2019) between original and paraphrased generations under different watermark algorithms and paraphrasers. All numbers are expressed in percentages. The first number in each entry is the result under regular sentence-level paraphrase attack in Hou et al. (2023), while the second number is the result under the bigram paraphrase attack. **Compared to regular paraphrase attacks, bigram paraphrase attack only slightly corrupts the semantic similarity between paraphrased outputs and original generations.**

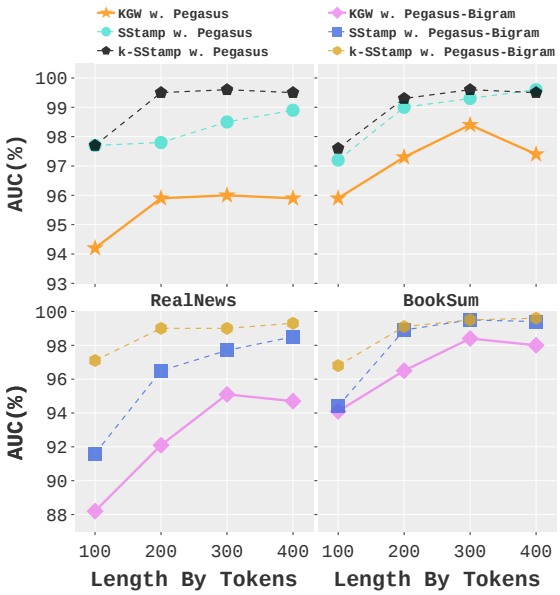

Figure 5: Detection results (AUC) under different generation lengths. **k-SemStamp is more robust than SemStamp and KGW across length 100-400 tokens in most cases.**