# SemStamp: A Semantic Watermark with Paraphrastic Robustness for Text Generation

Anonymous ACL submission

## Abstract

Existing watermarked generation algorithms employ *token-level* designs and therefore, are vulnerable to paraphrase attacks. To address this issue, we introduce watermarking on the *semantic representation* of sentences. We propose SemStamp, a robust sentence-level semantic watermarking algorithm that uses locality-sensitive hashing (LSH) to partition the semantic space of sentences. The algorithm encodes and LSH-hashes a candidate sentence generated by a language model, and conducts rejection sampling until the sampled sentence falls in watermarked partitions in the semantic embedding space. To test the paraphrastic robustness of watermarking algorithms, we propose a "bigram paraphrase" attack that produces paraphrases with small bigram overlap with the original sentence. This attack is shown to be effective against existing token-level watermark algorithms, while posing only minor degradations to SemStamp. Experimental results show that our novel semantic watermark algorithm is not only more robust than the previous state-of-the-art method on various paraphrasers and domains, but also better at preserving the quality of generation.

## 1 Introduction

This work focuses on algorithms for detecting machine-generated text via *watermarked generation*—adding signatures during text generation which are algorithmically detectable, yet are imperceptible to human eye (Atallah et al., 2001). This problem is of extreme importance now that large language models (LLMs) such as GPT-4 (OpenAI, 2023) generate realistic text, increasing risks of LLM misuse, such as generation of misinformation, impersonation, and copyright infringements (Weidinger et al., 2021; Ippolito et al., 2022; Pagnoni et al., 2022; House, 2023).

The dominant body of recent works on watermarked generation operate by injecting token-level signatures during decoding time (Kuditipudi et al., 2023; Yoo et al., 2023; Wang et al., 2023; Christ et al., 2023; Fu et al., 2023, *i.a.*). As a representative example, Kirchenbauer et al. (2023a) propose a watermarked generation algorithm that injects watermark signals that are extracted based on the previously generated *tokens*. Despite its efficiency, follow-up work has shown that corrupting the generated text, especially paraphrasing, could weaken its robustness (Krishna et al., 2023; Sadasivan et al., 2023; Kirchenbauer et al., 2023b).

We propose SemStamp, a *semantic watermark algorithm* that is robust to sentence-level paraphrase attacks (§2.2). Depicted in Figure 1, our core intuition is that while paraphrasing alters the surface-form tokens, the sentence-level semantics are unchanged. Thus, instead of partitioning the vocabulary, our watermark operates on the semantic space of sentence embeddings, partitioned by locality-sensitive hashing (LSH; Indyk and Motwani, 1998; Charikar, 2002). We develop two key components—a sentence encoder trained with contrastive learning (CL; Wieting et al., 2022) and a margin-based constraint—to enhance paraphrastic robustness.

To stress-test the robustness of watermarking algorithms, we develop a novel attack method that minimizes bigram overlap during paraphrasing, and name it the bigram paraphrase attack (§2.3). Experimental results (§3) demonstrate that our proposed semantic watermark remains effective while token-level watermarks suffer significantly from the bigram attack.

We summarize our main contributions as follows. First, we propose a sentence-level semantic watermark for LLMs and show that it is robust to paraphrasing and more quality-preserving than a token-level watermark algorithm. Second, we develop a novel attack method for watermarking algorithms, namely the bigram paraphrase attack, which can effectively weaken token-level watermarking but

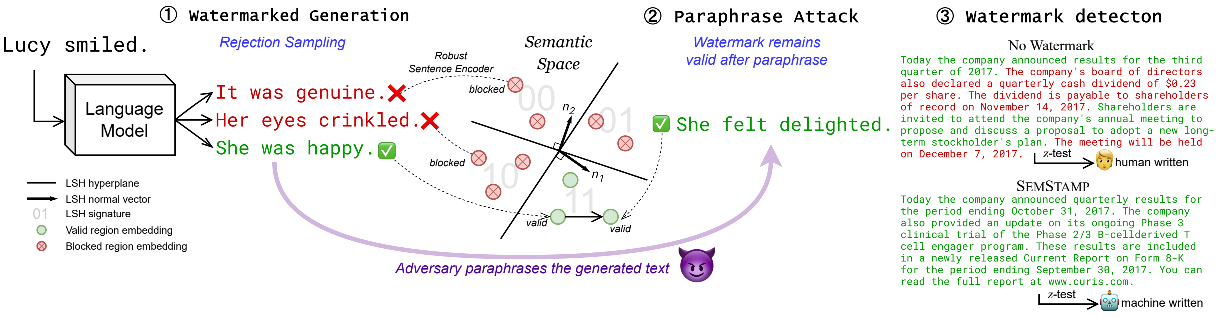


Figure 1: An overview of the proposed SemStamp algorithm. **Left**: During generation, the watermark is injected by mapping candidate sentences into embeddings through a robust sentence encoder, dividing the semantic space through locality-sensitive hashing, and rejection sampling from the LM to generate sentences with valid region embeddings. **Right**: Detection is determined by the number of valid sentences in a candidate generation.

only poses minor degradations to our semantic watermark. Third, we fine-tune a paraphrase-robust sentence encoder with a contrastive learning objective and develop a rejection margin constraint to enhance the paraphrastic robustness of our semantic watermark algorithm.[1]

## 2 Approach

### 2.1 Preliminaries

**Text Generation from Autoregressive LMs** An autoregressive LM, denoted by $P_{\text{LM}}$, models the conditional distribution of the next token over the vocabulary $V$. Given a token history $w_{1:t} = w_1, \dots, w_t$ where each token $w_i \in V$, the next token is generated by sampling $w_{t+1} \sim P_{\text{LM}}(\cdot | w_{1:t})$. We introduce a sentence-level notation: $s^{(t+1)} \sim P_{\text{LM}}(\cdot | s^{(1)} \dots s^{(t)})$ refers to the sampling of the next sentence given sentence history $s^{(1)} \dots s^{(t)}$.

**Detecting Machine-Generated Text through Watermarking** The goal of watermarked generation (Kuditipudi et al., 2023; Zhao et al., 2023, *i.a.*) is to facilitate the detection of machine-generated text. A watermarked generation algorithm adds a statistical signal during the decoding stage of LLMs. The watermarked text is then provided to the user. At the detection stage, a piece of text is classified as machine-generated if the watermark is detected. Because malicious users could postprocess LLM-generated texts before detection, it is crucial that the watermark remains detectable under various text perturbations attacks, including text insertion, substitution, deletion, and paraphrasing.

**Token-Level Watermarking and its Susceptibility to Paraphrase Attacks** Kirchenbauer et al. (2023a) propose a watermark that is injected at the token level. At each time step of the generation, the vocabulary $V$ is pseudorandomly partitioned into a "green list" and a "red list". The random seed for partition is computed by a hash of the previously generated token. A globally fixed bias parameter $\delta > 0$ is added to the logit of each green list token so that the LLM is induced to generate more green list tokens. The watermark is detected by conducting one proportion $z$-test (detailed in §B) on the number of green list tokens in the generated text.

Because of the token-level nature of the watermark algorithm, perturbing a token $w_t$ in a generated sequence $w_{1:T}$ through paraphrasing would change the green list for token $w_{t+1}$. As a result, a green token $w_{t+1}$ might be considered red, which undermines the detectability of the watermark (Krishna et al., 2023). Moreover, because the watermark changes logits directly, it can degrade the quality of generated text (Fu et al., 2023).

**Locality-Sensitive Hashing** We will use LSH (Indyk and Motwani, 1998) to partition the semantic embedding space. It hashes similar inputs into similar signatures, thereby reducing the dimensionality and providing a similarity measure for a high-dimensional input space $\mathbb{R}^h$. Given an LSH dimension $d$, we adopt the cosine-preserving method from Charikar (2002) which produces a $d$-bit binary signature through random hyperplane projections, and each hyperplane is represented by a random normal vector $n^{(i)}$ drawn from the $h$-dimensional Gaussian distribution.[2] The LSH signature for an embedding vector $v \in \mathbb{R}^h$ is then determined by the sign of the dot product between the candidate vector and the normal vectors: $\text{LSH}_i : \mathbb{R}^h \mapsto \{0, 1\}$ which gives the $i$-th digit signature, is defined by $\text{LSH}_i(v) = \mathbb{1}\left(n^{(i)} \cdot v > 0\right)$[3],

[1] Our code, model, and data will be released publicly.

[2] Normal vector $n^{(i)} \in \mathbb{R}^h$ represents the hyperplane that is orthogonal to $n^{(i)}$ and passes through the origin.

[3] $\mathbb{1}(\cdot)$ is the indicator function.

**Algorithm 1** SEMSTAMP text generation algorithm

**Input:** language model $P_{\text{LM}}$, prompt $s^{(0)}$, number of sentences to generate $T$.
**Params:** sentence embedding model $M_{\text{embd}}$ with embedding dimension $h$, maxout number $N_{\text{max}}$, margin $m > 0$, valid region ratio $\gamma \in (0, 1)$, LSH dimension $d$, a large prime number $p$.
**Output:** generated sequence $s^{(1)} \dots s^{(T)}$.

**procedure** SEMSTAMP
  **init** LSH$(\cdot)$, randomly initialize $d$ vectors $n^{(1)} \dots n^{(d)} \in \mathbb{R}^h$, to create $2^d$ semantic subspaces.
  **for** $t = 1, 2, \dots, T$ **do**
    1. Compute the LSH signature of the previously generated sentence, SIG$(s^{(t-1)})$, and use $[\text{SIG}(s^{(t-1)})]_{10} \cdot p$ as the seed to randomly divide the space of signatures $\{0, 1\}^d$ into a "valid region set" $G^{(t)}$ of size $\gamma \cdot 2^d$ and a "blocked region set" $R^{(t)}$ of size $(1 - \gamma) \cdot 2^d$.
    2. **repeat** Sample a new sentence from LM,
       **until** the signature of the new sentence is in the "valid region set", SIG$(s^{(t)}) \in G^{(t)}$ and the margin requirement MARGIN$(s^{(t)}, m)$ is satisfied.
       **or** has repeated $N_{\text{max}}$ times
    3. Append the selected sentence $s^{(t)}$ to context.
  **end for**
  **return** $s^{(1)} \dots s^{(T)}$
**end procedure**

**Algorithm 2** SEMSTAMP subroutines

**function** SIG$(s)$
  $v \leftarrow M_{\text{embd}}(s)$ // obtain embeddings of sentence $s$
  $c \leftarrow \text{LSH}(v)$ // obtain signature $c$ of the embedding
  **return** $c$
**end function**

**function** MARGIN$(s, m)$
  $v \leftarrow M_{\text{embd}}(s)$ // obtain embeddings of sentence $s$
  $x \leftarrow \min_{i=1,\dots,d}\{|\cos(v, n^{(i)})|\}$ // compute the minimum distance between $v$ and all LSH normal vectors $n^{(i)}$.
  **return** `True` **If** $x \geq m$ **Else** `False`
**end function**

and $\text{LSH}(v) = [\text{LSH}_1(v)||\dots||\text{LSH}_d(v)]$ is the concatenation of all $d$ digits.

### 2.2 SEMSTAMP: A Semantic Watermark with Paraphrastic Robustness

We begin with a high-level overview of the SEMSTAMP (Alg. 1). Our approach is motivated by the intuition that paraphrasing alters the surface-form tokens but preserves sentence-level semantics. We apply the watermark at the sentence-level semantic space (instead of the token-level vocabulary) to preserve the watermark under token changes. To do so, we use a semantic sentence encoder $M_{\text{embd}}$ that produces vectors in $\mathbb{R}^h$. In practice, we fine-tune an off-the-shelf encoder with a contrastive objective (Wieting et al., 2022) for paraphrastic robustness.

During the initialization of SEMSTAMP watermarked generation, we partition the space of sentence embeddings (produced by $M_{\text{embd}}$) with the LSH introduced in §2.1. Concretely, we initialize the $\text{LSH} : \mathbb{R}^h \mapsto \{0, 1\}^d$ function by sampling normal vectors $n^{(1)} \dots n^{(d)}$ to represent $d$ hyperplanes, and treat the space of LSH signatures $\{0, 1\}^d$ as a natural partitioning of $\mathbb{R}^h$ into $2^d$ regions.

At each generation step, given a sentence history $s^{(0)} \dots s^{(t-1)}$, we first produce the LSH signature of the previously generated sentence SIG$(s^{(t-1)})$, where SIG$(\cdot)$ encodes and LSH-hashes the sentence, as defined in Alg. 2. Next, we pseudorandomly divide the LSH partitions into a set of "valid" regions $G^{(t)}$ and a set of "blocked" regions $R^{(t)}$, where the masking is seeded by SIG$(s^{(t-1)})$.[4] To produce the watermarked next sentence, we sample with rejection a new sentence $s^{(t)}$ from the LM until its embedding lies in the "valid" region in the semantic space.[5]

To detect the SEMSTAMP watermark, we conduct a one-proportion $z$-test on the number of valid-region sentences in the generated text. Since this detection is similar to Kirchenbauer et al. (2023a), we defer the details to §B.

Because a proper paraphrase should retain the meaning of the original sentence, we hypothesize that the LSH signature is likely to remain the same after paraphrasing (Figure 4 provides empirical results). Therefore, the valid region partition for the next sentence would not change, ensuring the watermark is still detectable after the paraphrase attack. Below we explain each core component of

[4] Kirchenbauer et al. (2023a) use "green/red" for vocabulary split. Instead, we adopt "valid/blocked" as the terminology for semantic region partition to be more accessible.

[5] We set a maxout number $N_{\text{max}}$ so that if there is still no valid sentence after sampling $N_{\text{max}}$ times, we choose the last sample as the next sentence.

SemStamp in detail.

**Paraphrase-Robust Sentence Encoder** A requirement for SemStamp is a semantic encoder to map sentences into semantic embeddings. Our encoder is built upon Sentence-BERT (SBERT; Reimers and Gurevych, 2019), a fine-tuned siamese network trained to produce sentence embeddings whose cosine similarity mirror the semantic similarity of the STS benchmark (Cer et al., 2017).

To enhance the encoder's robustness to paraphrase, we further fine-tune the SBERT model using contrastive learning (Wieting et al., 2022). For each sentence $s_i$ in a corpus, we first produce its paraphrase $t_i$ using an off-the-shelf paraphrasing model, Pegasus (Zhang et al., 2020).[6] Next, we sample a random sentence $t'_i$ from the corpus that is not a paraphrase of $s_i$ to serve as the negative example. The objective promotes the original sentence to be more similar to the paraphrase than the negative example by a margin of $\delta > 0$:

$$\min_{\theta} \sum_{i} \max\Big\{\delta - f_\theta(s_i, t_i) + f_\theta(s_i, t'_i), 0\Big\}, \quad (1)$$

where $f_\theta$ is the cosine similarity between the embedded sentences, $f_\theta(s, t) = \cos\big(M_\theta(s), M_\theta(t)\big)$, and $M_\theta$ is the encoder model with parameter $\theta$.

**Semantic Space Partitioning through LSH** During the initialization of watermarked generation, normal vectors $n^{(1)} \ldots n^{(d)}$ are randomly drawn from the $h$-dimensional Gaussian distribution to represent $d$ LSH hyperplanes in the semantic space $\mathbb{R}^h$. The hyperplanes are fixed during generation and detection to serve as the basis for partitioning. As introduced in §2.1, this induces a $d$-bit binary signature $\textsc{Lsh}(v)$ for a vector $v \in \mathbb{R}^h$. Consequently, we use each of the $2^d$ signatures $c \in \{0, 1\}^d$ to represent a region in the semantic space consisting of points with signature $c$.

During the generation of a new sentence $s^{(t)}$, we apply a watermarking "mask" on the semantic space by pseudorandomly partitioning the space of signatures $\{0, 1\}^d$ into a valid region set $G^{(t)}$ of size $\gamma \cdot 2^d$ and a blocked region set $R^{(t)}$ of size $(1 - \gamma) \cdot 2^d$, where $\gamma \in (0, 1)$ determines the ratio of valid regions. The masking is seeded by the LSH signature of the last sentence $s^{(t-1)}$ and thus varies for each time-step $t$. Specifically, we convert the binary signature $\textsc{Sig}(s^{(t-1)})$ to decimal and use $[\textsc{Sig}(s^{(t-1)})]_{10} \times p$ to seed the randomization. Here $p$ is a large prime number and $[.]_{10}$ an operator that casts binary numbers to decimal numbers. The condition for rejection sampling is that the LSH signature of the new sentence must fall into one of the valid regions, i.e., $\textsc{Lsh}(M_{\text{embd}}(s^{(t)}) \in G^{(t)}$.

[6] Link to Pegasus paraphraser.

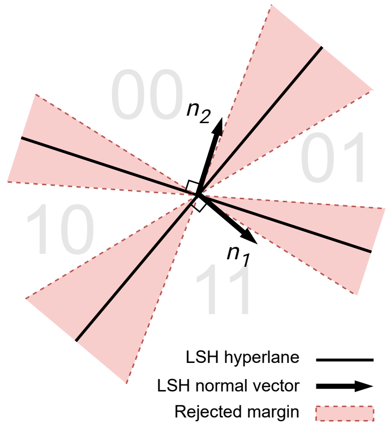


Figure 2: An illustration for margin-based rejection. Sentence embeddings at LSH hyperplane boundaries are rejected (highlighted in red).

**Margin-Based Constraint for Robustness** For the SemStamp algorithm to be robust, the LSH signature of the sentences should remain the same under paraphrase attack. Empirically, we found the robustness from contrastive learning (Eq. 1) is not strong enough to preserve consistent LSH signature under paraphrasing. Therefore, we add an additional rejection sampling requirement that the sampled sentence $s^{(t)}$ must have the absolute value of cosine similarity with each normal vector $n^{(i)}$ larger than a margin $m > 0$:

$$\min_{i=1,\ldots,d} |\cos(n^{(i)}, v_t)| > m, \quad (2)$$

where $v_t = M_{\text{embd}}(s^{(t)})$ is the embedding of the candidate next sentence.[7]

Visually, this is akin to rejecting sentences whose embeddings lie near the boundaries of an LSH hyperplane. We illustrate this in Figure 2. In our experiments (§3), we show that this margin-based rejection requirement can effectively increase the LSH signature robustness under paraphrasing.

## 2.3 The Bigram Paraphrase Attack

We develop a strong "bigram" paraphrase attack with the following intuition. Because existing token-level watermark algorithms hash the last generated token to determine the watermarking signature (Kirchenbauer et al., 2023a), any choice of token at position $t$ would affect the watermark of

[7] We discuss additional details on the condition for consistent LSH signature in §E.

position $t+1$. Therefore, we hypothesize that token-level watermarks might be especially sensitive to bigram (two adjacent tokens) perturbation.

Motivated by this intuition, we propose and explore the bigram paraphrase attack, a simple yet effective variant of the basic sentence-level paraphrase attack. Specifically, given a neural paraphrase model, we first decode a large number of top-raking sequences $s'_1 \dots s'_k$ with beam search, obtaining $k$ paraphrase candidates. Next, we select the candidate that has the smallest bigram overlap with the original sentence. Moreover, to preserve the paraphrasing quality, we constrain the paraphrase attack with BERTScore (Zhang et al., 2019) between paraphrases and original sentences:

$$s' = \underset{x \in \{s'_1, \dots, s'_k\}}{\arg\min} \quad \mathcal{B}(x, s),$$

$$\text{subject to} \quad \mathcal{S}(s'_1, s) - \mathcal{S}(x, s) \leq \Delta \cdot \mathcal{S}(s'_1, s),$$

where $s$ denotes the original sentence, $\mathcal{B}(x, s)$ is a simple counting of overlapped bigrams between sequences $x$ and $s$, $\mathcal{S}(x, s)$ denotes the BERTScore between sequence $x$ and $s$, and $\Delta$ is the BERTScore threshold ratio. See Figure 5 for an example in action.

## 3 Experiments

### 3.1 Experimental Setup

**Datasets** We conduct experiments to validate the detection robustness and quality of SEMSTAMP on the RealNews subset of the C4 dataset (Raffel et al., 2020) and on the BookSum (Kryściński et al., 2021). We further analyze the detection results and generation quality on 1000 random samples.

**Metrics** We use binary classification metrics: (1) area under the receiver operating characteristic curve (***AUC***), and (2) the true positive rate when the false positive rate is 1% or 5% (***TP@1%***, ***TP@5%***), i.e., the percentage of machine-generated text (the "positive" class in the classification setting) that is correctly detected when 1% and 5% of human texts (the "negative" class) are misclassified as machine-generated texts. A piece of text is classified as machine-generated when its $z$-score exceeds a threshold chosen based on a given false positive rate, which we explain in detail in §B. Differing from KGW algorithm (Kirchenbauer et al., 2023a), our algorithm treat sentences as the unit during $z$-score computation.

To evaluate generation quality, we measure the perplexity (***PPL***) with OPT-2.7B (Zhang et al., 2022). Generation diversity is measured with trigram text entropy (Zhang et al., 2018) (***Ent-3***), i.e., the entropy of the trigram frequency distribution of the generated text. We also evaluate generations with ***Sem-Ent*** (Han et al., 2022), an automatic metric for semantic diversity. Following the setup in Han et al. (2022), we use the last hidden states of OPT-2.7B models on generations as their semantic representation and perform $k$-means clustering. Sem-Ent is the entropy of semantic cluster assignments of test generations. We evaluate the quality of paraphrases using BERTScore (Zhang et al., 2019) between original generations and their paraphrases.

**Training, Generation, and Baselines** For contrastive learning of SBERT, we paraphrase 8k paragraphs of the RealNews dataset (Raffel et al., 2020) using the Pegasus paraphraser (Zhang et al., 2020) through beam search with 25 beams. We then finetune an SBERT model[8] with an embedding dimension $h = 768$ on this subset for 3 epochs with a learning rate of $4 \times 10^{-5}$, using contrastive learning objective (Eq. 1). We set the contrastive learning margin $\delta = 0.8$ which is tuned from the dev set.

For watermarked generation, we use OPT-1.3B (Zhang et al., 2022) as our base model and conduct sampling at a temperature of 0.7 following Kirchenbauer et al. (2023a) with a repetition penalty of 1.05. Setting 32 as the prompt length, we let 200 be our default generation length but also experiment on various different lengths (Fig. 3). To generate from SEMSTAMP, we sample at a LSH dimension $d = 3$ with valid region ratio $\gamma = 0.25$ and rejection margin $m = 0.02$. See §3.2 for the impact on hyperparameter choices.

We choose the popular watermarking algorithm Kirchenbauer et al. (KGW; 2023a) as our main baseline. In the paraphrase attack phase, we paraphrase generations by SEMSTAMP and KGW and compare their post-hoc detection rates after attacks. We also experiment with a distortion-free watermark by Kuditipudi et al. (KTH; 2023), but preliminary results show that KTH performs poorly compared to both KGW and SEMSTAMP against our paraphrase attacks for the AUC metric. We include the detection results with KTH in §D.

**Paraphrase Attack** For paraphrase attack experiments, watermarked generations are paraphrased sentence-by-sentence with the Pegasus paraphraser (Zhang et al., 2020), the Parrot paraphrase used in

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

| *Paraphraser* | *Algorithm* | RealNews *AUC* ↑ | RealNews *TP@1%* ↑ | RealNews *TP@5%* ↑ | BookSum *AUC* ↑ | BookSum *TP@1%* ↑ | BookSum *TP@5%* ↑ |
|---|---|---|---|---|---|---|---|
| No Paraphrase | KGW | 99.6 | 98.4 | 98.9 | 99.9 | 100.0 | 99.6 |
| | SSTAMP | 99.2 | 93.9 | 97.1 | 99.9 | 100.0 | 99.2 |
| Pegasus | KGW | 95.9 | 82.1 | 91.0 | 97.3 | 89.7 | 95.3 |
| | SSTAMP | **97.8** (+1.9) | **83.7** (+1.6) | **92.0** (+1.0) | **99.2** (+1.9) | **90.1** (+0.4) | **96.8** (+1.5) |
| Pegasus-bigram | KGW | 92.1 | 42.7 | 72.9 | 96.5 | 56.6 | 85.3 |
| | SSTAMP | **96.5** (+4.4) | **76.7** (+34.0) | **86.8** (+13.9) | **98.9** (+2.4) | **86.0** (+29.4) | **94.6** (+9.3) |
| Parrot | KGW | 88.5 | 31.5 | 55.4 | 94.6 | 42.0 | 75.8 |
| | SSTAMP | **93.3** (+4.8) | **56.2** (+24.7) | **75.5** (+20.1) | **97.5** (+2.9) | **70.3** (+28.3) | **88.5** (+12.7) |
| Parrot-bigram | KGW | 83.0 | 15.0 | 39.9 | 93.1 | 37.4 | 71.2 |
| | SSTAMP | **93.1** (+10.1) | **54.4** (+39.4) | **74.0** (+34.1) | **97.5** (+4.4) | **71.4** (+34.0) | **89.4** (+18.2) |
| GPT3.5 | KGW | 82.8 | 17.4 | 46.7 | 87.6 | 17.2 | 52.1 |
| | SSTAMP | **83.3** (+0.5) | **33.9** (+16.5) | **52.9** (+6.2) | **91.8** (+4.2) | **55.0** (+37.8) | **70.8** (+18.7) |
| GPT3.5-bigram | KGW | 75.1 | 5.9 | 26.3 | 77.1 | 4.4 | 27.1 |
| | SSTAMP | **82.2** (+7.1) | **31.3** (+25.4) | **48.7** (+22.4) | **90.5** (+13.4) | **47.4** (+43.0) | **63.6** (+36.5) |

Table 1: Detection results under different paraphraser settings. All numbers are in percentages. ↑ indicates higher values are preferred. The numbers in parenthesis show the changes over our baseline. **SEMSTAMP is more robust than KGW on multiple paraphrasers, datasets, and both the regular and bigram paraphrase attacks.**

| | *PPL*↓ | *Ent-3*↑ | *Sem-Ent*↑ |
|---|---|---|---|
| No watermark | 10.02 | 12.17 | 5.53 |
| KGW | 12.17 | 12.10 | 5.47 |
| SEMSTAMP | 10.20 | 12.16 | 5.51 |

Table 2: Quality evaluation results. ↑ and ↓ indicate the direction of preference (higher and lower). **SEMSTAMP preserves the quality of generated text**.

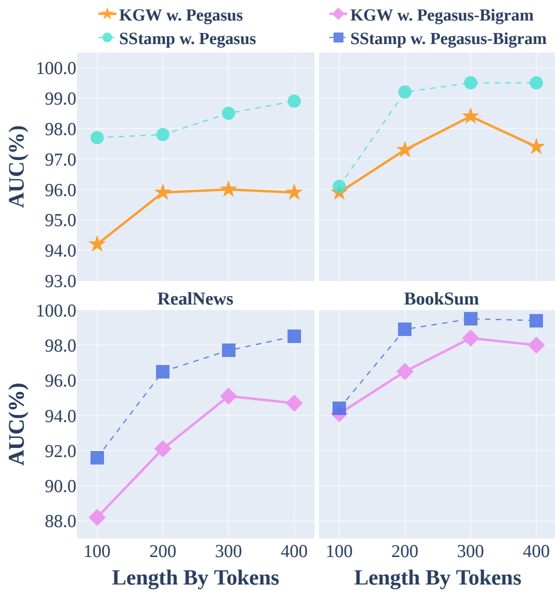


Figure 3: Detection results (AUC) under different generation lengths. **SEMSTAMP is more robust than KGW across length 100-400 tokens.**

Sadasivan et al. (2023), and GPT-3.5-Turbo (OpenAI, 2022). We use beam search with 25 beams for both Pegasus and Parrot. For GPT-3.5-Turbo, we provide the sentences before the current sentence as the context and prompt the model to paraphrase via the OpenAI API.[9] A detailed description of prompts is included in §E.

To implement the bigram paraphrase attack, we prompt the GPT-3.5-Turbo to return 10 paraphrases of the same sentence. For the Pegasus and Parrot paraphrasers, we select the candidate sentence with the least bigram overlap among the 25 beams from beam-search, subject to a BERTScore constraint of dropping no more than 10% of the score from the first beam. For GPT-3.5-Turbo, the paraphrase sample with the highest BERTScore is treated as the first beam.

## 3.2 Results

**Detection** Table 1 shows detection results under different paraphrasers and the bigram attack at generation length 200. **SEMSTAMP is more robust to paraphrase attacks than KGW across the Pegasus, Parrot, and GPT-3.5-Turbo paraphrasers, as measured by AUC, TP@1%, and TP@5%.** Although we only fine-tune the SBERT model on data from the Pegasus paraphraser, SEMSTAMP algorithm generalizes its robustness to different paraphrasers (Parrot, GPT-3.5-Turbo) and works on texts from different domains.

**The bigram paraphrase attack effectively weakens the token-level KGW algorithm while**

[9] https://platform.openai.com/playground/

**SemStamp is relatively unaffected.** Pegasus bigram attack can lower KGW's AUC by 7.9% and TP@5% by 27.1% on RealNews, whereas SemStamp only decreases by 3.5% and 13.2%. Furthermore, the BERTScore for bigram paraphrase does not change drastically compared to the regular paraphrases (Table 4 in §D), showing that the bigram paraphrase attack still preserves paraphrase quality due to the BERTScore constraints we add. Kirchenbauer et al. (2023b) propose several alternative hashing schemes to the KGW algorithm. We conduct paraphrase attack experiments on a recommended scheme named SelfHash, and do not find visible improvements to KGW, thus omitting the results for brevity.

**Quality** Table 2 compares quality metrics of non-watermarked generations with KGW and SemStamp generations. **While KGW notably degrades perplexity due to the token-level noise added to logits, the perplexity of SemStamp generation is on par with the base model without watermarking.** This confirms our hypothesis that the sentence-level nature of SemStamp is less disruptive of token selections and preserves the generation quality. Figure 5 and 6 provide qualitative examples of SemStamp generations and the bigram paraphrase attack. Compared to non-watermarked generation, the sentences are equally coherent and contextually sensible. **SemStamp also preserves token and semantic diversity of generation compared to non-watermarked generation and KGW generation**, as measured by the Ent-3 and Sem-Ent metrics, respectively.

**Generation Length** Figure 3 highlights that SemStamp is robust to both regular and bigram paraphrase attacks across different generation lengths as measured by the number of tokens. SemStamp has consistently higher AUC than KGW (Kirchenbauer et al., 2023a).

**Analysis** Figure 4 shows that increasing margin size $m$ will increase the consistency of LSH signatures (*LSH consistency*), the ratio of sentences that remain in the same valid region after being paraphrased. A higher rejection margin will ensure the sampled generations are further away from the region boundary, thus less likely to shift to a different region after paraphrasing. However, a larger margin will result in a slower generation speed, and we find $m = 0.02$ works well empirically.

We also compare the LSH consistency before and after fine-tuning SBERT with contrastive learning in Figure 4. Fine-tuning the encoder on Pegasus-paraphrased data improves the LSH consistency across different margins.

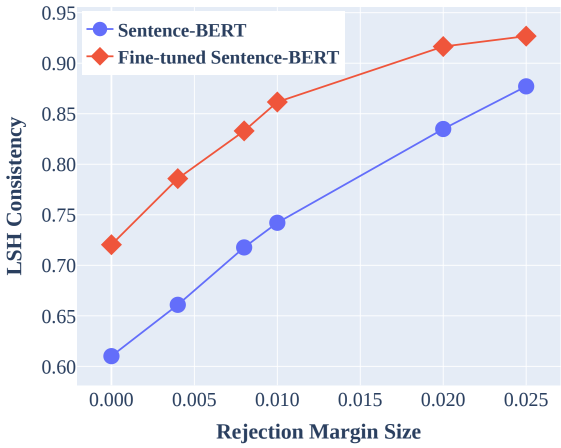


Figure 4: Rejection margin and contrastive fine-tuning effectively improve LSH Consistency.

Applying the masking of semantic space partitions and the rejection margin, SemStamp makes a trade-off between watermark detection accuracy and generation speed. For our current hyperparameter setting, 13.8 sentences are needed on average to sample one valid sentence. As we explain in the Limitations and Discussion section, this limitation can be mitigated if we conduct batched sampling of next sentences.

## 4 Related Work

Machine-generated text detection, aiming at distinguishing LLM-generated texts from human-written ones, can be categorized into proactive and post-hoc methods. Our focus, watermarked generation, belongs to the first category.

**Watermarked Generation** Early approaches to watermarking include text-meaning representation tree for information hiding (Atallah et al., 2002), and a watermarking scheme for machine translation using an output selector that considers hashing operation (Venugopal et al., 2011). Watermarked language generation, specifically pertinent to LLMs, is a renewed trend of proactive machine-generated text detection. The scheme works by adding signatures imperceptible to humans during decoding time to enable stable detection at a later time. Kirchenbauer et al. (2023a) propose a watermarking algorithm by adding token-level bias (reviewed in §2). Kuditipudi et al. (2023) proposes a distortion-free watermark that preserves the original distribution of LM during watermarking. Yoo et al. (2023) embeds multi-bit information into wa-

**Prompt:** Aylesbury's new deputy area commander has been getting stuck in to his new role and says he's determined to make himself 'approachable'.
**Non-Watermarked Generation:** Chief inspector Neil Kentish joined Thames Valley Police less than a week ago as deputy chief Constable. He's already got his work cut out as the force tries to tackle crime in the post Brexit age. Here he shares his top tips for doing your job well. 1. Look the part – be respectful, look presentable, dress appropriately and be on time. Keep an eye out for the telly chaps who may be trying to use you for your money – you should never let this happen.

**SEMSTAMP:** Chief inspector Neil Kentish joined Thames Valley Police less than a month ago and has been thrust into the hot seat of the Aylesbury District Commander team. He was chosen from a wide range of candidates put forward by Deputy Chief Constable Peter Borsack, who said, "I am delighted to have appointed Neil as deputy chief officer of police in Aylesbury District. He is a real character and I believe he will be a great addition to the team."
**Pegasus Paraphrase:** Neil Kentish was the chief inspector of Thames Valley Police. "I was thrust into the hot seat of the district commander team a month ago." He was chosen from a wide range of candidates put forward by Peter Borsack who said: "I am delighted to have appointed Neil as deputy chief officer of police in Aylesbury District. I think he will be a great addition to the team."
**Pegasus Bigram Paraphrase:** Neil Kentish was the chief inspector of Thames Valley Police. He was put into the hot seat of the district commander team a month ago. Neil was chosen from a wide range of candidates put forward by Peter Borsack, who said he was delighted to have appointed Neil as deputy chief officer of police. "I think he will be a good addition to the team. He will bring a good level of leadership and management skills to the community."

Figure 5: Generation Examples. Paraphrase examples are based on SEMSTAMP generations. Additional examples are presented in Figure 6 in the Appendix. **SEMSTAMP generations are equally coherent and contextually sensible compared to non-watermarked generations.**

termark and enhances performance against corruption through a robust infilling model. They inject the watermark via word replacement after initial generation, which is incorporated into one-stage watermarked generation by Wang et al. (2023). Christ et al. (2023) propose a watermarking scheme that is computationally undetectable without the secret key in theory.

Importantly, these existing works employ a token-level design and focus on span-level corruption such as editing and cropping, which renders the watermarks susceptible to paraphrase attacks.

More related to our focus on paraphrase attack, Krishna et al. (2023) propose a retrieval-based method that requires saving all previously-generated sequences, and Kirchenbauer et al. (2023b) empirically shows that Kirchenbauer et al. (2023a) is more robust under longer generation length. Contemporary to our work, Zhao et al. (2023) improves robustness via a cryptographic-free watermark without hashing previous tokens, which is more robust to editing and paraphrasing attacks. To the best of our knowledge, our work is the first sentence-level semantic watermark algorithm targeted against paraphrase attacks.

**Post-Hoc Detection of Machine-Generated Text** In post-hoc methods, applying binary classification models is the most straightforward approach (Zellers et al., 2019; Jawahar et al., 2020; Liu et al., 2022; Mireshghallah et al., 2023; Pu et al., 2023). These methods are applicable to black-box generators but need sufficiently large corpus for fine-tuning. Another type of post-hoc detection is based on statistical patterns within generation, including token likelihood (Gehrmann et al., 2019), rank (Solaiman et al., 2019), entropy (Ippolito et al., 2020), and likelihood gap at perturbation (Mitchell et al., 2023; Su et al., 2023). These methods have better interpretability but are reliable only with white-box access to generators. Sadasivan et al. (2023) question the theoretical reliability of detection while Chakraborty et al. (2023) support detection is achievable.

We defer further related works on LSH, watermarking for copyright, and contrastive learning to §A due to space reasons.

## 5 Conclusion

We introduce SEMSTAMP, a novel sentence-level semantic watermark for LLMs. The watermark is injected by mapping candidate sentences into embeddings with a paraphrase-robust encoder, partitioning the semantic space through LSH, and rejection sampling to generation sentences with valid region embeddings. Empirical results show that SEMSTAMP is not only robust to paraphrase attacks but also more quality-preserving than a token-level baseline watermark algorithm. We also propose a bigram paraphrase attack which effectively weakens the token-level watermark while only causing minor performance deterioration to SEMSTAMP. We hope SEMSTAMP can serve as an effective tool for regulating the proliferation of machine-generated texts.

## Limitations and Discussion

**Robustness to Stronger Attacks** Since SEMSTAMP operates on the sentence level, it is not robust against attacks on the inter-sentence level. For example, a recently proposed paraphraser Dipper (Krishna et al., 2023) includes sentence reordering. Our algorithm is also less effective when the machine text is embedded in a relatively large portion of human text. We leave the exploration of stronger attacks to future work.

**Semantic Constraint from LSH** While the LSH partitioning divides the full semantic space into sub-regions, enforcing the "valid region" requirement during generation may potentially reduce the generation flexibility. Interestingly, we use a small LSH dimension ($d = 3$) and we do not observe a visible quality degradation. A potential explanation is that with a smaller LSH dimension, the valid partition also becomes larger, which does not impose a strong semantic constraint and provides enough diversity for generations, as we found in our experiments (§3.2).

**Speed** Due to the nature of rejection sampling, text generation with SEMSTAMP is slower than non-watermarked generation by a factor of 13.8 with LSH dimension $d = 3$ and margin $m = 0.02$ (§3.2), and by a factor of 5.26 when $d = 3$ and $m = 0$ (Table 3). However, since candidate sentences for rejection sampling have the same LM context, it is possible to conduct batch sampling of candidate next sentences, which speeds up watermarked generation while increasing the memory overhead. We see the additional computation cost for SEMSTAMP as a cost for robustness: adding the watermark on the semantic space trades-off speed for better detection accuracy under paraphrase attacks. Further, a potential mitigation is through sampling candidate sentences with multiple devices at the same time.

**Reverse Engineering** Since our sentence encoder and LSH hyperplanes are not public, it is not straightforward for a curious attacker to reverse engineer the configurations and we leave it for future work to explore. The difficulty of reverse engineering can also be increased by using a larger LSH dimension, while the watermark could be less robust to paraphrase attack.

**Bigram Paraphrase Attack Control** We control the "intensity" degree of bigram paraphrase attack by constraining the paraphrase candidate selection with a BERTScore constraint. Removing the constraint will more forcefully lower AUROC at the expense of paraphrase quality.

## Ethical Impacts

As language models become increasingly capable of generating realistic texts, the risk of misusing language model generations, such as spreading misinformation, practicing plagiarism, and violating copyrights, has become imminent. Furthermore, on a fundamental level, the inability to distinguish humans from machines poses threats to establishing the basic level of mutual understanding and trust that bonds society. Robust detection of machine-generated text is crucial for preventing the misuse of large language models by properly attributing the source of online texts. Although current LLMs are often exposed to users as API endpoints, malicious users can still postprocess and paraphrase the API-generated response to escape the injected watermark. This motivates us to study watermark robustness against paraphrasing in this work. We hope that the proposed SEMSTAMP algorithm can mitigate the risk of LLM misuse by providing a reliable method to counter paraphrasing attacks on watermarked generations.

## References

Mikhail J. Atallah, Victor Raskin, Michael Crogan, Christian Hempelmann, Florian Kerschbaum, Dina Mohamed, and Sanket Naik. 2001. Natural language watermarking: Design, analysis, and a proof-of-concept implementation. In *Information Hiding*, pages 185–200, Berlin, Heidelberg. Springer Berlin Heidelberg.

Mikhail J. Atallah, Victor Raskin, Christian F. Hempelmann, Mercan Karahan, Radu Sion, Umut Topkara, and Katrina E. Triezenberg. 2002. Natural language watermarking and tamperproofing. In *International Workshop on Information Hiding*, pages 196–212.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Souradip Chakraborty, Amrit Singh Bedi, Sicheng Zhu, Bang An, Dinesh Manocha, and Furong Huang. 2023. On the possibilities of ai-generated text detection. *arXiv preprint arXiv:2304.04736*.

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.

Seungtaek Choi, Myeongho Jeong, Hojae Han, and Seung won Hwang. 2022. C2l: Causally contrastive learning for robust text classification. In *AAAI Conference on Artificial Intelligence*.

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

Hongchao Fang, Sicheng Wang, Meng Zhou, Jiayuan Ding, and Pengtao Xie. 2020. Cert: Contrastive self-supervised learning for language understanding. *ArXiv*, abs/2005.12766.

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.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush. 2019. Gltr: Statistical detection and visualization of generated text. *arXiv preprint arXiv:1906.04043*.

Chenxi Gu, Chengsong Huang, Xiaoqing Zheng, Kai-Wei Chang, and Cho-Jui Hsieh. 2022. Watermarking pre-trained language models with backdooring. *arXiv preprint arXiv:2210.07543*.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

R. Hadsell, S. Chopra, and Y. LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742.

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*.

The White House. 2023. FACT SHEET: Biden-Harris Administration Secures Voluntary Commitments from Leading Artificial Intelligence Companies to Manage the Risks Posed by AI.

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.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2020. Automatic detection of generated text is easiest when humans are fooled. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1808–1822, Online. Association for Computational Linguistics.

Daphne Ippolito, Florian Tramèr, Milad Nasr, Chiyuan Zhang, Matthew Jagielski, Katherine Lee, Christopher A Choquette-Choo, and Nicholas Carlini. 2022. Preventing verbatim memorization in language models gives a false sense of privacy. *arXiv preprint arXiv:2210.17546*.

Ganesh Jawahar, Muhammad Abdul-Mageed, and Laks Lakshmanan, V.S. 2020. Automatic detection of machine generated text: A critical survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2296–2309, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Taeuk Kim, Kang Min Yoo, and Sang goo Lee. 2021. Self-guided contrastive learning for bert sentence representations. *ArXiv*, abs/2106.07345.

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.

Tassilo Klein and Moin Nabi. 2020. Contrastive self-supervised learning for commonsense reasoning. *ArXiv*, abs/2005.00669.

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.

Shuang Li, Xuming Hu, Li Lin, and Lijie Wen. 2022. Pair-level supervised contrastive learning for natural language inference. *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8237–8241.

Xiaoming Liu, Zhaohan Zhang, Yichen Wang, Yu Lan, and Chao Shen. 2022. Coco: Coherence-enhanced machine-generated text detection under data limitation with contrastive learning. *ArXiv*, abs/2212.10341.

Yixin Liu, Hongsheng Hu, Xuyun Zhang, and Lichao Sun. 2023. Watermarking text data on large language models for dataset copyright protection. *ArXiv*, abs/2305.13257.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. *arXiv preprint arXiv:1803.02893*.

Fatemehsadat Mireshghallah, Justus Mattern, Sicun Gao, R. Shokri, and Taylor Berg-Kirkpatrick. 2023. Smaller language models are better black-box machine-generated text detectors. *ArXiv*, abs/2305.09859.

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*.

OpenAI. 2022. ChatGPT.

OpenAI. 2023. GPT-4 Technical Report.

Artidoro Pagnoni, Martin Graciarena, and Yulia Tsvetkov. 2022. Threat scenarios and best practices to detect neural fake news. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1233–1249, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Xiao Pu, Jingyu Zhang, Xiaochuang Han, Yulia Tsvetkov, and Tianxing He. 2023. On the zero-shot generalization of machine-generated text detectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4799–4808, Singapore. Association for Computational Linguistics.

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).

Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. 2005. Randomized algorithms and NLP: Using locality sensitive hash functions for high speed noun clustering. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 622–629, Ann Arbor, Michigan. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

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

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, and Jasmine Wang. 2019. Release strategies and the social impacts of language models. *ArXiv*, abs/1908.09203.

Jinyan Su, Terry Yue Zhuo, Di Wang, and Preslav Nakov. 2023. Detectllm: Leveraging log rank information for zero-shot detection of machine-generated text. *arXiv preprint arXiv:2306.05540*.

Benjamin Van Durme and Ashwin Lall. 2010. Online generation of locality sensitive hash signatures. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 231–235, Uppsala, Sweden. Association for Computational Linguistics.

Ashish Venugopal, Jakob Uszkoreit, David Talbot, Franz Och, and Juri Ganitkevitch. 2011. Watermarking the outputs of structured prediction with an application in statistical machine translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*.

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.

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, et al. 2021. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*.

Nathaniel Weir, João Sedoc, and Benjamin Van Durme. 2020. COD3S: Diverse generation with discrete semantic signatures. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5199–5211, Online. Association for Computational Linguistics.

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.

Hongwei Yao, Jian Lou, Kui Ren, and Zhan Qin. 2023. Promptcare: Prompt copyright protection by watermark injection and verification. *ArXiv*, abs/2308.02816.

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*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 9054–9065. Curran Associates, Inc.

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 Additional Related Works

**Locality-Sensitive Hashing in NLP** The application of locality-sensitive hashing (Indyk and Motwani, 1998; Charikar, 2002) in NLP dates back to Ravichandran et al. (2005), where LSH is used for high-speed noun clustering. Van Durme and Lall (2010) show that the LSH method of Charikar (2002) can enable fast approximated online computation of cosine similarity. Guu et al. (2018) use LSH to efficiently compute lexically similar sentences in a prototype-then-edit sentence generation model. Closely related to our work, Weir et al. (2020) generate semantically diverse sentences by conditioning a sequence-to-sequence model on the LSH signature of sentence embeddings.

**Watermarked Natural Language Data for Copyright** Watermarked generation can be further applied for data copyright protection. Gu et al. (2022) embed backdoor trigger words as black-box watermarks into LLMs. Liu et al. (2023) propose a novel watermark via backdoor-based membership inference, where backdoor watermarked texts poison unauthorized training models. Yao et al. (2023) focus on protecting the copyright of prompts through inserting the secret key into the prompt optimization stage. These works mainly apply watermark techniques for data copyright protections , whereas our work focuses on exploring the robustness of watermark against paraphrasing.

**Contrastive Learning in NLP** Contrastive learning (Hadsell et al., 2006) aims at improving the distinguishability of representation by pulling over positive pairs and pushing off negative pairs. In the NLP domain, contrastive learning can be applied to sentence embedding (Logeswaran and Lee, 2018), and further used in downstream tasks like natural language inference (Li et al., 2022), understanding (Fang et al., 2020), reasoning (Klein and Nabi, 2020), classification (Choi et al., 2022) etc. Logeswaran and Lee (2018) apply unsupervised contrastive learning between current sentence candidates and context sentences to effectively learn sentence representation. Gao et al. (2021) further apply supervised contrastive learning in sentence embedding by using annotated pairs from natural language inference. Kim et al. (2021) propose a self-guided contrastive learning between embeddings from a fixed model and a fine-tuned model.

## B Watermark Detection

Kirchenbauer et al. (2023a) proposes using a one-proportion $z$-test on the number of green list tokens to detect watermarks, assuming the following null hypothesis:

$H_0$ : *The text is not generated (or written) knowing a watermarking green list rule.*

The null hypothesis is rejected when the $z$-score computed based on the number of green tokens in a piece of text $T$ exceeds a given threshold $M$:

$$z_{\text{KGW}} = \frac{N_G - \gamma N_T}{\sqrt{\gamma(1-\gamma)N_T}}, \tag{3}$$

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 $\gamma$ is a chosen ratio of green tokens. During detection time, the number of green tokens in each piece of text will be counted. According to Eq. 3, a higher ratio of detected green tokens means a higher $z$-score, determining with more confidence that the text is machine-generated.

We adapt this one proportion $z$-test to SEMSTAMP, modifying the null hypothesis and using sentence as our basic unit:

$H_0$ :
*The text is not generated (or written) knowing a rule of valid and blocked partitions in the semantic space.*

$$z_{\text{SEMSTAMP}} = \frac{S_V - \gamma S_T}{\sqrt{\gamma(1-\gamma)S_T}}, \tag{4}$$

where $S_V$ refers to the number of valid *sentences*, $\gamma$ is the ratio of valid sentences out of the total number of sentences $S_T$ in a piece of text $T$.

During detection time, we first break a piece of texts into individual sentences and detect the number of valid sentences $S_V$ to calculate $z_{\text{SEMSTAMP}}$. We detect a machine-generated text when $z_{\text{SEMSTAMP}} > M_r$, where $M_r$ is located according to a given false positive rate $r$: We define machine-generated as the positive class in classical classification setting and non-machine-generated as the negative class. We iterate through a range of possible $m \in [0, 4.0]$ until there is a $M_r = m$ such that $r$ percentage of human (negative-class) texts is misclassified as machine-generated. For example, we let $r = 0.05$ for the ***TP@5%*** metric in Table 1.

**Prompt:** NEW DELHI: Over fifteen years after the Concorde — the world's first and only supersonic aircraft to be used commercially — retired, US aerospace major Boeing has announced it is working on its successor.
**Non-Watermarked Generation:** The first commercial supersonic jet was developed in the late 1960s by British Aerospace. However, the technology was never adopted by the industry. Boeing aims to change that. It has set up a team of engineers who will work on developing the next Supersonic Jet.
**Baseline Watermark:** The company has called it the X-35. But it's not yet clear when it might begin taking orders for production. The project is named after Jason Xtra, who first proposed it in 1997 – after the US Air Force expressed interest in it. However, Boeing hasn't announced any orders for the X-35 yet.

**SEMSTAMP:** The company said it was committed to developing the space elevator and had launched a concept for a space elevator in 2003. Boeing's chief financial officer, Robert Lach Jr, said the company would spend about $2 billion over the next five years on what is called the Space Elevator Initiative. Boeing estimated that an elevator would cost between $8 billion and $10 billion to build, depending on the design.

**Pegasus Paraphrase:** The company launched a concept for a space elevator in 2003 and said it was committed to developing the space elevator. Boeing will spend $2 billion over the next five years on the Space Elevator Initiative, according to Robert Lach Jr., the company's chief financial officer. Depending on the design, an elevator could cost between $8 billion and $10 billion.
**Pegasus Bigram Paraphrase:** In 2003 the company launched a concept for a space elevator. The company will spend $2 billion over the next five years on the Space Elevator Initiative. Depending on the design, an elevator could cost as much as $10 billion.

Figure 6: Additional Generation Examples. Non-Watermarked refers to the original model without adding any watermark. Baseline Watermark refers to (Kirchenbauer et al., 2023a). Paraphrase examples are based on SEMSTAMP generations.

| LSH Dim ($d$) | Average # of Sentences Sampled ↓ | LSH Consistency ↑ |
|---|---|---|
| 3 | 5.26 | **.720** |
| 4 | 4.53 | .666 |
| 8 | 4.26 | .508 |
| 16 | **4.14** | .335 |

Table 3: Effects of Increasing LSH Dimensions at margin $m = 0.0$. The sampling rate is the average number of sentences sampled to produce one valid (watermarked) sentence.

# C Effect of LSH dimension $d$

In Table 3, we discover that fewer LSH dimensions will make a sentence more likely to stay in the same region after being paraphrased. We define LSH Consistency as the ratio of paraphrased sentences that have the same LSH signature as the original sentence over the total number of paraphrased sentences. A higher consistency ratio indicates better robustness.

Geometrically, when the LSH dimension is lower, there are fewer partitioned semantic regions, each having a larger space. A paraphrase will have a similar representation with its source sentence in the semantic space, which will be more likely to remain in the same semantic region if each region is larger.

On the other hand, lowering the number of LSH dimensions will also slightly increase the average number of sentences sampled to produce one valid sentence (Average Number of Sentences Sampled). We ultimately decide on a minor sacrifice in speed for the gain of accuracy and choose $d = 3$. We choose $\gamma = 0.25$ following Kirchenbauer et al. (2023a), where the authors show that larger green-list ratios will lower the $z$-score.

# D Additional Experimental Results

We include additional experimental results on paraphrase quality, i.e., the BERTScore between original and paraphrased generations under different settings, in Table 4.

We provide paraphrased detection results of the KTH algorithm Kuditipudi et al. (2023) in Table 5. We find that the KTH watermark performs poorly against KGW and SEMSTAMP.

**Computing Infrastruture and Budget** We run sampling and paraphrase attack jobs on 8 A40 GPUs, taking up a total of around 100 GPU hours.

# E Additional Details

**Condition for consistent LSH signature** For robustness, the SEMSTAMP algorithm would need the LSH signature of the paraphrased sentence to be unchanged from the signature of the original sentence. This requires that for each LSH digit $i$, the sign of the dot product between the embedded sentence and the normal vector $n^{(i)}$ should not change before and after paraphrasing:

$$\mathbb{1}\left(n^{(i)} \cdot v_{\text{orig}} > 0\right) = \mathbb{1}\left(n^{(i)} \cdot v_{\text{para}} > 0\right), \quad \forall i \in \{1 \dots d\}, \tag{5}$$

| *Algorithm*↓ *Paraphraser*→ | RealNews Pegasus | RealNews Parrot | RealNews GPT3.5 | BookSum Pegasus | BookSum Parrot | BookSum 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 | 72.7 / 70.2 | 62.9 / 62.4 | 61.8 / 58.4 |

Table 4: BERTScore between original and paraphrased generations under different settings. All numbers are in percentages. The first number in each entry is under vanilla paraphrase attack while the second number is under the bigram paraphrase attack. **Bigram paraphrase attack poses only minor degradation on semantic similarity with original sentence compared to vanilla paraphrase attack.**

| *Algorithm* | BookSum *AUC* ↑ | BookSum *TP@1%* ↑ | BookSum *TP@5%* ↑ |
|---|---|---|---|
| KGW | 95.9 | 82.1 | 91.0 |
| KTH | 51.7 | 5.0 | 5.8 |
| SEMSTAMP | **97.8** | **83.7** | **92.0** |

Table 5: Paraphrased detection results on the BookSum dataset. The paraphraser used is Pegasus. We find that the KTH watermark performs poorly against KGW and SEMSTAMP.

where $v_{\text{orig}} = M_{\text{embd}}(s^{(t)})$ and $v_{\text{para}} = M_{\text{embd}}(G(s^{(t)}))$ are the embeddings for the original and paraphrased sentences, respectively, and $G$ is the paraphraser.

**Cosine Similarity** In §2.2, we slightly abuse the notation and use $\cos(\boldsymbol{x}, \boldsymbol{y})$ to denote the *cosine similarity* between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. That is,

$$\cos(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{|\boldsymbol{x}||\boldsymbol{y}|}. \tag{6}$$

**Sentence Delimitation** During generation time, a full candidate next sentence is considered generated if the language model has generated a new delimiter punctuation, i.e., a comma, period, question mark, or exclamation mark.

**Data Preprocessing** We separate the data points, which are paragraphs of news (RealNews) and book summaries (BookSum), into sentences using `nltk.sent_tokenize`. Additionally, we add a period mark to every sentence that does not end in a comma, period, question mark, or exclamation mark.

**Prompt for GPT-3.5-Turbo Paraphrase** To use GPT-3.5-Turbo as a paraphraser, we provide the following prompt:

```
Previous context: {context} \n
Current sentence to paraphrase: {sent}
```

We define `sent` to be the target sentence to be paraphrased, and `context` as the list of sentences before the target sentence.

For the bigram paraphrase attack, we provide the following prompt:

```
Previous context: {context} \n
Paraphrase in {num-beams} different ways
and return a numbered list : {sent}
```

where `num-beams` specifies the number of candidate sentences. A higher `num-beams` will strengthen the bigram paraphrase attack but also at the cost of more computational resources.