SimMark: A Robust Sentence-Level Similarity-Based Watermarking Algorithm for Large Language Models

Anonymous ACL submission

Abstract

The widespread adoption of large language models (LLMs) necessitates reliable methods to detect LLM-generated text. We introduce ***SimMark***, a robust sentence-level watermarking algorithm that makes LLM's outputs traceable without requiring access to model internals, making it compatible with both open and API-based LLMs. By leveraging the similarity of semantic sentence embeddings combined with rejection sampling to embed detectable statistical patterns imperceptible to humans, and employing a *soft* counting mechanism, *SimMark* achieves robustness against paraphrasing attacks. Experimental results demonstrate that *SimMark* sets a new benchmark for robust watermarking of LLM-generated content, surpassing prior sentence-level watermarking techniques in robustness, sampling efficiency, and applicability across diverse domains, all while maintaining the text quality and fluency.

1 Introduction

The advent of deep generative models has made it increasingly important to determine whether a given text, image, or video was produced by artificial intelligence (AI), and recently, researchers across various domains have begun tackling this challenge (Aaronson and Kirchner, 2022; Fernandez et al., 2023; Teymoorianfard et al., 2025). In particular, LLMs such as GPT-4 (OpenAI et al., 2023) can now generate human-like text at scale and low cost, enabling powerful applications across numerous industries. However, this capability also introduces serious risks, including academic plagiarism, disinformation campaigns, and the manipulation of public opinion. For instance, the use of AI-generated content in news articles has raised concerns about transparency, accountability, and the spread of misinformation (Futurism, 2023). Moreover, reliably detecting LLM-generated content is crucial for enforcing copyright protections and ensuring accountability (Weidinger et al., 2021).

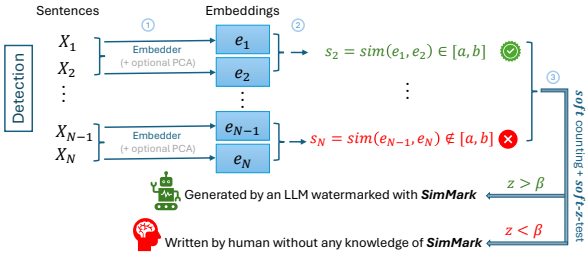

Figure 1: A high-level overview of *SimMark* detection algorithm. The input text is divided into individual sentences X_1 to X_N, which are embedded using a semantic embedding model. The similarity between consecutive sentence embeddings is computed. Sentences with similarities within a predefined interval $[a, b]$ are considered **valid**, while those outside are **invalid**. A statistical test is performed using the count of **valid** sentences to determine whether the text is watermarked.

Detecting LLM-generated text poses a unique challenge. These models are explicitly trained to emulate human writing styles, often rendering their outputs indistinguishable from human-authored text. As demonstrated by Kumarage et al. (2023) and Sadasivan et al. (2023), reliably differentiating between human-written and machine-generated text remains an open problem.

One promising approach is the use of imperceptible statistical signatures, or *watermarks*, embedded within a text. Watermarking imperceptibly alters text such that it remains natural to human readers but enables subsequent detection of its origin (Atallah et al., 2001). Effective watermarking must balance the preservation of the text quality with robustness against adversarial *paraphrasing*, where attackers modify the text to evade detection (Krishna et al., 2024). Additionally, watermarks must be resistant to *spoofing* attacks, wherein adversaries craft non-machine-generated text (often malicious) to falsely trigger detectors (Sadasivan et al., 2023).

In this paper, we introduce ***SimMark***, a robust sentence-level watermarking algorithm for LLMs based on sentence embedding similarity. *SimMark*

treats LLMs as black boxes that can be prompted to generate sentences given a context. This approach makes *SimMark* compatible with a wide range of models, including open-weight LLMs and closed-source proprietary models accessible only via APIs, as it does not require fine-tuning or access to the models' internal logits. Access to logits is often restricted by API providers due to their potential use in distilling LLMs and leaking proprietary information (Finlayson et al., 2024). *SimMark* leverages embeddings from semantic text embedding models to capture semantic relationships between sentences and embeds detectable statistical patterns on sentence similarity through rejection sampling. Specifically, rejection sampling involves querying the LLM multiple times until the similarity between the embeddings of consecutive sentences falls within a predefined interval. During detection, these patterns are analyzed using a statistical test to differentiate between human-written and LLM-generated text, as illustrated in Figure 1.

In summary, our contributions are as follows:

- We introduce a novel sentence-level watermarking algorithm that achieves state-of-the-art detection performance while maintaining low false positive rates for human-written text.
- Our approach demonstrates robustness against paraphrasing attacks through semantic-level watermarking and a soft counting mechanism for statistical testing.
- Compared to existing methods, *SimMark* provides a more practical solution that operates without access to LLM logits, offering high-quality watermark injection and detection.

The remainder of this paper is organized as follows. Section 2 reviews the background and related work on LLM watermarking techniques. Section 3 outlines our methodology. Section 4 describes our experimental setup and presents comparative results, while Section 5 concludes the paper.

2 Background

2.1 Autoregressive Decoding of LLMs

An LLM operates over a vocabulary V, a set of words or subwords termed as *tokens*. Let $f : V \to V$ be an LLM that takes a sequence of tokens $T_i = \{t_1, t_2, \ldots, t_i\}$ as input and generates the next token t_{i+1} as its output. To generate t_{i+1}, the LLM samples it from the conditional probability distribution $P(t_{i+1}|T_i)$ over the vocabulary V. After generating t_{i+1}, the updated sequence $T_{i+1} = T_i \cup \{t_{i+1}\}$ is fed back into the model, and the process is repeated iteratively to generate the subsequent tokens. This process of generating one token at a time, given the previously generated tokens, is known as *autoregressive decoding*.

2.2 Token-Level Watermarking

Token-level watermarking methods embed a statistical signal in the text by manipulating the token sampling process (Aaronson and Kirchner, 2022; Kirchenbauer et al., 2023; Fu et al., 2024). These methods typically alter the probability distribution over V, subtly biasing the selection of certain tokens to form detectable patterns.

KGW introduced by Kirchenbauer et al. (2023), groups V into *green* and *red* subsets *pseudo-randomly* seeded on the previous token before generating each new token. A predefined constant $\delta > 0$ is added to the logits of each token in the green list, increasing their likelihood of being selected during the sampling step. At detection, a z-test is applied to the number of tokens from the green list in the text to determine whether the text contains a watermark. This test compares the observed proportion of green tokens to the expected proportion under the null hypothesis of no watermark, providing a statistical measure to detect even subtle biases introduced by the watermark.

Detection of such watermarks involves analyzing tokens for statistical signatures that deviate from typical human text. However, token-level watermarks can still be vulnerable to paraphrasing, as rephrasing may disrupt the green and red token lists without altering the overall semantic (Krishna et al., 2024). Moreover, since these methods modify the logits, they directly impact the conditional probability distribution over V, potentially degrading the quality of the generated text (Fu et al., 2024).

Due to space constraints, additional related work—including token-level methods such as UNIGRAM-WATERMARK (UW) (Zhao et al., 2023) and the Semantic Invariant Robust (SIR) watermark (Liu et al., 2023), as well as post-hoc watermarking techniques—is deferred to Appendix A.

2.3 Sentence-Level Watermarking

One approach to mitigate the previously mentioned problems is to inject the watermark signal at the sentence level, making it less vulnerable to adversarial modifications (Topkara et al., 2006). Consider a similar notation for sentence generation using an autoregressive LLM that takes a sequence of

sentences $M_i = \{X_1, X_2, \ldots, X_i\}$ and generates the next sentence X_{i+1}. The updated sequence of sentences $M_{i+1} = M_i \cup \{X_{i+1}\}$ is then used to generate subsequent sentences iteratively.

SemStamp by Hou et al. (2024a) employs Locality-Sensitive Hashing (LSH) (Indyk and Motwani, 1998) to pseudo-randomly partition the semantic space of a embedding model into a set of *valid* and *blocked* regions, analogous to the green and red subsets in KGW. During rejection sampling, if the embedding of a newly generated sentence lies within the valid regions (determined based on the LSH signature of the previous sentence), the sentence is accepted. Otherwise, a new sentence is generated until a valid sentence is produced or the retry limit is reached. Similar to KGW, a z-test is applied to the number of valid sentences to determine whether the text contains a watermark.

To improve robustness against paraphrasing, SemStamp used a contrastive learning approach (Hadsell et al., 2006), fine-tuning an embedding model such that the embeddings of paraphrased sentences remain as close as possible to the original sentences. This was achieved by minimizing the distance between paraphrased and original embeddings while ensuring unrelated sentences remained distinct. They also introduce a margin constraint in the rejection sampling process to reject sentences whose embeddings lie near the region boundaries.

k-SemStamp (Hou et al., 2024b) builds upon SemStamp and aims to enhance robustness by partitioning the semantic space using k-means clustering (Lloyd, 1982) instead of random partitioning. They claim that in this way, sentences with similar semantics are more likely to fall within the same partition, unlike random partitioning, which may place semantically similar sentences into different partitions, reducing robustness; however, k-SemStamp assumes that the LLM generates text within a specific domain to apply k-means clustering effectively (Hou et al., 2024b), limiting its applicability in real-world, open-domain scenarios. The generation and detection procedures of k-SemStamp remain similar to the original SemStamp. In contrast to token-level algorithms, these sentence-level methods do not alter the internals of the LLM, therefore it is expected that their output to be of higher quality (Hou et al., 2024a,b).

Our work, similar to SemStamp and k-SemStamp, is a sentence-level algorithm; however, it injects its watermark signature into the semantic similarity of consecutive sentences. It achieves great generalizability across domains by leveraging an off-the-shelf, general-purpose embedding model without fine-tuning. At the same time, it outperforms these state-of-the-art (SOTA) sentence-level watermarking methods in robustness against paraphrasing while preserving text quality.

Algorithm 1 *SimMark* Generation Pseudo-Code

for each generated sentence X_i **do**
  Compute embedding e_i for X_i using the embedding model.
  $n \leftarrow 0$
  do
    Generate sentence X_{i+1} using the LLM.
    Compute embedding e_{i+1} for X_{i+1} using the embedding model.
    Optional: Reduce the dimension of e_i and e_{i+1} using the PCA model.
    Compute similarity s_{i+1}:

$$s_{i+1} \leftarrow \begin{cases} \frac{e_i \cdot e_{i+1}}{\|e_i\|_2 \cdot \|e_{i+1}\|_2} & \text{if cosine similarity,} \\ \|e_i - e_{i+1}\|_2 & \text{if Euclidean distance,} \end{cases}$$

    $n \leftarrow n + 1$
  while $s_{i+1} \notin [a, b]$ **and** $n < N_{\max}$
  Accept X_{i+1} as valid and continue generating the next sentence.
end for

3 *SimMark*: A Similarity-Based Watermarking Algorithm

3.1 Watermarked Text Generation

Similar to SemStamp and k-SemStamp, *SimMark* utilizes the embedding representations of the sentences. To compute the embeddings, in contrast with Hou et al. (2024a,b) that fine-tuned their embedder model (which could make it biased toward a specific paraphrasing model or domain), we employ Instructor-Large (Su et al., 2023), a general-purpose embedding model. The flexibility of our method in using any pretrained embedding model enables our approach to be more easily adaptable to different domains.

First, we compute the embedding for each sentence[1]. Then, we calculate the cosine similarity (or Euclidean distance) between the embedding of sentence $i + 1$ and the embedding of sentence i. If the

[1]In our experiments, we passed both the sentence and *"Represent the sentence for cosine similarity:"* or *"Represent the sentence for Euclidean distance:"* as the instruction to the Instructor-Large model.

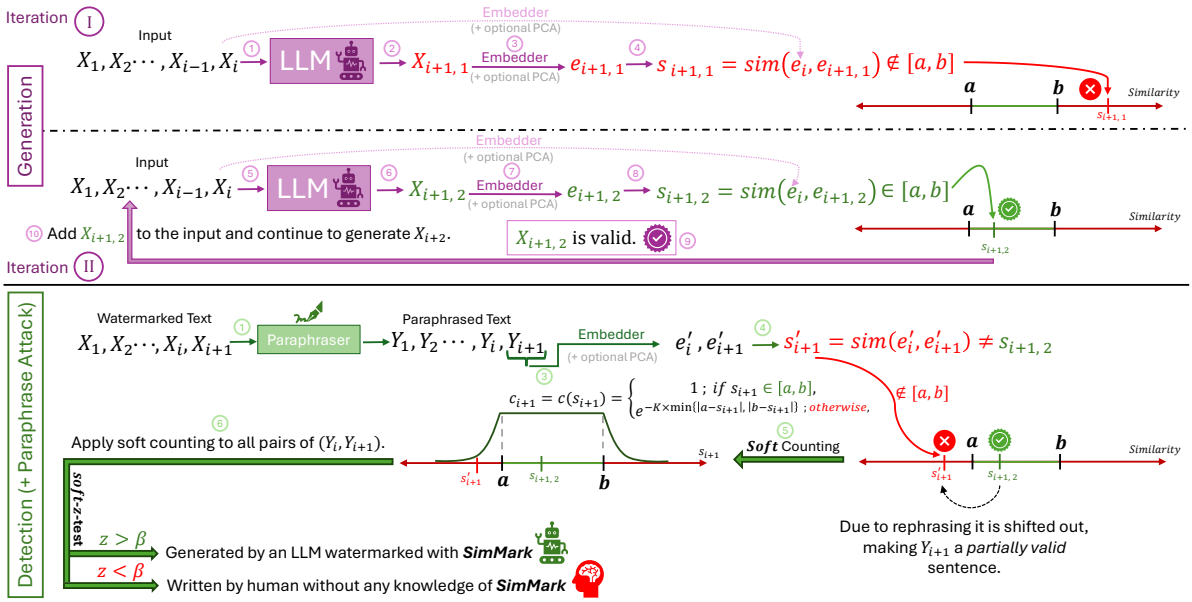

Figure 2: Overview of ***SimMark***. **Top: *Generation***. For each newly generated sentence (X_{i+1}), its embedding (e_{i+1}) is computed using a semantic text embedding model, optionally applying PCA for dimensionality reduction. The cosine similarity (or Euclidean distance) between e_{i+1} and the embedding of the previous sentence (e_i), denoted as s_{i+1}, is calculated. If s_{i+1} lies within the predefined interval $[a, b]$, the sentence is marked valid and accepted. Otherwise, rejection sampling generates a new candidate sentence until validity is achieved or the iteration limit is reached. Once a sentence is accepted, the process repeats for subsequent sentences. **Bottom: *Detection (+ Paraphrase attack)***. Paraphrased versions of watermarked sentences are generated (Y_i), and their embeddings (e'_i) are computed. The similarity between consecutive sentences in the paraphrased text is evaluated. If paraphrasing causes the similarity (s'_{i+1}) to fall outside $[a, b]$, it is mismarked as invalid. A *soft counting* mechanism (via function $c(s_{i+1})$ instead of a regular counting with a step function in $[a, b]$) quantifies partial validity based on proximity to the interval bounds, enabling detection of watermarked text via a ***soft***-z-test even under paraphrase attacks. It should be noted that soft counting is always applied during detection as we cannot assume prior knowledge of paraphrasing.

computed value lies within a predefined interval, sentence $i + 1$ is considered ***valid*** (analogous to the green subset in KGW). Otherwise, we prompt the LLM to generate a new sentence and repeat this procedure until a valid sentence is found or the maximum number of iterations is reached (in this case, we accept the last generated sentence), as shown in Algorithm 1. Optionally, we may apply Principal Component Analysis (PCA) method to the embeddings to reduce their dimensionality before calculating the similarity[2]. The reason for applying PCA is provided in Subsection 3.2. An overview of *SimMark* generation algorithm is depicted in the top part of Figure 2.

The predefined interval is a hyperparameter chosen a priori based on the distribution of similarities between consecutive sentences' embeddings generated by an unwatermarked LLM and human-written text. The choice of this hyperparameter is critical for the performance of *SimMark*. First, if the interval's width is too small or its position is far from the mean of the similarity distribution, generating sentences within this interval can be challenging or even infeasible for the LLM. Conversely, if the interval's width is large and centered around the mean of the similarity distribution, generating sentences becomes easier, but the false positive (FP) rate (i.e., human-written text misclassified as machine-generated) increases.

Furthermore, the choice of interval affects the robustness of *SimMark* against paraphrasing attacks. When an attacker paraphrases sentences, the similarities may change and fall outside the interval. Consequently, a larger interval provides greater robustness, as the watermark is less likely to be disrupted by paraphrasing. Therefore, the selection of the interval involves balancing several factors: it must not be too narrow to impede sentence generation while maintaining a low FP rate and adequate robustness against paraphrasing.

Finding a "sweet spot," for the interval depends on the distribution of similarities between consecutive sentences, which can vary across models. However, identifying such sweet spots is feasible when we analyze the similarity distributions of both human-authored and LLM-generated text (see Ap-

[2]In our experiments, this is the instruction in this case: *"Represent the sentence for PCA:"*

Algorithm 2 *SimMark* Detection Pseudo-Code

Split the input text into sentences excluding the first sentence (i.e, the prompt): $M = \{X_2, \ldots, X_{N_{\text{total}}}\}$, and set $N \leftarrow |M|$.
for each sentence pair (X_i, X_{i+1}) **do**
  Compute embeddings e_i for X_i and e_{i+1} for X_{i+1} using the embedding model.
  Optional: Reduce the dimensionality of e_i and e_{i+1} using the PCA model.
  Compute $s_{i+1} \leftarrow sim(e_i, e_{i+1})$.
  Compute c_{i+1} according to Eq. (1).
end for
Soft count of valid sentences: $N_{\text{valid\_soft}} \leftarrow \sum_{i=2}^{N+1} c_i$.
Estimate p_0 as the area under the human-written text embeddings similarity distribution curve within $[a, b]$.
Compute z_{soft} using Eq. (2).
if $z_{\text{soft}} > \beta$ **then**
  Reject H_0, i.e., text is likely generated by an LLM watermarked by *SimMark*.
else
  Accept H_0, i.e., text is likely human-written.
end if

pendix H for an example of finding a sweet spot).

3.2 Watermarked Text Detection

The detection of *SimMark* follows a similar methodology to KGW by employing a z-test for hypothesis testing. However, akin to Hou et al. (2024a), the detection operates at the *sentence level* rather than the token level. To perform detection, we first divide the input text into sentences[3] and use the same semantic embedding model to compute the embeddings for each sentence. If PCA was applied during the watermarking process, it must also be applied during detection to ensure consistency.

Next, we compute the similarity ($s_{i+1} = sim(e_i, e_{i+1})$) between consecutive sentences (X_{i+1} and X_i) and count the number of valid sentences ($N_{\text{valid\_soft}}$). A sentence X_{i+1} is deemed valid if s_{i+1} lies within the predefined interval $[a, b]$. However, paraphrasing may alter embeddings significantly, causing the similarity to deviate from the desired interval. To mitigate this, we adopt a **soft counting** approach, where a sentence is considered *partially valid* if its similarity is near the interval boundaries. Specifically, the soft count of X_{i+1}, denoted as c_{i+1}, is defined as follows:

$$c_{i+1} = c(s_{i+1}) = \begin{cases} 1 & \text{if } s_{i+1} \in [a, b], \\ e^{-K \min\{|a - s_{i+1}|, |b - s_{i+1}|\}} & \text{otherwise.} \end{cases} \quad (1)$$

Here, $K > 0$ is a decay factor controlling the smoothness of the soft counting. A higher K makes the function behave closer to a step function, while a lower K allows for smoother transitions, tolerating minor deviations outside the interval $[a, b]$. The total number of valid sentences is then computed as $N_{\text{valid\_soft}} = \sum_i c_i$. Refer to Appendix F for an ablation study on how this approach improves robustness against paraphrasing, with only a minimal impact on performance in non-paraphrased scenarios, by allowing for some degree of error.

During our initial experiments, we observed that, contrary to cosine similarity, Euclidean distance is very sensitive to paraphrasing and even a subtle change in the sentences would result in a huge difference in the distances of embeddings. We hypothesize that Euclidean distance is sensitive to noise in high-dimensional spaces such as the semantic space of an embedder. To mitigate this, we propose using PCA: We fit a PCA model on a dataset of human-written texts, to find the principal components of the sentence embeddings, i.e., the components that contribute the most to the semantical representation of the sentences. Then, we apply PCA to reduce embeddings' dimension. More details on the dimensionality reduction are provided in Section 4 and Appendix G.

This approach can make reverse-engineering more difficult, as it would require knowledge of not only the embedder model, but also the PCA setting (e.g., number of components, access to the dataset used for fitting it, etc.). Without all these details, reproducing the similarity distribution becomes less straightforward.

The null hypothesis H_0 is defined as follows:

> H_0: *The sentences are written by humans, i.e., the text sequence is generated without knowledge of the valid interval in the similarity of sentence embeddings.*

We calculate the z-statistic for the one-proportion z-test using the sample proportion $p = \frac{N_{\text{valid\_soft}}}{N}$, where N is the total number of samples (sentences). Since $N_{\text{valid\_soft}}$ is a soft count of valid sentences, we refer to it as ***soft***-z-score, which is given by $z_{\text{soft}} = \frac{p - p_0}{\sqrt{\frac{p_0(1-p_0)}{N}}}$ or alternatively:

$$z_{\text{soft}} = \frac{N_{\text{valid\_soft}} - p_0 N}{\sqrt{p_0(1-p_0)N}}. \quad (2)$$

[3] Using *sent_tokenize* method of NLTK (Bird et al., 2009).

Here, the population proportion p_0 represents the ratio of valid sentences to all sentences in human-written text (i.e., a text with no watermark), which is estimated as the area under the similarity distribution curve of consecutive human-written sentences within the interval $[a, b]$ (Like the one in Figure 7 in Appendix H). The value of z_{soft} can be interpreted as a normalized deviation of the number of valid sentences $N_{\text{valid\_soft}}$ from its expectation $p_0 N$.

As highlighted in Algorithm 2, the null hypothesis H_0 is rejected if $z_{soft} > \beta$, where β is a threshold determined empirically by running the detection algorithm on human-written text. The threshold β is selected to maintain a desired FP rate (i.e., minimizing the misclassification of human-written text as LLM-generated). Details on the computation of β are provided in Appendix I.

4 Experiments & Results

Performance of *SimMark* is evaluated across different datasets and models using the area under the receiver operating characteristic curve (ROC-AUC) and true positive rate (TP) at fixed FP rates of 1% and 5% (TP@1%FP and TP@5%FP). Higher values indicate better performance across all metrics[4].

For dimensionality reduction, we fitted a PCA model on 8000 samples from the RealNews subset of the C4 dataset (Raffel et al., 2020), reducing embedding dimensions from 768 to 16. After testing various principal component counts (ranging from 512 to 16), we found 16 to yield the best results. During our experiments, we evaluated both settings (with and without PCA). Specifically, PCA improved robustness against paraphrasing attacks when Euclidean distance was used (except on the BookSum dataset), but consistently degraded performance when cosine similarity was employed across all datasets. The results of these experiments are summarized in Table 5 in Appendix G.

Across all experiments, the decay factor was set to $K = 250$, as this value provided an optimal trade-off between performance under both non-paraphrased and paraphrased conditions (see Appendix F for an ablation study on this). The threshold β was determined empirically during the detection (refer to Appendix I for details) to achieve the specified FP rates (1% or 5%). The intervals $[0.68, 0.76]$ for cosine similarity and $[0.28, 0.36]$ for Euclidean distance with PCA and $[0.4, 0.55]$ for Euclidean distance without PCA were found to be near-optimal. While the intervals could have been further optimized for each dataset individually, we chose not to do so to show the general performance of our method.

4.1 Models and Datasets

For our experiments, we used the same fine-tuned version of OPT-1.3B (Zhang et al., 2022) as in Hou et al. (2024a,b)[5] to ensure fair comparison. However, we emphasize that our method is model-agnostic and it treats the LLM as a black-box text generator. As such, if the method performs well on one family of models, it is expected to generalize to others. To support this, we also tested our method on Gemma3-4B model (Team et al., 2025) and observed similar results (see Appendix C). For semantic embedding, we utilized Instructor-Large model[6] (Su et al., 2023). Appendix B includes additional details on the experimental configurations.

In our experiments, we used three English datasets: RealNews subset of C4[7] (Raffel et al., 2020), BookSum[8] (Kryscinski et al., 2022), and Reddit-TIFU[9] (Kim et al., 2019) datasets, as in Hou et al. (2024a,b). Specifically, 1000 samples from each dataset were chosen to analyze the detection performance and the text quality. Each sample was segmented into sentences, with the first sentence serving as the *prompt* to the LLM.

We evaluated text quality after applying *SimMark* using the following metrics:

- **Perplexity (PPL)**: Measures how surprising the text is to an *oracle* LLM[10].
- **Tri-gram Entropy (Ent-3)** (Zhang et al., 2018): Assesses textual diversity via the entropy of the tri-grams distribution.
- **Semantic Entropy (Sem-Ent)** (Han et al., 2022): Measures semantic informativeness and diversity of the text.

4.2 Paraphrase Attack

To evaluate the robustness of *SimMark* against paraphrase attacks, we tested it using three paraphrasers: I. Pegasus paraphraser[11] (Zhang et al., 2020), II. Parrot paraphraser[12] (Damodaran, 2021), III. GPT-3.5-Turbo (OpenAI, 2022).

[4] Like (Hou et al., 2024a), all results are from a single run.

[5] Used AbeHou/opt-1.3b-semstamp (1.3B) model.

[6] Used hkunlp/instructor-large (335M) model.

[7] Dataset card: allenai/c4 (Validation split)

[8] Dataset card: kmfoda/booksum (Validation split)

[9] Dataset card: ctr4si/reddit_tifu (Train split, short subset)

[10] Used facebook/opt-2.7b following (Hou et al., 2024a,b).

[11] Used tuner007/pegasus_paraphrase (568M) model.

[12] Used parrot_paraphraser_on_T5 (220M) model.

Dataset	Algorithm	No Paraphrase	Pegasus	Pegasus-Bigram	Parrot	Parrot-Bigram	GPT3.5	GPT3.5-bigram	Avg. Paraphrased
RealNews	UW (Zhao et al.)	**99.9** / 99.1 / **99.9**	98.5 / 85.6 / 95.3	97.9 / 73.5 / 91.7	97.9 / 70.9 / 91.9	97.4 / 62.8 / 89.4	**97.4** / 59.1 / **87.9**	**93.7** / 37.0 / 70.8	97.1 / 64.8 / 87.8
	KGW (Kirchenbauer et al.)	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	86.2 / 32.4 / 55.4
	SIR (Liu et al.)	**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	89.1 / 54.4 / 67.2
	SemStamp (Hou et al.)	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	91.0 / 56.0 / 71.6
	k-SemStamp (Hou et al.)	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	95.6 / 74.0 / 84.2
	Cosine-*SimMark* (Ours)	99.6 / 96.8 / 98.8	99.2 / **90.3** / **98.2**	**99.1** / **90.3** / **97.9**	**98.7** / **88.1** / **97.2**	**98.8** / **87.3** / **97.6**	95.7 / **59.7** / 86.7	92.0 / 38.8 / **73.7**	**97.2** / **75.8** / **91.9**
	Euclidean-*SimMark* [**] **(Ours)**	99.8 / 98.5 / 99.3	97.2 / 72.3 / 89.1	96.9 / 70.0 / 87.4	95.7 / 60.2 / 82.5	95.7 / 59.1 / 81.5	94.1 / 51.6 / 76.2	88.2 / 29.7 / 53.5	94.6 / 57.2 / 78.4
BookSum	UW	**100** / **100** / **100**	**99.5** / 89.8 / **98.5**	98.6 / 71.2 / 93.0	98.9 / 79.4 / 94.8	98.6 / 72.1 / 92.9	93.2 / 24.6 / 57.9	86.0 / 9.2 / 30.5	95.8 / 57.7 / 77.9
	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	91.0 / 41.2 / 67.8
	SIR	**100** / 99.8 / **100**	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	89.5 / 59.7 / 66.6
	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	95.0 / 72.8 / 82.1
	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	97.8 / 81.5 / 91.2
	Cosine-*SimMark* (Ours)	99.8 / 98.8 / 99.5	**99.5** / 93.3 / **98.5**	**99.6** / **94.1** / **98.5**	**99.3** / **88.5** / **98.0**	**99.3** / 87.0 / **98.2**	97.1 / 62.5 / 86.9	94.5 / 41.6 / 74.2	98.2 / 77.8 / 92.4
	Euclidean-*SimMark* (Ours)	**100** / **100** / **100**	98.8 / 82.6 / 94.9	98.6 / 80.4 / 93.4	97.9 / 75.3 / 91.1	97.9 / 73.3 / 91.6	**99.7** / **94.4** / **98.8**	**99.5** / **91.9** / **97.6**	**98.7** / **83.0** / **94.6**
Reddit-TIFU	UW	**99.9** / **99.5** / **99.8**	97.3 / 73.4 / 91.1	94.1 / 48.3 / 77.2	90.6 / 37.1 / 64.0	89.2 / 33.7 / 60.4	86.3 / 26.9 / 52.9	74.3 / 13.2 / 30.0	88.6 / 38.8 / 62.6
	KGW	99.3 / 97.5 / 98.1	94.1 / 87.2 / 87.2	91.7 / 67.2 / 67.6	79.5 / 22.8 / 43.3	82.8 / 27.6 / 49.7	84.1 / 27.3 / 50.9	79.8 / 19.3 / 41.3	85.3 / 41.9 / 56.7
	SIR	99.6 / 97.2 / 99.7	90.0 / 48.7 / 77.4	90.9 / 33.1 / 71.1	87.1 / 15.0 / 50.9	86.9 / 12.8 / 49.8	91.1 / 15.0 / 61.4	84.3 / 5.5 / 39.1	88.4 / 21.7 / 58.3
	SemStamp	99.7 / 97.7 / 98.2	98.4 / 92.8 / 95.4	98.0 / 89.0 / 92.9	90.2 / 56.2 / 70.5	93.9 / 71.8 / 82.3	87.7 / 47.5 / 58.2	87.4 / 43.8 / 55.9	92.6 / 66.9 / 75.9
	Cosine-*SimMark* (Ours)	99.1 / 96.3 / 97.6	98.9 / 94.5 / 96.4	98.7 / **93.6** / 96.1	98.5 / **91.6** / 96.0	98.5 / **91.7** / 95.5	97.8 / **88.4** / 94.7	96.3 / **72.9** / **88.4**	98.1 / **88.8** / **94.5**
	Euclidean-*SimMark* [**] **(Ours)**	99.8 / 98.7 / 99.2	**99.0** / **94.7** / **97.6**	**99.0** / 91.9 / **96.2**	97.8 / 75.9 / 89.5	97.7 / 76.4 / 90.4	**98.7** / 83.7 / **95.2**	**96.8** / 65.8 / 87.3	**98.2** / 81.4 / 92.7

Table 1: Performance of different algorithms across datasets and paraphrasers, evaluated using ROC-AUC ↑ / TP@FP=1% ↑ / TP@FP=5% ↑, respectively (↑: higher is better). In each column, **bold** value indicates the best performance for a given dataset and metric, while underlined value denotes the second-best. *SimMark* consistently outperforms or is on par with other state-of-the-art methods across datasets, paraphrasers, and is the best on average.

Algorithm	**PPL** ↓	**Ent-3** ↑	**Sem-Ent** ↑
No watermark	11.89	11.43	3.10
UW	13.02	11.96	2.90
KGW	14.92	11.32	2.95
SIR	20.34	11.57	3.18
SemStamp	12.89	11.51	3.17
k-SemStamp	11.82	11.48	3.11
Cosine-*SimMark* (Ours)	12.69	11.50	3.39
Euclidean-*SimMark*** (Ours)	9.67	11.50	3.28

Table 2: Comparison of the quality of text watermarked using different algorithms on BookSum dataset (↓: lower is better, ↑: higher is better). *SimMark* yields quality metrics comparable to the no-watermark baseline, indicating minimal impact on text quality and semantic diversity. In contrast, token-level methods (top three rows) notably degrade the text quality, especially in terms of perplexity.

Kirchenbauer et al. (2024) observed that prompting models to paraphrase entire texts often results in summarized outputs, with the summarization ratio worsening for longer inputs. To prevent any information loss caused by such summarization, we adopted a sentence-by-sentence paraphrasing scheme, which also ensures our results are comparable to Hou et al. (2024a,b). Refer to Appendix K to find the prompts used with GPT-3.5-Turbo. The quality of paraphrases was assessed using BertScore[13] (Zhang* et al., 2020), with all settings consistent with Hou et al. (2024a,b). The bottom part of Figure 2 demonstrates the paraphrase attack and detection phase in more detail. We also included the results for the *bigram* paraphrase attack introduced by Hou et al. (2024a), with identical settings (25 rephrases for each sentence when using Pegasus and Parrot, etc.). This attack involves generating multiple paraphrases for each sentence, and choosing the one that increases the likelihood of disrupting statistical signatures embedded in the text, especially for token-level algorithms (Hou et al., 2024a). While this attack significantly impacts most other methods, *SimMark* demonstrates greater robustness against it. We must highlight that k-SemStamp relies on domain-specific clustering of semantic spaces, making it domain-dependent. In contrast, both *SimMark* and SemStamp are domain-independent. Still, *SimMark* outperforms both in almost all cases across various datasets and metrics, further underscoring its universality and robustness.

4.3 Robustness to Sentence-Level Perturbations

To further evaluate the robustness of our method and its real-world applicability, we introduce more challenging attack scenarios: *Paraphrase+Drop Attack* and *Paraphrase+Merge Attack*. These scenarios simulate realistic adversarial editing strategies where a user attempts to remove or merge sentences after paraphrasing, while maintaining fluency.

Paraphrase+Drop assumes that an adversary not only paraphrases the text but also drops sentences deemed redundant. This reflects common editing practices, especially since LLMs often produce verbose outputs due to imperfect reward modeling (Chiang and Lee, 2024). To simulate this, we first paraphrase the input text and then randomly drop sentences with a specified probability p. Similarly, *Paraphrase+Merge* is designed to test robustness under more subtle structural changes. After paraphrasing, we replace end-of-sentence punctuations (e.g., ., ?, !) with the word "and" with probability $0 < p < \frac{1}{2}$. In our experiments, we avoid higher

[**] PCA is applied.

[13] Used deberta-xlarge-mnli (750M) (He et al., 2021).

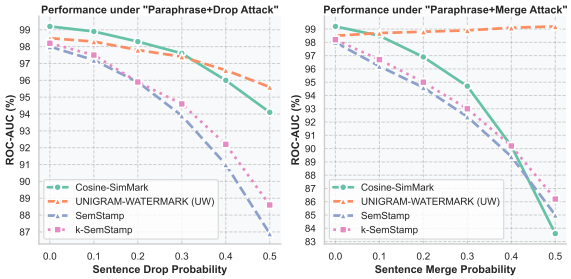

Figure 3: Detection performance under two adversarial settings using RealNews dataset: **Left**: *Paraphrase+Drop Attack*, where random sentences are removed after paraphrasing. *SimMark*, in almost all parameter regimes, outperforms other methods under this attack. **Right**: *Paraphrase+Merge Attack*, where sentence boundaries (punctuations) are probabilistically replaced with "and" to merge sentences. Although *SimMark* performs best among sentence-level approaches, UW remains highly robust due to its token-level nature.

values of p as they result in unnaturally long and less fluent sentences. This setup simulates a realistic scenario where an adversary attempts to merge sentences while preserving the overall coherence of the text. These combined attacks serve to stress-test the resilience of watermarking methods under more naturalistic adversarial conditions that go beyond simple paraphrasing.

4.4 Results & Discussion

We compared the performance of *SimMark* against SOTA watermarking algorithms through extensive experiments. Our primary baseline was SemStamp, a sentence-level semantic watermarking method. We also included k-SemStamp, an improvement over SemStamp tailored to specific domains. Results for SemStamp, k-SemStamp, KGW and SIR, were extracted directly from Hou et al. (2024a,b)[14].

Table 1 presents detection performance pre and post paraphrase attacks, while Table 2 provides text quality evaluation results. Our algorithm impacts the text quality minimally while being effective and consistently outperforming or matching other SOTA methods, achieving the highest average performance across all paraphrasers and datasets. Notably, our method *SimMark* (domain-independent), surpasses the primary baseline, SemStamp (domain-independent), and is on par or exceeds k-SemStamp (domain-dependent). A key aspect to consider is that the fine-tuning of SemStamp and k-SemStamp's embedding model on text paraphrased by Pegasus likely contributes to their improved robustness against this paraphraser but may introduce bias. Additionally, the results for the Reddit-TIFU dataset were only available for SemStamp and not k-SemStamp, likely due to the dataset's informal, diverse text style and k-SemStamp's limitation for text to belong to a specific domain, such as news articles or scientific writings (Hou et al., 2024b).

Figure 3 presents the ROC-AUC performance of UW (the best token-level method in our experiments), *SimMark*, k-SemStamp, and SemStamp under *Paraphrase+Drop* and *Paraphrase+Merge* attacks, evaluated on the RealNews dataset. Under *Paraphrase+Drop*, across most parameter regimes, *SimMark* outperforms all other methods, sustaining higher detection performance even as the attack intensity increases. While SimMark shows superior performance among sentence-based methods under *Paraphrase+Merge*, UW maintains highest robustness because it operates at the token level.

Regarding sampling efficiency, for BookSum dataset for instance, *SimMark* required an average of 7.1 samples per sentence from the LLM, compared to k-SemStamp and SemStamp, which averaged 13.3 and 20.9 samples, respectively (Hou et al., 2024b). This demonstrates that our method not only outperforms these baselines but is also 2-3 times more efficient. See Appendix J for further analysis of this (including theoretical estimates). Finally, refer to Appendix E for qualitative examples of *SimMark*.

5 Conclusion

In this paper, we introduced *SimMark*, a similarity-based, robust sentence-level watermarking algorithm. Unlike existing approaches, *SimMark* operates without requiring access to the internals of the model, ensuring compatibility with a wide range of LLMs, including API-only models. By utilizing a pre-trained general-purpose embedding model and integrating a *soft* counting mechanism, *SimMark* combines robustness against paraphrasing with applicability to diverse domains. Experimental results show that *SimMark* outperforms SOTA sentence-level watermarking algorithms in both efficiency and robustness to paraphrasing, representing a step forward in fully semantic watermarking for LLMs.

[14]Despite Hou et al. (2024a,b) releasing their code and data, we were unable to reproduce their reported results fully. Consequently, there are minor discrepancies between our reproduction results (shown in Figure 3 for cases with $p = 0$) and those presented in Table 1 (extracted directly from their paper). Additionally, the results reported in Table 2 (our reproduction) also show slight differences from their paper, likely due to hyperparameter details that were not explicitly documented.

Limitations

While *SimMark* demonstrates outstanding performance, there are still some areas that warrant further exploration:

Rejection Sampling Overhead. The rejection sampling process requires generating multiple candidate sentences until a valid sentence is accepted. Although our method is significantly (2-3 times) more efficient than prior approaches such as SemStamp and k-SemStamp, there is still a notable decrease in generation speed due to rejection sampling. Techniques like batch sampling or parallel sampling could potentially mitigate this issue, though at the expense of higher computational resource usage. Future research should focus on optimizing the method to balance efficiency and resource requirements.

Resistance to More Advanced Attacks. While *SimMark* demonstrates robustness against paraphrasing attacks, it may not be immune to more sophisticated adversarial transformations. In particular, detection could become less effective when watermarked text is interleaved with or embedded within larger body of of unwatermarked content. Additionally, although reverse engineering the exact watermarking rules is non-trivial, an adversary may attempt a spoofing attack by approximating our setup—for instance, by employing a publicly available embedding model or fitting a PCA model with publicly available datasets. While such attempts may not perfectly replicate the original embedding distribution, they could still pose a threat. We leave a thorough investigation into vulnerabilities and corresponding defences against reverse engineering to future work.

Dependency on Predefined Intervals. In our experiments, we used consistent predefined intervals across all datasets and observed consistently strong performance. Notably, we did not observe any noticable degradation in text quality due to this interval constraint during rejection sampling (as shown in Table 2), likely because the constraint applies only to consecutive sentences. Nonetheless, slight variations in the embeddings similarity distribution of LLM-generated text across different models may impact watermarking effectiveness. Adaptive strategies for setting these intervals dynamically (or pseudo-randomly) could not only improve performance but also make reverse-engineering the algorithm more difficult.

Ethical Considerations[15]

Potential Risks. By enabling robust detection of LLM-generated text, particularly under paraphrasing attacks, *SimMark* tries to address ethical concerns surrounding the transparency and accountability of AI-generated content. However, like any watermarking algorithm, there are potential risks, such as falsely implicating human authors or adversaries developing more advanced techniques for spoofing attacks or bypassing detection. We acknowledge these limitations and advocate for the responsible deployment of such tools in combination with other verification mechanisms to mitigate these risks and ensure ethical, fair deployment. The primary goal of this work is to advance research in watermarking techniques to support the responsible use of LLMs. We believe that the societal impacts and ethical considerations of our work align with those outlined in Weidinger et al. (2021).

Use of Models and Datasets. Our research used datasets and pretrained models from the Hugging Face Hub[16], a public platform hosting machine learning resources under various licenses. We adhered to all license terms and intended usage guidelines for each artifact, which are documented on their individual Hugging Face model or dataset cards cited in the paper. All resources were used solely for research purposes in accordance with their intended use and respective licenses, with no additional data collection, scraping, or annotation performed by the authors. The datasets employed are publicly available, widely used in prior research, and to the best of our knowledge, free of personally identifiable information or offensive content. Any outputs generated by their method are intended for academic use, not for real-world or commercial applications, and no personal or sensitive data was processed. Any new artifacts we create (e.g., watermarked samples) are intended solely for academic evaluation, and we do not release any derivative data that violates original licensing terms.

References

Scott Aaronson and Hendrik Kirchner. 2022. Watermarking gpt outputs.

[15] Generative AI tools, such as ChatGPT, were used to refine this manuscript. The authors retain full responsibility for all content presented in the paper, ensuring adherence to academic integrity and ethical research practices.

[16] https://huggingface.co

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: 4th International Workshop, IH 2001 Pittsburgh, PA, USA, April 25–27, 2001 Proceedings 4*, pages 185–200. Springer.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Yapei Chang, Kalpesh Krishna, Amir Houmansadr, John Frederick Wieting, and Mohit Iyyer. 2024. PostMark: A robust blackbox watermark for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8969–8987, Miami, Florida, USA. Association for Computational Linguistics.

Cheng-Han Chiang and Hung-yi Lee. 2024. Over-reasoning and redundant calculation of large language models. *arXiv preprint arXiv:2401.11467*.

Prithiviraj Damodaran. 2021. Parrot: Paraphrase generation for nlu.

Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. 2023. The stable signature: Rooting watermarks in latent diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22466–22477.

Matthew Finlayson, Xiang Ren, and Swabha Swayamdipta. 2024. Logits of api-protected llms leak proprietary information. *arXiv preprint arXiv:2403.09539*.

Yu Fu, Deyi Xiong, and Yue Dong. 2024. Watermarking conditional text generation for ai detection: Unveiling challenges and a semantic-aware watermark remedy. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):18003–18011.

Futurism. 2023. Cnet quietly deletes ai-generated articles amid backlash. Accessed: January 28, 2025.

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*, pages 934–950, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jifei Hao, Jipeng Qiang, Yi Zhu, Yun Li, Yunhao Yuan, and Xiaoye Ouyang. 2025. Post-hoc watermarking for robust detection in text generated by large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5430–5442, Abu Dhabi, UAE. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

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

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

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

Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. 2019. Abstractive summarization of Reddit posts with multi-level memory networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2519–2531, Minneapolis, Minnesota. Association for Computational Linguistics.

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

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

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

Wojciech Kryscinski, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2022.

BOOKSUM: A collection of datasets for long-form narrative summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6536–6558, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tharindu Kumarage, Paras Sheth, Raha Moraffah, Joshua Garland, and Huan Liu. 2023. How reliable are AI-generated-text detectors? an assessment framework using evasive soft prompts. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1337–1349, Singapore. Association for Computational Linguistics.

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*.

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

OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, and 263 others. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

OpenAI. 2022. ChatGPT.

Leyi Pan, Aiwei Liu, Zhiwei He, Zitian Gao, Xuandong Zhao, Yijian Lu, Binglin Zhou, Shuliang Liu, Xuming Hu, Lijie Wen, Irwin King, and Philip S. Yu. 2024. MarkLLM: An open-source toolkit for LLM watermarking. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 61–71, Miami, Florida, USA. 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*, 21(140):1–67.

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

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2023. One embedder, any task: Instruction-finetuned text embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1102–1121, Toronto, Canada. Association for Computational Linguistics.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Mohammadreza Teymoorianfard, Shiqing Ma, and Amir Houmansadr. 2025. Vidstamp: A temporally-aware watermark for ownership and integrity in video diffusion models. *Preprint*, arXiv:2505.01406.

Mercan Topkara, Umut Topkara, and Mikhail J Atallah. 2006. Words are not enough: sentence level natural language watermarking. In *Proceedings of the 4th ACM international workshop on Contents protection and security*, pages 37–46.

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, and 1 others. 2021. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*.

Xi Yang, Kejiang Chen, Weiming Zhang, Chang Liu, Yuang Qi, Jie Zhang, Han Fang, and Nenghai Yu. 2023. Watermarking text generated by black-box language models. *arXiv preprint arXiv:2305.08883*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, and 1 others. 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. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and Bill Dolan. 2018. Generating informative and diverse conversational responses via adversarial information maximization. *Advances in Neural Information Processing Systems*, 31.

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 Aditional Related Work

A.1 Token-Level Watermarking

Zhao et al.'s (2023) UNIGRAM-WATERMARK (UW) builds upon KGW by fixing the red and green lists instead of pseudo-randomly selecting them, proving that, compared to KGW, their method is more robust to paraphrasing and editing (Zhao et al., 2023). However, as outlined by Hou et al. (2024a), this algorithm can be reverse-engineered, rendering it impractical for high-stakes, real-world applications.

The Semantic Invariant Robust (SIR) watermark in Liu et al. (2023) is also similar to KGW but is designed to be less sensitive to attacks involving synonym replacement or advanced paraphrasing. SIR achieves this by altering the LLM logits based on the semantics of previously generated tokens, using a semantic embedding model to compute semantic representations and training a model that adjusts LLM's logits based on the semantic embeddings of prior tokens (Liu et al., 2023).

A.2 Post-hoc Watermarking

Chang et al.'s (2024) PostMark is a post-hoc watermarking algorithm designed to work without access to model logits, making it compatible with API-only LLMs. It constructs an input-dependent set of candidate words using semantic embeddings and then prompts another LLM (e.g., GPT-4o) to insert these words into the generated text. Detection relies on statistical analysis of the inserted words. While PostMark's compatibility with black-box LLMs is a strength, the approach is computationally expensive—watermarking 100 tokens is estimated to cost around $1.2 USD (Chang et al., 2024).

Yang et al. (2023) propose another post-hoc method that encodes each word in the text as a binary bit via a Bernoulli distribution ($p = 0.5$), embedding the watermark through synonym substitution: words representing bit 0 are replaced with synonyms representing bit 1. Detection is again done via statistical testing. However, this method is fragile: synonyms are not reliably preserved under paraphrasing and often fail to capture subtle contextual meanings, which can noticeably degrade text quality and watermark robustness.

Hao et al. (2025) is a post-hoc watermarking technique similar to Yang et al. (2023) that improves robustness by selecting semantically or syntactically essential words—those less likely to be altered during paraphrasing—as anchor points for embedding. The method uses paraphrase-based lexical substitution to insert watermarks while preserving the original semantics. However, empirical results in Chang et al. (2024) demonstrate that this method is not robust to paraphrasing compared to other methods such as KGW, SemStamp, and PostMark.

B Experimental Settings

In all combinations of the experiments, following Kirchenbauer et al. (2023), sampling from the LLM was performed with a temperature of 0.7 and a repetition penalty of 1.05, while the minimum and the maximum number of generated tokens were set to 195 and 205, respectively. The maximum number of rejection sampling iterations was set to 100, again to align with the code provided by Hou et al. (2024a,b). However, this setting reflects a trade-off between detection performance and generation speed. Based on our experiments, setting it to 25 achieves strong performance, with higher values offering only marginal improvements (see Appendix D). For token-level watermarking baselines, in cases where results were not directly extracted from Hou et al. (2024a,b), we employed the open-source MarkLLM watermarking framework (Pan et al., 2024), with their recommended configurations ($\gamma = 0.5, \delta = 2$, `prefix_length=1`, etc.) to run the experiments.

The majority of the experiments, including text generation and detection tasks, were conducted on a workstation equipped with an Intel Core i9 processor, 64GB of RAM, and an Nvidia RTX 3090 GPU with 24GB of VRAM. Some of the experiments involving bigram paraphrasing were performed on compute nodes with an Nvidia V100 GPU with 32GB of VRAM. Generating 1000 *SimMark*-watermarked samples using OPT-1.3B required approximately 8-10 GPU hours on a single RTX 3090.

C Additional Experimental Results

To demonstrate the model-agnostic nature of *SimMark*, we applied our algorithm to the recently released Gemma3-4B model[1] (Team et al., 2025). We evaluated both Cosine-*SimMark* and Euclidean-*SimMark* under different paraphrasing models across the same three datasets as before: RealNews

[1]We employed google/gemma-3-4b (4B) model.

subset of C4, BookSum, and Reddit-TIFU. The effectiveness and robustness of watermarking techniques can depend heavily on the characteristics of the underlying LLM and the nature of the generated text. To maintain consistency and reliability across experiments, we made the following modifications:

- **Predefined Interval Adjustment:** The sentences' embedding similarity distribution under Gemma3-4B differed from those in OPT-1.3B, requiring new intervals. We set the predefined interval to [0.86, 0.90] for cosine similarity (without PCA), and [0.11, 0.16] for Euclidean distance (with PCA).
- **Threshold Transferability:** In contrast to our earlier experiments—where the detection threshold β was determined per dataset—we fixed β across all datasets in these experiments. Specifically, we determined the threshold using only non-watermarked data from the BookSum dataset, and then applied it across all three datasets without modification. This approach simulates a more realistic setting where the detector is calibrated on a single corpus but expected to generalize to others. The results demonstrate that our method maintains high detection performance even under this general configuration.
- **Longer Generations:** Since Gemma3-4B tends to generate longer sentences compared to OPT-1.3B model that we employed earlier, we increased the number of generated tokens from 200 to 300 to ensure a sufficient number of sentences for reliable hypothesis testing.

Table 3 reports the detection performance in terms of ROC-AUC ↑ / TP@1%FP ↑ / TP@5%FP ↑, for each setting (↑: higher is better). Across all datasets and paraphrasing scenarios, *SimMark* remains highly effective, with both cosine similarity and Euclidean distance variants maintaining strong ROC-AUC and TP rates. These results affirm that *SimMark* maintains its performance across different LLM families (e.g., OPT and Gemma3) and datasets/domains, further validating the general applicability of our proposed algorithm.

D Effect of Sampling Budget on Detection Performance

To better understand the trade-off between generation speed and detection performance, we analyze the impact of the `max_trials` hyperparameter, which defines the upper limit on the number of rejection sampling iterations during watermark injection. While we set this value to 100 in our main experiments (to align with prior works of Hou et al. (2024a,b)), it is important to examine whether such a large value is necessary.

Figure 4 shows two evaluation metrics—ROC-AUC ↑ and TP@FP=1% ↑ (↑: higher is better)—on RealNews dataset for cosine-*SimMark* and OPT-1.3B model under different values of `max_trials`. As shown in the plots, performance improves significantly when increasing `max_trials` from 5 to 25, but plateaus thereafter. In particular, both ROC-AUC and TP@FP=1% show diminishing returns beyond 25 trials, indicating that additional sampling brings little performance gain.

These results suggest that setting `max_trials` to 25 achieves a good balance between robustness and efficiency, and using larger values (e.g., 100) is not strictly necessary in practice. These findings, together with the average sampling statistics reported in the main paper (e.g., 7.1 samples per sentence), highlight *SimMark*'s ability to balance robustness with generation speed compared to SemStamp and k-SemStamp (e.g., 20.9 and 13.3 samples per sentence, respectively).

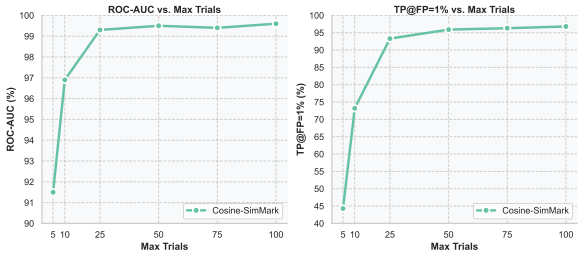

Figure 4: Impact of the maximum number of rejection sampling trials on detection performance. Increasing `max_trials` improves both ROC-AUC ↑ and TP@1%FP ↑ (↑: higher is better), but the improvement plateaus around 25. Results are reported on RealNews dataset using cosine-*SimMark* and OPT-1.3B model.

E Examples of Watermarked Text

Figures 5 and 6 provide examples of text generated with and without the *SimMark* watermark using OPT-1.3B. These examples illustrate the imperceptibility of the watermark to human readers while enabling robust detection through our proposed algorithm. They also highlight *SimMark*'s robustness to paraphrasing while maintaining quality comparable to non-watermarked text.

Dataset	Method	No Paraphrase	Pegasus	Pegasus-Bigram	Parrot	Parrot-Bigram	Avg. Paraphrased
RN	Cosine-*SimMark*	99.5 / 96.2 / 97.9	93.5 / 57.9 / 75.6	93.0 / 54.9 / 73.9	92.3 / 50.1 / 71.7	92.1 / 49.2 / 72.8	92.7 / 53.0 / 73.5
	Euclidean-*SimMark***	99.5 / 97.4 / 98.2	93.2 / 65.3 / 79.5	91.8 / 61.8 / 78.5	91.9 / 58.0 / 73.9	91.5 / 58.7 / 72.8	92.1 / 61.0 / 76.2
BS	Cosine-*SimMark*	99.9 / 99.6 / 99.8	97.7 / 75.8 / 90.5	97.4 / 73.3 / 90.0	97.0 / 67.8 / 87.6	96.9 / 67.7 / 86.7	97.2 / 71.1 / 88.7
	Euclidean-*SimMark***	99.9 / 99.7 / 99.9	97.6 / 82.0 / 91.4	97.4 / 77.9 / 89.9	97.4 / 78.0 / 91.2	91.5 / 58.7 / 72.8	96.0 / 74.2 / 86.3
TIFU	Cosine-*SimMark*	99.8 / 99.1 / 99.5	97.4 / 78.8 / 91.5	97.0 / 76.6 / 89.3	89.4 / 32.8 / 61.7	90.7 / 36.6 / 63.8	93.6 / 56.2 / 76.6
	Euclidean-*SimMark***	99.6 / 98.8 / 99.1	97.3 / 82.0 / 91.4	96.9 / 81.0 / 90.1	92.2 / 53.6 / 73.9	92.8 / 58.4 / 76.0	94.8 / 68.8 / 82.8

Table 3: Performance of *SimMark* using Gemma3-4B model across paraphrasers and datasets (RealNews denoted as RN, BookSum denoted as BS, and Reddit-TIFU denoted as TIFU). Each cell reports AUC ↑ / TP@FP=1% ↑ / TP@FP=5% ↑ (↑: higher is better). The results demonstrate that *SimMark* maintains strong performance across different datasets and paraphrasing conditions, highlighting its robustness and model-agnostic nature.

Non-Watermarked: *I've always had an interest in colour.* I like to paint and colour (9.77e-13). It's relaxing and it makes you feel good (9.31e-13). But, what I haven't done until now is paint on a piece of paper. I'm going to use different colours on my collages (8.89e-7). Using lots of colours gives a dynamic to the pieces (3.57e-14). I can also use a bit of white space to draw the eye (4.85e-6). You can see the other images in this project here (9.47e-9). I used four different papers for the collage (3.7e-7). Copying those images onto other pieces of paper gave me a few ideas (1.52e-11). Also, by using white space I was able to draw out more details without getting them all in the way (1.02e-7). I like making collages and I think that these are a great way to display them (0.005). I wanted to show different aspects of my painting (3.53e-8). I wanted to limit myself to only a few colours but I did want to limit myself to black and white (1.3e-7). I really love the colour black (8.24e-7). It always makes me think about nature. I like drawing sketches and I enjoy painting as well.

$z_{soft} = 0.14 < 5.033$

Cosine-*SimMark*: *I've always had an interest in colour.* I don't know why. But I have a couple of tins of it, and that makes me happy. The first piece of art I bought was a painting by Japanese artist Kata-Yoko on canvas. It made me feel very happy. That is one of the things I want to do when I retire: to become an artist. My mother is still alive. She would be very pleased with all this. I think it is important for a young person to see these people and their work. I am always taking pictures of my family. My father is a retired teacher, so he is not around too much these days. But he keeps saying that he wants to watch me become an artist. That is really comforting. I want to develop my own style There is no time like the present. Let's hope I can make some money doing this. I am 60 now. There is no time like the present. I hope I can combine work and pleasure. The other day, I made sketches on paper.

$z_{soft} = 9.48 > 5.033$

Paraphrased cosine-*SimMark*: *I've always been fascinated by color.* I can't explain the reason behind my fascination with color (5.24e-13). But having a few cans of it brings me joy (0.15). The initial artwork I purchased was a canvas painting by the Japanese artist Kata-Yoko. It brought me a great sense of joy. One of my aspirations for retirement is to pursue a career as an artist. My mother remains living. She would be delighted with all of this. I believe it is crucial for a young individual to witness these individuals and their creations. I constantly capture memories of my family through photography (0.01). My father, who is a former educator, is not present often these days due to his retirement. However, he continues to express his desire to witness my journey as I pursue becoming an artist (0.48). That is truly reassuring. I aspire to cultivate a unique artistic expression. Now is the best time. I hope to be able to earn some income from pursuing this passion. I have reached the age of 60 (0.16). Now is the perfect moment. I hope to find a balance between my work and my passion. I hope to find a balance between my work and my passion. Recently, I created some drawings on paper.

$z_{soft} = 6.94 > 5.033$

Figure 5: Example of text generated with and without cosine-*SimMark* using RealNews dataset and OPT-1.3B model. The first sentence (in black) is the prompt for the model, the green sentences are valid, and red sentences are invalid/partially valid. Numbers in parentheses represent the *soft count* for partially valid sentences. The top panel shows non-watermarked text, which fails to produce a significant detection signal ($z_{\text{soft}} = 0.14 < 5.033$, false negative). The middle panel demonstrates text generated using *SimMark* with cosine similarity-based watermarking, producing a strong detection signal ($z_{\text{soft}} = 9.48 > 5.033$). The bottom panel shows paraphrased watermarked text using GPT-3.5-Turbo, where the embedded watermark remains detectable despite semantic alterations ($z_{\text{soft}} = 6.94 > 5.033$).

F Ablation Study on Soft Count Smoothness Factor K

In this section, we analyze the impact of the smoothness factor K on the performance of *SimMark*. Recall that K controls the degree of smoothness in the soft counting mechanism as defined in Eq. (1). A larger K makes the soft counting function behave more like a step function, while smaller values provide smoother transitions between valid and invalid sentences. Table 4 presents the results of this ablation study, conducted on RealNews dataset with Pegasus as the paraphraser. Metrics include ROC-AUC, TP@FP=1%, and TP@FP=5%. Higher values indicate better performance across all metrics. The results demonstrate the following trends:

- A smoothness factor of $K = 250$ provides a good trade-off, achieving strong performance both before and after paraphrasing attacks for both cosine-*SimMark* and Euclidean-*SimMark*.
- For $K = \infty$, corresponding to regular counting with a step function, the performance is slightly higher in the absence of paraphrasing but significantly degrades under paraphrasing attacks, highlighting the benefits of soft counting in adversarial scenarios.

These findings confirm that soft counting loses a

**PCA is applied.

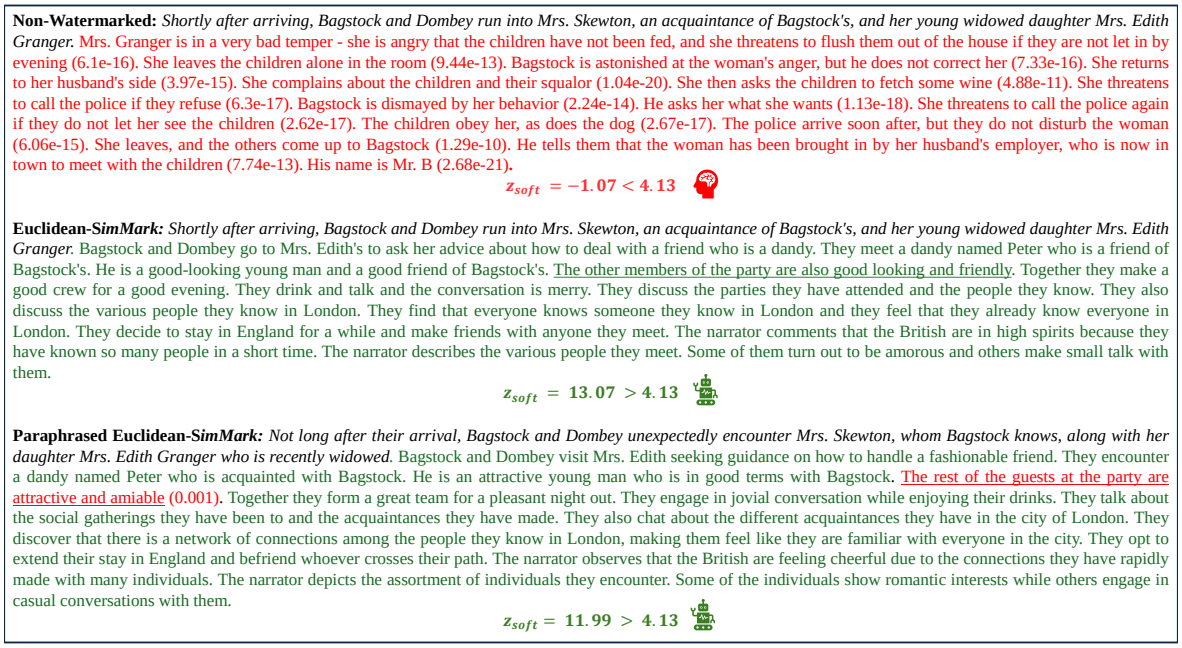

Figure 6: Example of text generated with and without Euclidean-*SimMark* using BookSum dataset and OPT-1.3B model. The first sentence (in black) is the prompt for the model, the green sentences are valid, and red sentences are invalid/partially valid. Numbers in parentheses represent the *soft count* for partially valid sentences. The top panel shows the non-watermarked text, which fails to produce a significant detection signal ($z_{\text{soft}} = -1.07 < 4.13$, false negative). The middle panel demonstrates text generated using *SimMark* with Euclidean distance-based watermarking, producing a strong detection signal ($z_{\text{soft}} = 13.07 > 4.13$). The bottom panel shows paraphrased watermarked text using GPT-3.5-Turbo, where the embedded watermark remains detectable despite semantic alterations ($z_{\text{soft}} = 11.99 > 4.13$).

small amount of performance when no paraphrasing is applied, but it gains substantial robustness under paraphrasing. For example, TP@FP=1% improves by 1.6–2.3% for Pegasus-paraphrased text when $K = 250$, and the improvement is likely to be even more significant for stronger paraphrasers.

G Ablation Study on Impact of PCA

Table 5 presents the results of an ablation study investigating the impact of applying PCA to reduce the dimensionality of sentence embeddings across RealNews, BookSum, and Reddit-TIFU datasets. Metrics include ROC-AUC, and TP at fixed FP rates (FP=1% and FP=5%). Higher values indicate better performance across all metrics, with PCA applied to embeddings to explore its effect on detection accuracy and robustness.

The results reveal that the effect of PCA depends on the choice of similarity measure. For *Euclidean distance-based SimMark*, applying PCA generally improves robustness against paraphrasing attacks across most datasets, except for the BookSum dataset. This improvement likely arises because reducing dimensionality helps mitigate noise in the embeddings, especially after the paraphrasing attack. On the other hand, for *cosine similarity-based SimMark*, applying PCA reduces performance across all datasets. This reduction may be due to PCA altering the embeddings in a way that disrupts the angular relationships critical for cosine similarity calculations. These findings highlight the importance of adapting PCA usage based on the similarity measure employed to achieve optimal watermarking performance.

H Finding an Optimal Interval

Figure 7 shows the distribution of distances between embeddings of consecutive sentences for both human and LLM-generated text, calculated on a sample of size 1000 from BookSum dataset (no PCA applied to the embeddings in this case). A small but noticeable distribution shift between the two can be observed. Based on this, the interval $[0.4, 0.55]$ appears to be a reasonable choice for *SimMark* watermarking in this case. It is important to note that changes to the embedding representations, such as applying PCA or using a different embedding model, will lead to altered distance

Count Method	K	cosine-*SimMark*	Paraphrased cosine-*SimMark*	Euclidean-*SimMark*	Paraphrased Euclidean-*SimMark*
Soft Count	50	99.0 / 89.2 / 97.2	98.6 / 78.5 / 96.5	99.4 / 91.2 / 98.1	96.9 / 49.3 / 87.9
	150	99.6 / 95.7 / 98.8	**99.2** / 88.7 / **98.2**	**99.8** / 97.6 / 99.3	**97.3** / 67.8 / **90.4**
	250	**99.7** / **96.9** / 98.8	**99.2** / 90.3 / **98.2**	**99.8** / **98.5** / 99.2	97.2 / **72.3** / 88.9
	350	**99.7** / **96.9** / **98.9**	**99.2** / **90.4** / 98.1	**99.8** / **98.5** / **99.4**	97.2 / 71.1 / 88.9
Regular Count	∞	99.7 / 97.2 / 99.1	99.1 / 88.7 / 97.6	99.8 / 98.5 / 99.7	97.0 / 70.0 / 88.2

Table 4: Ablation study on the smoothness factor K in soft counting (Eq. (1)) using the RealNews dataset, with Pegasus as the paraphraser. Metrics reported include ROC-AUC ↑, TP@FP=1% ↑, and TP@FP=5% ↑, from left to right. The last row ($K = \infty$) corresponds to regular counting with a step function in the interval $[a, b]$. A smoothness factor of $K = 250$ provides a good balance between performance before and after paraphrase attacks for both cosine-*SimMark* and Euclidean-*SimMark*. Notably, while soft counting slightly reduces performance in the absence of paraphrasing, it demonstrates enhanced robustness against paraphrasing, yielding an increase across all metrics for Pegasus paraphraser and potentially larger gains against more advanced paraphrasers.

Dataset	Configuration	No paraphrase	Pegasus
RealNews	Cosine-*SimMark* (No PCA)	**99.7** / 96.9 / 98.8	**99.2** / **90.3** / **98.2**
	Cosine-*SimMark* (PCA)	99.6 / 96.9 / **99.1**	92.1 / 33.8 / 71.2
	Euclidean-*SimMark* (No PCA)	99.4 / 92.6 / 98.4	90.5 / 19.7 / 58.0
	Euclidean-*SimMark* (PCA)	**99.8** / **98.5** / **99.2**	**97.2** / **72.3** / **88.9**
BookSum	Cosine-*SimMark* (No PCA)	99.8 / 98.8 / 99.5	**99.5** / **93.3** / **98.5**
	Cosine-*SimMark* (PCA)	**100** / **99.9** / **99.9**	98.7 / 87.3 / 95.1
	Euclidean-*SimMark* (No PCA)	**100** / **100** / **100**	**98.8** / **82.6** / **94.9**
	Euclidean-*SimMark* (PCA)	99.9 / 99.3 / 99.5	97.4 / 69.8 / 88.6
Reddit-TIFU	Cosine-*SimMark* (No PCA)	99.1 / 96.3 / 97.6	**98.9** / **94.5** / **96.4**
	Cosine-*SimMark* (PCA)	**99.7** / **98.8** / **99.3**	96.6 / 78.9 / 89.3
	Euclidean-*SimMark* (No PCA)	99.6 / 98.1 / 99.1	96.7 / 72.6 / 90.0
	Euclidean-*SimMark* (PCA)	**99.8** / **98.7** / **99.2**	**99.0** / **94.7** / **97.6**

Table 5: Ablation study on the impact of applying PCA to embeddings across three datasets. Metrics reported include ROC-AUC ↑, TP@FP=1% ↑, and TP@FP=5%↑, respectively, from left to right. Higher values indicate better performance across all metrics. For cosine-*SimMark*, not applying PCA yields better results, while for Euclidean-*SimMark*, applying PCA improves performance except on the BookSum dataset.

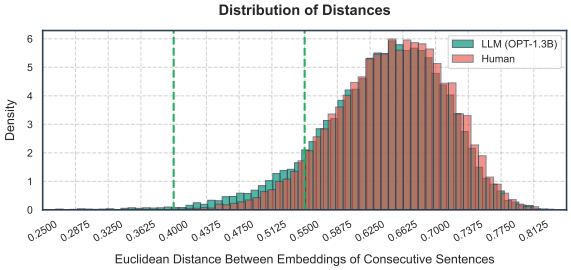

Figure 7: Distribution of Euclidean distances between embeddings of consecutive sentences for human-written and LLM-generated text on BookSum dataset, generated using OPT-1.3B. The figure demonstrates that the interval $[0.4, 0.55]$ is a reasonable choice for Euclidean-*SimMark* in this case, though it is not necessarily the only viable option.

distributions. Consequently, the interval must be adjusted accordingly to maintain optimal performance. For instance, if PCA is applied, the interval $[0.28, 0.36]$ is suitable. Similarly, if we plot the figure for when cosine similarity is used instead of Euclidean distance, intervals $[0.81, 0.94]$ and $[0.68, 0.76]$ are good candidates for cases with and without PCA, respectively. This variability in the distance distributions may also strengthen the algorithm's resistance to reverse engineering. Selecting the optimal interval $[a, b]$ is a critical step in achieving a robust and reliable watermarking with *SimMark*. In general, selecting an optimal interval involves balancing low FP rates, high TP rates, and robustness against paraphrasing attacks. It is often beneficial to choose intervals toward the tails of the distribution rather than around the mean. Finally, further exploration of dynamic interval selection mechanisms could enhance *SimMark*'s robustness.

I Computing Threshold β for *soft*-z-test

Recall that a text is classified as LLM-generated when $z_{\text{soft}} > \beta$, and as human-written otherwise. z_{soft} is the statistic used in the statistical test described in Eq. (2). To determine the threshold β that limits the FP rate to 1% or 5%, we first need to estimate p_0, the probability that the consecutive embeddings' similarity or distance falls within the predefined interval $[a, b]$. This value of p_0 is a key component in calculating the z_{soft}, as it represents the proportion of valid sentences in human-written text under the given interval. p_0 serves as an indicator of how frequently valid sentences are expected to occur in human-authored text.

To compute p_0, we analyze the distribution of similarities (or distances) using a histogram approach, such as the one depicted in Figure 7. Specifically, we employ a binning technique to approximate the area under the curve of distribution in the interval $[a, b]$. The process involves dividing the entire range of distances or similarities into a fixed number of bins—1000 bins in our implemen-

tation. Each bin represents a small segment of the range, and the histogram is used to calculate the proportion of samples that fall within the interval $[a, b]$.

Mathematically, p_0 is estimated as:

$$p_0 = \frac{\text{Number of samples in bins corresponding to } [a, b]}{\text{Size of the dataset}},$$

Once p_0 is estimated, the detection threshold β is determined by iterating over a range of possible values, typically from -10 to 10, to find the one that results in the desired false positive rate. Specifically, the threshold is chosen such that the proportion of human-written texts misclassified as LLM-generated matches the target FP rate (e.g., 1% or 5%).

It is worth noting that the distribution of similarities or distances may vary depending on different factors such as embedding model, and similarity measure (e.g., cosine or Euclidean). As a result, p_0 and therefore β are determined programmatically during the detection to ensure reliable performance of the watermarking algorithm.

J Analysis of Sampling Efficiency

The average number of samples required to generate a valid sentence is influenced by the chosen interval $[a, b]$. To provide further insights into this relationship, we estimated the area under the curve (AUC) of the embedding similarity distribution for an unwatermarked LLM. For instance, for the interval $[0.68, 0.76]$ (for cosine-*SimMark*), the estimated AUC is approximately 0.194.

Using the mean of the geometric distribution, which is given by $\frac{1}{p}$, where p is the probability of success (in this case, the probability of falling within the interval), this translates to an expected average of $\frac{1}{0.194} \approx 5.1$ samples per valid sentence. This estimate is on par with the experimental results reported in the main paper. The AUC estimates were computed using the binning technique, as described in Appendix I.

This analysis underscores the importance of carefully selecting the interval $[a, b]$, as narrower intervals may increase the number of required samples, leading to reduced sampling efficiency but better performance, while broader intervals may compromise the effectiveness of the watermark. By understanding the interplay between the interval choice and sampling efficiency, we can better optimize *SimMark*'s performance.

K Prompts Used with GPT-3.5-Turbo

Table 6 presents the prompts we used to obtain paraphrases using GPT-3.5-Turbo (accessed via OpenAI API[2]) for both regular paraphrasing and more aggressive bigram paraphrasing attacks[3]. By using the same prompts as Hou et al. (2024a), we ensured that our results were directly comparable to those extracted from their paper, maintaining consistency in evaluation methodology.

Prompt for Regular Attack
`Previous context: {context} \n Current sentence to paraphrase: {sent}`
Prompt for Bigram Attack
`Previous context: {context} \n Paraphrase in {num_beams} different ways and return a numbered list: {sent}`

Table 6: Prompts used to generate paraphrases with GPT-3.5-Turbo for regular and bigram attacks. These are the same prompts used by Hou et al. (2024a) for consistent and comparable evaluation. Here, `sent` represents the target sentence to rephrase, `context` includes all preceding sentences, and `num_beams` specifies the number of paraphrases generated for the bigram attack. A higher `num_beams` value indicates a more aggressive attack. Following Hou et al. (2024a), we set it to 10 to have 10 rephrases of each sentence.

[2] https://platform.openai.com/docs/api-reference

[3] Used "`gpt-3.5-turbo-16k`" model.