Adaptive Text Watermark for Large Language Models

Yepeng Liu [1] **Yuheng Bu** [1]

Abstract

The advancement of Large Language Models (LLMs) has led to increasing concerns about the misuse of AI-generated text, and watermarking LLM-generated text has emerged as a potential solution. However, it is challenging to generate high-quality watermarked text while maintaining robustness, security, and the ability to detect watermarks without prior knowledge of the prompt and model. This paper proposes an adaptive text watermarking strategy to address such a challenge. To improve the text quality and maintain robustness, we adaptively add watermarking to token distributions with high entropy measured by an auxiliary model and keep the low-entropy token distributions untouched. For the sake of security and to further minimize the watermark's impact on text quality, instead of using a fixed green/red list generated from a random secret key, which can be vulnerable to decryption and forgery, we adaptively scale up the output logits based on the semantic embedding of previously generated text using a well designed semantic mapping model. Our experiments involving various LLMs demonstrate that our approach achieves comparable robustness performance to existing watermark methods. Additionally, the text generated by our method has perplexity comparable to that of *un-watermarked* LLMs while maintaining sufficient security.[1]

1. Introduction

The rapid development of Large Language Models (LLMs) (Zhang et al., 2022; Touvron et al., 2023) has ushered in transformative possibilities but also brings forth potential challenges. One primary challenge lies in the misuse of AI-generated text for malicious purposes, such as spreading disinformation, generating fake news, or engaging in plagiarism. The sheer volume and quality of AI-generated text make it increasingly difficult to discern between human and AI-generated content. As a solution, implementing a text watermark on LLMs proves to be an effective method for detecting AI-generated text.

Specifically, the watermarking technique encodes certain patterns into the text generated by LLMs, which are imperceptible to humans but can be detected by the algorithm. This is usually achieved by introducing well-designed perturbations to the original logits or using a special sampling method. Kirchenbauer et al. (2023a) design a watermark for LLM by randomly separating the vocabulary into 'green' and 'red' lists and increasing the frequency of tokens from the 'green' list. The previously generated token is used as a secret key to determine the 'green/red' list for the next token. This method achieves good detection performance when the watermarked text remains unmodified. However, after modifications such as word replacement and paraphrasing, this method will fail as the 'green/red' list cannot be reliably recovered from the modified text. As it is common for individuals to alter AI-generated text rather than use it directly, improving the *robustness* of the watermark becomes an essential requirement.

To improve the robustness of LLM watermarking, Zhao et al. (2023) propose to use a fixed 'green/red' list for each token, which is generated from a random secret key shared with the detector. Both empirical and theoretical evidence demonstrate that this method substantially enhances robustness. However, more recently, Sadasivan et al. (2023) introduced a spoofing attack specifically designed to deduce hidden watermark patterns by analyzing the token frequency, thereby enabling the forgery of watermarked text. Such an attack offers a method for adversaries to circumvent the watermark, potentially damaging the reputation of the watermarking system. Regrettably, it has been shown that the algorithm proposed in Zhao et al. (2023) is vulnerable to spoofing attack, and one can infer the fixed 'green' token list and forge the watermarked text.

The aforementioned papers enhance the frequency of tokens in the 'green' list during generation by uniformly adding a fixed value to the logits of those 'green' tokens. This way

---

[1] University of Florida. Correspondence to: Yepeng Liu <yepeng.liu@ufl.edu>, Yuheng Bu <buyuheng@ufl.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1] Our code is available at `https://github.com/yepengliu/adaptive-text-watermark.git`.

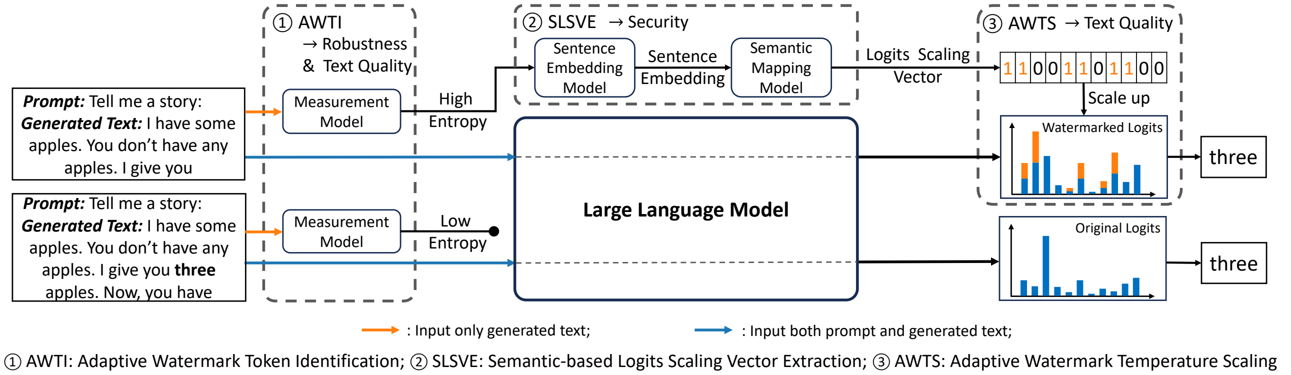

① AWTI
→ Robustness
& Text Quality
② SLSVE → Security
③ AWTS → Text Quality
Prompt: Tell me a story:
Generated Text: I have some apples. You don't have any apples. I give you
Prompt: Tell me a story:
Generated Text: I have some apples. You don't have any apples. I give you three apples. Now, you have
Measurement Model
Measurement Model
High Entropy
Low Entropy
Sentence Embedding Model
Sentence Embedding
Semantic Mapping Model
Logits Scaling Vector
1 1 0 0 1 1 0 1 1 0 0
Scale up
Watermarked Logits
Original Logits
Large Language Model
three
three
: Input only generated text;
: Input both prompt and generated text;
① AWTI: Adaptive Watermark Token Identification; ② SLSVE: Semantic-based Logits Scaling Vector Extraction; ③ AWTS: Adaptive Watermark Temperature Scaling

Figure 1: Workflow of the proposed Adaptive Watermark. Our Adaptive Watermark method will assess the entropy of the distribution for all previously generated tokens and add watermarks only to high-entropy tokens. This figure illustrates two cases in the text generation process. For the first example, the distribution of the next token has high entropy as measured by AWTI, indicating high uncertainty. Then, the SLSVE module will extract the logits scaling vector based on the semantics of this text. Subsequently, we apply AWTS to perturb the sampling distribution. As for the second text, it results in low entropy measured by AWTI, suggesting a low uncertainty for the next token. Then, the next token will be directly sampled from the original probability distribution.

of perturbing the distribution may change the distribution significantly, deteriorating the quality of the watermarked text. To tackle this issue, Kuditipudi et al. (2023) presents a distortion-free watermarking method that maintains the original probability distribution of LLMs and embeds the watermark during the token sampling process. However, such a distortion-free watermarking lacks sufficient robustness against modifications, like paraphrasing.

Inspired by existing research, designing an effective watermark algorithm requires us to consider at least the following three factors comprehensively. (1) **Robustness:** The watermark should remain detectable after modifications without significantly changing the semantics of the watermarked text. (2) **Security:** The watermark should be resilient against efforts to reverse-engineer or decryption without authorization. (3) **Text quality:** LLMs-generated watermarked text should preserve a level of perplexity comparable to the unwatermarked text. Among existing works, achieving high-quality text often comes with the trade-off of robustness, and prioritizing strong robustness can potentially compromise security. Therefore, designing a watermarking scheme that simultaneously guarantees robustness, security, and text quality poses a significant challenge.

In this work, our goal is to provide a holistic approach that integrates robustness and security with the production of high-quality text. Additionally, we ensure the watermark can be detected in an agnostic manner, meaning that the watermark detection process is independent of the original LLMs and prompts. Our watermark strategy consists of three components, as shown in Figure 1.

- To minimize the impact of the watermark on the text quality while maintaining the robustness, we propose the Adaptive Watermark Token Identification (AWTI) so that the watermarking is added adaptively to high-entropy token distributions and leaves the low-entropy token distributions intact. This method employs an auxiliary Measurement Model to assess the entropy of each token, ensuring that the entropy can be reliably estimated without access to the original LLMs and prompts.
- To further enhance the security of our method, we introduce the Semantic-based Logits Scaling Vector Extraction (SLSVE). Our logits scaling vector is obtained by passing the semantics of the generated text (using a pre-trained Sentence Embedding Model) to a Semantic Mapping Model, which is hard to reverse-engineer.
- We propose the Adaptive Watermark Temperature Scaling (AWTS), which proportionally scales up the original logits using the extracted logits scaling vector. Through this approach, the perturbation is applied by scaling the *temperature* of the distribution, which further reduces the influence on the text quality due to watermarking.
- In the detection process, the AWTI is first applied to the text under analysis to identify potential *watermarked tokens*. After this, the SLSVE is employed to extract the logits scaling vectors and calculate the detection score.

We conduct extensive experiments over various language models (OPT-6.7B (Zhang et al., 2022), GPT-J-6B (Wang & Komatsuzaki, 2021), and Mistral-7B (Jiang et al., 2023)) to show the effectiveness of our method. We employ perplexity as a metric to evaluate text quality, and our results indicate that the quality of the watermarked text remains comparable to that of the *un-watermarked* text. To show the robustness of our method, we paraphrase the watermarked text using GPT3.5 and DIPPER (Krishna et al., 2023). The outcomes are on par with those achieved by Zhao et al. (2023), which is the most robust method in the literature. The spoofing attack is used to illustrate that our watermark is hard to decrypt. Lastly, our detection process is agnostic and does not rely on any original prompt and LLMs.

2. Related Work

As LLMs evolve, the need to distinguish AI-generated text from human-written text is crucial for authenticity, accountability, and intellectual property protection. Text watermarking, embedding hidden markers in text, emerges as a promising solution for this challenge. There are currently two main types of watermark methods in use: generated text watermarking and LLMs watermarking.

Generated text watermarking primarily operates on existing text, altering aspects like format (Brassil et al., 1995; Por et al., 2012; Sato et al., 2023; Rizzo et al., 2016), lexical choices (Yoo et al., 2023; Yang et al., 2023; Munyer & Zhong, 2023; Yang et al., 2022), or syntax (Atallah et al., 2001; Topkara et al., 2006; Meral et al., 2009), while preserving the underlying semantics. EASYMARK (Sato et al., 2023), a format-based watermarking, utilizes subtle changes in Unicode characters and spacing to embed watermarks in text. For word-level modifications, Yang et al. (2022), based on context-aware lexical substitution, utilizes BERT to identify semantically suitable synonym candidates for watermarking. Meral et al. (2009) introduces a morphosyntax-based natural language watermarking scheme that uses syntactic tree diagrams to analyze the structure and functional dependencies of text. In addition to the aforementioned methods that primarily rely on specific rules and might lead to unnatural modifications, some neural network-based approaches (Zhang et al., 2023b; Abdelnabi & Fritz, 2021) add watermarking by regenerating the text with specific message signatures using a pre-trained language model, while keeping the semantics unchanged.

LLMs watermarking. Rather than adding a watermark to existing text, an increasing number of studies are focusing on embedding watermarks directly into the text during the LLM generation (Wang et al., 2023; Fairoze et al., 2023; Hu et al., 2023; Lee et al., 2023; Huo et al., 2024; Fu et al., 2024; Zhao et al., 2024; Zhang et al., 2023a; Tu et al., 2023).

Some research efforts are dedicated to enhancing the *robustness* of watermarks (Kirchenbauer et al., 2023a;b; Liu et al., 2023b; Zhao et al., 2023; Ren et al., 2023), aiming to ensure the persistence of watermarking even after exposure to various attacks. Zhao et al. (2023) introduce a provable robust watermark using a fixed vocabulary 'green/red' list generated using a shared random key. Ren et al. (2023) utilize the semantics of the generated text as a hash key to separate the vocabulary, which is achieved by discretizing the continuous embedding space. Hou et al. (2023) propose a sentence-level semantic watermarking technique that utilizes the semantics of previously generated sentences to divide the semantic space into "valid" and "blocked" regions and generates new sentences through rejection sampling until its semantics reside within the "valid" region.

Some works focus on enhancing the *security* of watermarks (Wu et al., 2023; Christ et al., 2023; Liu et al., 2023a), aiming to make them difficult to forge. Liu et al. (2023b) proposed a semantic invariant watermark that balances robustness and security. However, this method relies on the original prompts for detection, which are often impractical to access. Additionally, the way it perturbs the distribution results in noticeably higher text perplexity compared to un-watermarked text.

All the watermarks mentioned above are inserted into the text by manipulating the logits produced by LLMs, which inevitably introduces some distortion in text generation. Some studies Kuditipudi et al. (2023); Christ et al. (2023) introduce watermarks using specially crafted sampling methods. These methods are designed to ensure that the watermark has a minimal distortion on the original LLM distributions to improve *text quality*.

Our work focuses on the LLMs watermarking. Previous works in this field have significantly improved the robustness, text quality, and security aspects of watermarks in a separate manner. However, it remains challenging to consider all these aspects simultaneously. Therefore, we intend to propose a holistic watermark approach that achieves strong robustness and high-quality text generation while also being difficult to forge.

Notations. Given a Language Model (LM), denote $\mathcal{V}$ as the vocabulary of LM, and denote $|\mathcal{V}|$ as the size of vocabulary. For a input prompt $R = \{R^{(1)}, R^{(2)}, ..., R^{(N)}\}$, the LM will generate a sequence of tokens $S = \{S^{(1)}, S^{(2)}, ..., S^{(T)}\}$ based on the prompt. We denote $S_{0:t} = \{S^{(1)}, ..., S^{(t)}\}$ and $S_0 = \emptyset$, and $r^{(n)}$ and $s^{(t)}$ are concrete tokens in the vocabulary, i.e., $r^{(n)}$, $s^{(t)} \in \mathcal{V}$. As an auto-regressive model, LM will predict the probability distribution $P_{LM}(\cdot|R, S_{0:t})$ over each token in $\mathcal{V}$ at time step $t+1$ based on the prompt and the preceding sequence, and sample the next token based on this probability distribution until the token of termination is sampled.

3. Adaptive Watermark Generation

In this section, we present our adaptive watermarking technique designed to minimize the impact on text quality, enhance security, and maintain robustness simultaneously. Section 3.1 introduces Adaptive Watermark Token Identification to ensure watermark robustness while minimizing the impact of text quality in generated content. The Semantic-based Logits Scaling Vector Extraction, as discussed in Section 3.2, generates different logits scaling vectors based on the semantics of the generated text, which adaptively perturb the distribution of the LLM. In Section 3.3, we propose a new distribution perturbation method to inject watermarking, further improving watermarked text quality.

3.1. Adaptive Watermark Token Identification (AWTI)

The Adaptive Watermark Token Identification (see Figure 1-①) is designed to minimize the impact of the watermark on the text quality while maintaining its robustness by reducing the percentage of perturbed tokens during the text generation process. The watermarking is added in an adaptive manner, i.e., only the distribution of those tokens flagged by our auxiliary language model as having high entropy will be perturbed by our watermarking method.

Specifically, given a prompt R, the LLM will generate a logits vector $l^{(t+1)}$ over all the tokens in $\mathcal{V}$ at time step $t+1$. The *softmax* function is then applied to convert the logits into a probability distribution $P_{LM}(\cdot|R, S_{0:t})$, where the probability of the k-th token in the $\mathcal{V}$ is $p_k^{(t+1)} = \exp(l_k^{(t+1)})/\sum_{i=1}^{|\mathcal{V}|}\exp(l_i^{(t+1)})$. The LLM will sample a token from this probability distribution using methods such as Top-K sampling or Top-p sampling. As shown in the example of Figure 1, the sampling distribution may vary based on the context of the generated text at different steps.

Shannon Entropy (Shannon, 1948) measures the uncertainty of a distribution, which is defined as $H(P_X) \triangleq -\sum_{x\in\mathcal{X}} P_X(x)\log P_X(x)$, where P_X is a discrete probability distribution defined over the space $\mathcal{X}$, and $0 \le H(P_X) \le \log|\mathcal{X}|$. When the entropy is low, the distribution becomes more concentrated on specific tokens. Conversely, when the entropy is high, the distribution becomes more spread, indicating larger uncertainty. A high entropy distribution in LLMs means that the original distribution offers multiple reasonable choices during text generation. Thus, inserting certain patterns by only perturbing the high entropy distribution will minimize the impact of the sampling process, ensuring that the generated tokens remain contextually appropriate.

We note that Lee et al. (2023) also discusses the strategy of perturbing only the token with high entropy distributions for code generation. Their approach, however, requires access to both the original LLM and prompts, rendering it impractical for watermarking detection. Additionally, utilizing the original LLM for detection is not only time-consuming but also incurs substantial computational costs.

To ensure reliable entropy estimation during the text generation and detection process without relying on the watermarked LLM or the original prompts, we use an auxiliary Measurement Model (MM) to compute the entropy of the generated text at each time step. The entropy measured with MM is expressed as

$$H\left(P_{MM}(\cdot|S_{0:t})\right) = -\sum_{i=1}^{|\mathcal{V}|} p_i^{(t+1)} \log p_i^{(t+1)}. \qquad (1)$$

The MM can be a language model that is significantly smaller in scale compared to the watermarked LLMs, such as GPT-2 (Radford et al., 2019) or OPT-350M (Zhang et al., 2022). We use a predefined entropy threshold, denoted as α, and our watermark only perturbs those distributions with $H\left(P_{MM}(\cdot|S_{0:t})\right) \ge \alpha$. We refer to the token sampled from the intact distribution as *un-watermarked token* and the token sampled from the perturbed distribution as *watermarked token*. As the threshold α increases, two effects become evident. Firstly, there will be more un-watermarked tokens during the generation process. Secondly, the impact of each watermarked token diminishes, mainly due to the higher inherent uncertainty in their distributions. As a result, both effects will enhance the quality of the watermarked text.

It is important to acknowledge that the entropy computed by the auxiliary MM may differ from that of the original watermarked LLM, but this discrepancy does not affect the efficacy of our watermarking. The auxiliary MM is employed primarily to assist in identifying potential watermarked tokens during the detection phase, eliminating the need for the original watermarked LLM and prompt. As long as we can identify watermarked tokens in the detection phase using such entropy estimates, the performance will not be affected. In addition, due to the fact that only watermarked tokens are involved in the detection process as described in Section 4, adaptive watermarking may also benefit robustness by filtering out un-watermarked tokens.

3.2. Semantic-based Logits Scaling Vector Extraction

The Semantic-based Logits Scaling Vector Extraction (see Figure 1-②) is primarily developed to ensure the security of the watermark. The intuition of this approach is to make sure that the perturbation in the distribution is determined by the semantics of the generated text using a well-designed neural network so that the inserted watermarking cannot be easily decrypted by attackers.

Previous work uses the hash of the preceding token or a fixed random key to generate the ‘green’ and ‘red’ lists. However, such lists can be recovered by analyzing the token frequency in watermarked text using statistical approaches, e.g., spoofing attacks, which enable adversaries to forge watermarks. In contrast, the proposed SLSVE will adaptively generate the logits scaling vector to perturb the distribution based on the semantics of the generated text using a trainable neural network. The complex high-dimensional semantic space and neural networks further increase the complexity of the watermark, making it more difficult for adversaries to decrypt.

In particular, for tokens with distributions satisfying $H(P_{MM}(\cdot|S_{0:t-1})) \ge \alpha$ at step t, we proceed to extract logits scaling vector based on the semantics of generated text. First, we obtain the semantic embedding using a Sentence Embedding (SE) Model, i.e., $u^{(t)}(S) = \mathrm{SE}(S_{0:t-1})$,

Algorithm 1 Semantic-based Logits Scaling Vector Extraction (SLSVE)

```
Input: Sentence Embedding Model (SE), Semantic Mapping Model (SM), Input sentence $S$.
$u^{(t)}(S) = \mathrm{SE}(S_{0:t-1})$
$v^{(t)}(S) = \mathrm{SM}(u^{(t)}(S); \theta)$
$\hat{v}^{(t)}(S) = \left[\mathbb{1}\{v_1^{(t)}(S) > 0\}, ..., \mathbb{1}\{v_{|\mathcal{V}|}^{(t)}(S) > 0\}\right]$
Output: logits scaling vector $\hat{v}^{(t)}$.
```

where $u^{(t)} \in \mathbb{R}^L$. We employ a pre-trained LM as the Sentence Embedding Model, e.g., Sentence-BERT (Reimers & Gurevych, 2019), to extract the sentence embedding. Then, we train a Semantic Mapping (SM) Model to convert the sentence embedding into a logits scaling vector, denoted as $v^{(t)}(S) = \mathrm{SM}(u^{(t)}(S); \theta)$, where $v^{(t)} \in \mathbb{R}^{|\mathcal{V}|}$ and θ denotes the parameters of the Semantic Mapping Model.

Specifically, we train a feedforward neural network with multiple residual connections to ensure that the logits scaling vector satisfies the following properties.

(1) **Smoothness with respect to semantics:** As we need to recover the logits scaling vector to perform detection even with modifications in the watermarked text, the Semantic Mapping Model needs to be insensitive to small changes in semantics to improve the robustness of the watermark. In order to make the logits scaling vector align with the change in semantic embedding, we want to make sure that if two different sentences S and S' are close in embedding space, they will have similar logits scaling vector v and v'. We will minimize the loss $\left|D(u(S), u(S')) - D(v(S), v(S'))\right|$, where we use the Euclidean distance $D(a, b) = \sqrt{\sum_{i=1}(a_i - b_i)^2}$ to measure the distance between two sentence embeddings.

(2) **Uniform Perturbation:** To make the watermark easy to detect, we want to ensure the logits scaling vector increases the probability of roughly half of the tokens in the vocabulary. For any sentence S within the training dataset $\mathcal{T}$, we ensure that $\sum_{i \in \mathcal{V}} \mathrm{sign}\big(v_i(S)\big) = 0$. Therefore, an equal rate of positive and negative entries in $v_i(S)$.

(3) **Unbiased token preference:** The logits scaling vector should be designed to prevent consistent bias towards specific tokens in the vocabulary. This ensures that specific tokens do not always exhibit a high probability in distributions across various semantic contexts, preventing their frequent recurrence in the watermarked text and improve security. Therefore, for any token i in the vocabulary $\mathcal{V}$, we will ensure $\sum_{S \in \mathcal{T}} \mathrm{sign}\big(v_i(S)\big) = 0$.

Additionally, during the training of the Semantic Mapping Model, we ensure that the logits scaling vector remains stable against semantic variations using the following three implementation tricks. This further guarantees that the watermark's robustness remains unaffected by semantic variations within a specific range. (i) Due to the high-dimensional nature of sentence embeddings, experiments on the original training dataset revealed that most sentence embeddings are essentially orthogonal to each other. To improve the model's capability in effectively handling slight semantic perturbations, we enlarged our training dataset by performing data augmentation, including paraphrasing, expanding, and shortening the sentences in the original training dataset. (ii) With the augmented data mentioned above, a contrastive loss term is applied during training to minimize the distance between the sentence ($S \in \mathcal{T}$) and its augmented counterparts($\widetilde{S} \in \widetilde{\mathcal{T}}$), i.e., $\min \sum_{S \in \mathcal{T}, \widetilde{S} \in \widetilde{\mathcal{T}}} \left|D(v(S), v(\widetilde{S}))\right|$. This contrastive term improves the stability of the model when a modified sentence is presented. (iii) We rescale the Euclidean distance between two sentence embeddings $d = D(u(S), u(S')) \in [L, U]$ to a wider range $d' \in [L', U']$ using a linear transformation, where $L' < L$ and $U' > U$. Rescaling makes the distance between two sentence embeddings smaller when they are close to each other and increases the distance when they are originally farther apart in the embedding space.

Therefore, considering all the desired properties, the loss function of the Semantic Mapping Model is:

$$\begin{aligned}\mathcal{L}(\widetilde{\mathcal{T}}, \theta) = & \sum_{S, S' \in \widetilde{\mathcal{T}}} \left|D(u(S), u(S')) - D(v(S), v(S'))\right| \\ & + \sum_{S \in \widetilde{\mathcal{T}}} \Big|\sum_{i \in |\mathcal{V}|} \mathrm{sign}(v_i(S))\Big| + \sum_{i \in |\mathcal{V}|} \Big|\sum_{S \in \widetilde{\mathcal{T}}} \mathrm{sign}(v_i(S))\Big| \\ & + \sum_{S \in \mathcal{T}, \widetilde{S} \in \widetilde{\mathcal{T}}} \left|D(v(S), v(\widetilde{S}))\right|. \end{aligned} \tag{2}$$

After getting the logits scaling vector $(v^{(t)})$ using the trained model, we apply the indicator function to transfer every entry in $v^{(t)}$ to either 0 or 1, i.e., $\hat{v}_i^{(t)}(S) = \mathbb{1}\{v_i^{(t)}(S) > 0\}$.

3.3. Adaptive Watermark Temperature Scaling (AWTS)

To further minimize the impact of perturbation on the original probability distribution, we introduce the Adaptive Watermark Temperature Scaling method (see Figure 1-③). This technique enhances the consistency of the perturbed distribution with the original distribution by adaptively adjusting the *temperature* of the distribution.

Specifically, we add watermark to the original LLM logits $l^{(t)}$ using temperature scaling:

$$\hat{l}^{(t)} = l^{(t)} \cdot (1 + \delta \cdot \hat{v}^{(t)}), \tag{3}$$

where $\delta > 0$ controls the strength of the watermark. In this way, we adaptively perturb the logit of each token by proportionally scaling up a factor of $(1 + \delta \cdot \hat{v}^{(t)})$ using

Algorithm 2 Adaptive Watermark Generation

```
Input: Language model $P_{LM}$, measurement model $P_{MM}$,
    prompt $R$, watermark strength $\delta$, opening sentence $\hat{S}$,
    entropy threshold $\alpha$, measure threshold $M$.
for $t = 1, 2, ..., T$ do
    if $t \leq M$ then
        $\hat{v}^{(t)} = \mathrm{SLSVE}(\hat{S})$
        $q_k^{(t)} = \frac{\exp(l_k^{(t)} \cdot (1 + \delta \cdot \hat{v}_k^{(t)}))}{\sum_{i \in \mathcal{V}} \exp(l_i^{(t)} \cdot (1 + \delta \cdot \hat{v}_i^{(t)}))}$
        $s^{(t)} \sim q^{(t)}$.
    else
        if $H(P_{MM}(\cdot|S_{0:t-1}) \geq \alpha$ then
            $\hat{v}^{(t)} = \mathrm{SLSVE}(S_{0:t-1})$
            $q_k^{(t)} = \frac{\exp(l_k^{(t)} \cdot (1 + \delta \cdot \hat{v}_k^{(t)}))}{\sum_{i \in \mathcal{V}} \exp(l_i^{(t)} \cdot (1 + \delta \cdot \hat{v}_i^{(t)}))}$
            $s^{(t)} \sim q^{(t)}$.
        else
            $p_k^{(t)} = \frac{\exp(l_k^{(t)})}{\sum_{i \in \mathcal{V}} \exp(l_i^{(t)})}$,
            $s^{(t)} \sim p^{(t)}$.
        end if
    end if
end for
Output: watermarked sequence $s^{(1)}, s^{(2)}, ..., s^{(T)}$.
```

the obtained SLSVE. If the original logit $l_i^{(t)}$ is small, the $\hat{l}_i^{(t)}$ after the manipulation remains small, indicating that our watermark slightly perturbs these logits. Temperature scaling mainly affects tokens with large original logits $l_i^{(t)}$, causing the corresponding $\hat{l}_i^{(t)}$ to increase more significantly. In contrast to uniformly increasing the logits of tokens in the 'green list' as in previous work (Kirchenbauer et al., 2023a), our approach inserts a watermark by modulating the *temperature* of the logits. Therefore, our watermark strength adaptively depends on the original distribution, which will make the generated watermarked text more coherent. As we will show in Figure 4, AWTS also contributes to the improvement of text quality.

Algorithm 2 describes the entire procedure of the proposed Adaptive Watermark algorithm. At the beginning of the text generation process, only a limited number of tokens can be used for extracting semantic embeddings, which makes the detection of the watermark unstable under attack. To maintain the stability and robustness of the watermark, we set a secret opening sentence $\hat{S}$. When the number of generated tokens is smaller than a pre-determined measure threshold M, we extract the semantic embedding from the opening sentence and apply SLSVE.

4. Adaptive Watermark Detection

In this section, we propose to approximate the Likelihood Ratio Test (LRT) to detect the watermark. Consider the following null hypothesis H_0 and the alternative H_1,

$$
\begin{aligned}
H_0 &: \text{The candidate text is not watermarked.} \\
H_1 &: \text{The candidate text is watermarked.}
\end{aligned}
$$

Given a sequence of tokens $S = \{S^{(1)}, S^{(2)}, ..., S^{(T)}\}$, we input the sequence to MM to calculate the entropy and select the potential *watermarked token* at each time step, following the same process as the AWTI described in Section 3.1. Denote the set of potential watermarked token as $\mathcal{W}$, i.e., for all $s^{(t)} \in \mathcal{W}$, we have $H(P_{MM}(\cdot|S_{0:t-1})) \geq \alpha$. For any $s^{(t)} \in \mathcal{W}$, if it is sampled from the perturbed distribution, the distribution would be $P_{H_1}(\cdot|R, S_{0:t})$. Conversely, if it is sampled from the original distribution, the distribution would be $P_{H_0}(\cdot|R, S_{0:t})$. Subsequently, the logits scaling vector is extracted utilizing both the Semantic Embedding Model and the Semantic Mapping Model, as detailed in Section 3.2. As we perturb the distribution with temperature scaling, the likelihood ratio can be written as

$$
\begin{aligned}
&\log \frac{P_{H_1}(S^{(t)} = k|R, S_{0:t-1})}{P_{H_0}(S^{(t)} = k|R, S_{0:t-1})} \\
=\ &\log \frac{e^{l_k \cdot (1+\delta \cdot \hat{v}_k)}}{\sum_i e^{l_i \cdot (1+\delta \cdot \hat{v}_i)}} + \log \frac{e^{l_k}}{\sum_i e^{l_i}} \\
=\ &l_k \cdot (\delta \cdot \hat{v}_k) + \log \frac{\sum_i e^{l_i}}{\sum_i e^{l_i \cdot (1+\delta \cdot \hat{v}_i)}} \\
\approx\ &l_k \cdot (\delta \cdot \hat{v}_k),
\end{aligned} \tag{4}
$$

where we omitted the superscript of time t, and in the last step, we approximated the ratio of the normalization factor in the softmax function by 1.

In the detection process, the original logits l_k are unknown, and our focus is only on identifying which tokens in the logits have been scaled up. For this purpose, we simply use $\delta \cdot \hat{v}_k$ as the test score to facilitate detection. Therefore, the test score is given by

$$
\mathrm{Score}(S = \{s^{(1)}, ..., s^{(T)}\}) = \frac{\sum_{s^{(t)} \in \mathcal{W}} \delta \cdot \hat{v}_{s^{(t)}}^{(t)}}{|\mathcal{W}|}, \tag{5}
$$

where $|\mathcal{W}|$ represents the number of all potential watermarked tokens.

5. Experiments

5.1. Experiment Setting

Implementation Details. In our watermarking algorithm, the default hyperparameters are set as follows: $\alpha = 2$, $M = 50$. We use Sentence-Transformers (Reimers & Gurevych, 2019) to extract the sentence embedding. The

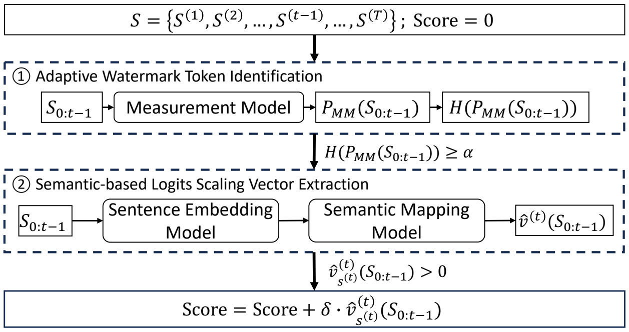

$S = \{S^{(1)}, S^{(2)}, \dots, S^{(t-1)}, \dots, S^{(T)}\}$; Score = 0
① Adaptive Watermark Token Identification
$S_{0:t-1}$
Measurement Model
$P_{MM}(S_{0:t-1})$
$H(P_{MM}(S_{0:t-1}))$
$H(P_{MM}(S_{0:t-1})) \geq \alpha$
② Semantic-based Logits Scaling Vector Extraction
$S_{0:t-1}$
Sentence Embedding Model
Semantic Mapping Model
$\hat{v}^{(t)}(S_{0:t-1})$
$\hat{v}^{(t)}_{s^{(t)}}(S_{0:t-1}) > 0$
Score = Score + $\delta \cdot \hat{v}^{(t)}_{s^{(t)}}(S_{0:t-1})$

Figure 2: Workflow of Adaptive Detection. This figure illustrates a single step during the detection process. At each time step, we will first check if the current token is a potential *watermarked token* by applying AWTI to estimate the entropy of the current distribution. For potential *watermarked token*, the preceding text will be used to extract the logits scaling vector through SLSVE. If the corresponding value of the potential *watermarked token* in the logits scaling vector is positive, it will be added to the total score.

specific model we used is `all-mpnet-base-v2`[2]. More details for the Semantic Mapping Model are provided in Appendix A. We apply our method to three different models: OPT-2.7B, OPT-6.7B (Zhang et al., 2022), GPT-J-6B (Wang & Komatsuzaki, 2021), and Mistral-7B (Jiang et al., 2023) (in Appendix B.1). At each time step, the next token is sampled from the distribution using a combination of Top-K and Top-p sampling methods, with $K = 50$ and $p = 0.9$. It is worth noting that the watermarking strength δ should be chosen differently for different LLMs based on their specific capabilities. For example, we set $\delta = 1.5$ for OPT-6.7B and set $\delta = 0.6$ for Mistral-7B.

Baselines. Our method is compared with existing methods, including KGW-0 (Zhao et al., 2023), KGW-1/KGW-2 (Kirchenbauer et al., 2023a), EXP-edit (Kuditipudi et al., 2023). In KWG-1/KWG-2, the numeral denotes the hashing of the previous 1 or 2 tokens, respectively. For KGW-0, KGW-1, and KGW-2, the 'green' list token percentage is set to be 0.5. Additional comparisons against two model non-agnostic baselines, i.e., SWEET (Lee et al., 2023) and SIR (Liu et al., 2023b), can be found in Appendix C.

Dataset and Prompt. To test the performance of our methods, the results on `realnewslike` subset in C4 dataset (Raffel et al., 2020) are provided here, and additional results on ELI5 (Fan et al., 2019) can be found in Appendix B.2. For C4, we use the first two sentences in each text as a prompt for LLMs and the subsequent 200 tokens as human-generated text. For ELI5, we use the question as a prompt and the corresponding answer as human-generated text. For both datasets, we generate 200 ± 30 tokens using the LLMs and prompts. The Semantic Mapping Model is trained using the Sentence-Compression dataset [3].

Evaluation Methods. Watermark detection performance is evaluated through the ROC-AUC value, the Best F1 Score, and the ROC curve, which illustrates the true positive rate (TPR) and false positive rate (FPR) at various thresholds. The TPR and FPR indicate the accurate identification of watermarked text and the incorrect recognition of human-written text as being watermarked. The perplexity of the text is calculated using LLama-13B (Touvron et al., 2023), measuring the quality of the text. In Appendices B.1 and B.2, we also use a more powerful LLM, GPT-3, to calculate the perplexity.

5.2. Robustness

To demonstrate the robustness of our watermark strategy, we perform a paraphrase attack on the watermarked text using two different paraphrasing methods: GPT-3.5 and DIPPER (Krishna et al., 2023). For GPT-3.5, we use the `gpt-3.5-turbo-instruct` version, and we employ the same prompt as used in Kirchenbauer et al. (2023b) to paraphrase the watermarked text. DIPPER is a paraphrase generation model to rewrite the text and evade the watermark detection, and the lexical diversity there is set to be 60.

The results are summarized in Table 1, which shows that all methods demonstrate effective performance when the watermarked texts remain unaltered. However, when the watermarked text is paraphrased using GPT-3.5 and DIPPER, the proposed adaptive watermarking method remains high ROC-AUC and Best F1 Score, surpassing those of the baselines across different language models. The corresponding ROC curves, showcasing the performance of different watermarking methods on various language models under paraphrasing attacks, can be found in Appendix B.3. Moreover, we evaluate the TPR scores at fixed 1% and 10% FPR as detailed in Appendix B.4.

5.3. Text Quality

The quality of the generated text is evaluated by perplexity. A lower perplexity value suggests better text quality. In Figure 3, we compare the perplexity of text generated by our method with human-written text, un-watermarked text (texts generated by the model without any watermarking), and other baseline methods. Our method has a negligible impact on the perplexity compared to the *un-watermarked* text. It also shows comparable results with EXP-edit, which is a distortion-free watermark method. The quality of text watermarked using our approach is clearly superior to that of KGW-0 and KGW-1. In Appendix B.5, we also report the repetition rate of watermarked text.

---

[2] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[3] https://huggingface.co/datasets/embedding-data/sentence-compression

Table 1: Performance evaluation of watermarked text under different conditions, including cases with no attack and paraphrase attacks.

Language Model	Setting	ROC-AUC					Best F1Score				
		KGW-0	KGW-1	KGW-2	EXP-edit	**Ours**	KGW-0	KGW-1	KGW-2	EXP-edit	**Ours**
OPT-2.7	No Attack	0.996	0.998	0.998	0.997	**1.000**	0.989	0.997	0.999	0.994	**1.000**
	GPT-3.5	0.971	0.881	0.768	0.919	**0.983**	0.916	0.818	0.719	0.871	**0.952**
	DIPPER	0.939	0.846	0.767	0.899	**0.960**	0.879	0.783	0.726	0.841	**0.902**
OPT-6.7	No Attack	0.995	0.995	**0.999**	0.993	0.996	0.997	0.997	0.998	0.995	**0.998**
	GPT-3.5	0.964	0.874	0.769	0.881	**0.981**	0.904	0.815	0.728	0.801	**0.932**
	DIPPER	0.936	0.842	0.795	0.883	**0.964**	0.867	0.782	0.743	0.807	**0.903**
GPT-J-6B	No Attack	**1.000**	0.997	1.000	0.997	0.998	0.995	0.993	**1.000**	0.995	0.996
	GPT-3.5	0.946	0.874	0.764	0.931	**0.966**	0.887	0.804	0.721	0.868	**0.913**
	DIPPER	0.923	0.850	0.803	0.960	**0.963**	0.855	0.783	0.753	0.898	**0.901**

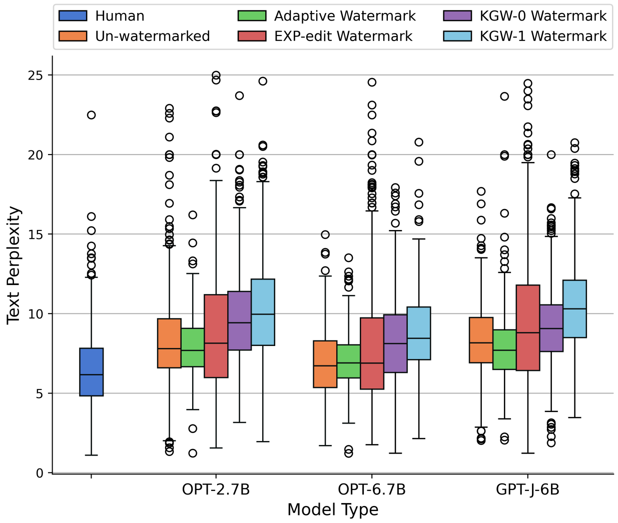


Figure 3: Comparison of text perplexity among human-written text, un-watermarked text, and texts using various watermark methods across different language models. For KGW-0 and KGW-1, the watermark strength and green list size are set as 2.0 and 0.5, respectively.

5.4. Security Results

The spoofing attack (Sadasivan et al., 2023) is used to evaluate the security of our method, which is an adversarial attack designed to produce non-watermarked text that can be mistakenly identified as AI-generated by the watermark detector. This is accomplished by frequently querying the watermarked language model and identifying the potential 'green' list as per a particular watermarking algorithm by analyzing token frequencies in the generated text.

Specifically, for KGW-1/KGW-2, the attacker seeks to uncover the green token list for any selected fixed prefix. This entails generating 5,000 sentences under the KGW-1 and examining the frequency of tokens among 181 common tokens (C). From C, the top 50 tokens with the highest frequency (H) are identified. For example, the process involves analyzing the occurrence frequency of tokens following 'the.' As the actual green token list (G) is determined using the token 'the' as a hash, the decryption rate is then computed based on the proportion of tokens in H that belong to G.

Table 2: Security performance for different watermark methods.

Language Model	Evaluation	KGW-0	KGW-1	KGW-2	Ours
OPT-6.7	Decryption Rate	0.963	0.912	0.723	0.571

For the proposed watermarking algorithm, as our 'green' token list (positive entries in logits scaling vector $v^{(t)}$) changes with the semantic meaning, the attacker needs to determine the list for any semantic meaning embedding. However, fixing a specific semantic is challenging because the semantics of the generated text can vary with each generation, making it difficult to measure and control consistently. Therefore, to help us qualify the decryption rate, we strengthen the spoofing attack. In our experiments, we maintain a fixed sentence embedding for all generations. This implies that the green token list remains unchanged, allowing us to verify if the green token list can be obtained for this fixed semantic embedding. We analyze the frequency of tokens in a manner similar to the previously mentioned methods.

We note that the decryption rate derived from this strengthened version of the spoofing attack serves as an upper bound because, in practice, attackers typically only have access to the watermarked LLM API, which generates watermarked text in response to input prompts. Even if an attacker repeatedly inputs the same prompt and generates multiple texts, the generated text may have different semantic meanings, and decrypting the mapping to the green token list becomes significantly more challenging.

As shown in Table 2, we can observe that KGW-0 exhibits the highest decryption rate because of its fixed 'green/red' list. For KGW-1 and KGW-2, the implementation of the previous token as a hash key adds a certain level of difficulty to the decryption process compared to the KGW-0. Our method demonstrates superior security performance compared to those works, even for the strengthened spoofing attack, with a decryption rate of only 0.571. This is because the AWTI further reduces the fraction of watermarked tokens in the text, thereby diluting the evident token preference within a specific semantic range. More discussion regarding the benefits of AWTI to security can be found in Appendix D.

5.5. Ablation Study

In this section, our investigation focuses on assessing the robustness and text quality of watermarked text using our watermark under varying entropy thresholds and watermark strength. Specifically, we adjust the threshold value α to 0, 1, 2, 3 and set the watermark strength δ at 0.5, 1, 1.5, and 2. In Appendix E, we further analyze the performance of our watermark under different measurement models, and evaluate the effectiveness of each component in our adaptive watermark.

Entropy Thresholds. In Table 3, we provide the ROC-AUC, Best F1 Score under different entropy thresholds when the watermarked text is paraphrased using GPT-3.5. In addition, we provide the Average Watermark Rate (AWR), which is the fraction of the watermarked tokens. Figure 4 (left) shows the text perplexity comparison under different entropy thresholds. When $\alpha = 0$, AWTI is skipped in watermark generation. In this case, the ROC-AUC and Best F1 Score are high, but the watermarked text exhibits high perplexity, with a median value exceeding 10. When setting α to 1, 2, or 3, there is a notable decrease in the perplexity of the watermarked text, as fewer tokens are perturbed, reflected by the decreasing AWR. In addition, the robustness of the watermark is improved when AWTI is applied ($\alpha = 1, 2$), potentially because some modified tokens are filtered out by AWTI in detection. For entropy threshold $\alpha = 3$, our method can achieve results comparable to other methods, with only about half of the watermarked tokens.

Table 3: Robustness performance for different entropy thresholds.

Language Model	Evaluation	$\alpha = 0$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$
OPT-6.7	ROC-AUC	0.978	0.988	0.981	0.972
	Best F1 Score	0.935	0.957	0.932	0.917
	AWR	1.000	0.835	0.707	0.553

Watermark Strength. In Table 4 and Figure 4 (right), we increase the watermark strength δ from 0.5 to 2 to examine its impact on robustness and perplexity. As shown in Table 4, there is little impact on the ROC-AUC values in terms of robustness. Interestingly, in contrast to previous works (Kirchenbauer et al., 2023a; Zhao et al., 2023) where perplexity tends to rise with larger watermark strength, the perplexity of the text watermarked with our method decreases as the δ increases in a reasonable range. This benefits from the proposed AWTS, which embeds the watermark by adjusting *temperature* as explained in Section 3.3. However, this does not imply that a larger δ consistently results in better text quality. A higher watermark strength can cause the token selection process to resemble greedy sampling more closely, potentially leading to the generation of repetitive content.

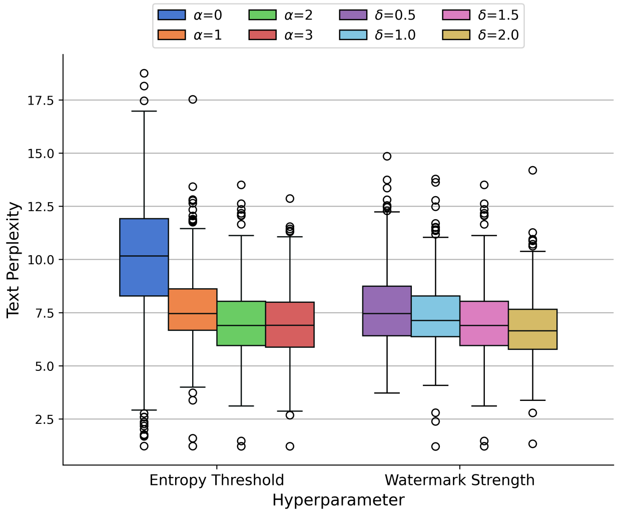


Figure 4: Comparison of text perplexity at varying entropy thresholds (left) and across different watermark strengths (right). α represents the entropy threshold. δ represents the watermark strength.

Table 4: Robustness performance for different watermark strength.

Language Model	Evaluation	$\delta = 0.5$	$\delta = 1$	$\delta = 1.5$	$\delta = 2$
OPT-6.7	ROC-AUC	0.982	0.984	0.981	0.978
	Best F1 Score	0.938	0.945	0.932	0.938

6. Conclusion

In this paper, we propose a holistic LLM watermark method to achieve the following properties: strong robustness, high security, high text quality, and accurate detection without the need to access the original language model and prompt. Our experiments show that, compared to baseline methods, our approach demonstrates better performance under different paraphrase attacks. Notably, our method has a negligible impact on text perplexity when compared to *unwatermarked* text. Additionally, we highlight the security of our approach by employing the spoofing attack. Specifically, the capacity of language models differs among various models. Therefore, the model owner may need to tune the entropy threshold and watermark strength carefully based on the performance of the language model to get better watermarking performance. Moreover, it is recommended for watermarked model owners to train their own small-sized measurement model based on the watermarked model for performance and security considerations. In future research, we aim to conduct a theoretical analysis to deepen the understanding of the performance characteristics of the watermarking approach and offer an optimal solution.

Impact Statement

Adding a watermark to the LLMs can significantly enhance transparency and accountability by making it easier to identify AI-generated content. Its impact extends broadly in the era of AI, including safeguarding intellectual property (IP), reducing the dissemination of misinformation, and mitigating the misuse of AI-generated content across educational and other domains. Furthermore, looking ahead to societal consequences, effective watermarking has the potential to bolster public trust in AI technologies by offering a clear means of distinguishing between human and AI-generated content.

References

Abdelnabi, S. and Fritz, M. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 121–140. IEEE, 2021.

Atallah, M. J., Raskin, V., Crogan, M., Hempelmann, C., Kerschbaum, F., Mohamed, D., and Naik, S. Natural language watermarking: Design, analysis, and a proof-of-concept implementation. In *Information Hiding*, pp. 185–200, 2001.

Brassil, J., Low, S., Maxemchuk, N., and O'Gorman, L. Electronic marking and identification techniques to discourage document copying. *IEEE Journal on Selected Areas in Communications*, 13(8):1495–1504, 1995. doi: 10.1109/49.464718.

Christ, M., Gunn, S., and Zamir, O. Undetectable watermarks for language models. *arXiv preprint arXiv:2306.09194*, 2023.

Fairoze, J., Garg, S., Jha, S., Mahloujifar, S., Mahmoody, M., and Wang, M. Publicly detectable watermarking for language models. *arXiv preprint arXiv:2310.18491*, 2023.

Fan, A., Jernite, Y., Perez, E., Grangier, D., Weston, J., and Auli, M. Eli5: Long form question answering. *arXiv preprint arXiv:1907.09190*, 2019.

Fu, J., Zhao, X., Yang, R., Zhang, Y., Chen, J., and Xiao, Y. Gumbelsoft: Diversified language model watermarking via the gumbelmax-trick. *arXiv preprint arXiv:2402.12948*, 2024.

Hou, A. B., Zhang, J., He, T., Wang, Y., Chuang, Y.-S., Wang, H., Shen, L., Van Durme, B., Khashabi, D., and Tsvetkov, Y. Semstamp: A semantic watermark with paraphrastic robustness for text generation. *arXiv preprint arXiv:2310.03991*, 2023.

Hu, Z., Chen, L., Wu, X., Wu, Y., Zhang, H., and Huang, H. Unbiased watermark for large language models. *arXiv preprint arXiv:2310.10669*, 2023.

Huo, M., Somayajula, S. A., Liang, Y., Zhang, R., Koushanfar, F., and Xie, P. Token-specific watermarking with enhanced detectability and semantic coherence for large language models. *arXiv preprint arXiv:2402.18059*, 2024.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Kirchenbauer, J., Geiping, J., Wen, Y., Katz, J., Miers, I., and Goldstein, T. A watermark for large language models. *arXiv preprint arXiv:2301.10226*, 2023a.

Kirchenbauer, J., Geiping, J., Wen, Y., Shu, M., Saifullah, K., Kong, K., Fernando, K., Saha, A., Goldblum, M., and Goldstein, T. On the reliability of watermarks for large language models. *arXiv preprint arXiv:2306.04634*, 2023b.

Krishna, K., Song, Y., Karpinska, M., Wieting, J., and Iyyer, M. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *arXiv preprint arXiv:2303.13408*, 2023.

Kuditipudi, R., Thickstun, J., Hashimoto, T., and Liang, P. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*, 2023.

Lee, T., Hong, S., Ahn, J., Hong, I., Lee, H., Yun, S., Shin, J., and Kim, G. Who wrote this code? watermarking for code generation. *arXiv preprint arXiv:2305.15060*, 2023.

Liu, A., Pan, L., Hu, X., Li, S., Wen, L., King, I., and Yu, P. S. An unforgeable publicly verifiable watermark for large language models. *arXiv preprint arXiv:2307.16230*, 2023a.

Liu, A., Pan, L., Hu, X., Meng, S., and Wen, L. A semantic invariant robust watermark for large language models. *arXiv preprint arXiv:2310.06356*, 2023b.

Meral, H. M., Sankur, B., Sumru Özsoy, A., Güngör, T., and Sevinç, E. Natural language watermarking via morphosyntactic alterations. *Computer Speech & Language*, 23(1):107–125, 2009. doi: https://doi.org/10.1016/j.csl.2008.04.001.

Munyer, T. and Zhong, X. Deeptextmark: Deep learning based text watermarking for detection of large language model generated text. *arXiv preprint arXiv:2305.05773*, 2023.

Por, L. Y., Wong, K., and Chee, K. O. Unispach: A text-based data hiding method using unicode space characters. *Journal of Systems and Software*, 85(5):1075–1082, 2012.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), jan 2020. ISSN 1532-4435.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

Ren, J., Xu, H., Liu, Y., Cui, Y., Wang, S., Yin, D., and Tang, J. A robust semantics-based watermark for large language model against paraphrasing. *arXiv preprint arXiv:2311.08721*, 2023.

Rizzo, S. G., Bertini, F., and Montesi, D. Content-preserving text watermarking through unicode homoglyph substitution. In *Proceedings of the 20th International Database Engineering & Applications Symposium*, pp. 97–104. Association for Computing Machinery, 2016. doi: 10.1145/2938503.2938510.

Sadasivan, V. S., Kumar, A., Balasubramanian, S., Wang, W., and Feizi, S. Can ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*, 2023.

Sato, R., Takezawa, Y., Bao, H., Niwa, K., and Yamada, M. Embarrassingly simple text watermarks. *arXiv preprint arXiv:2310.08920*, 2023.

Shannon, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

Topkara, M., Topkara, U., and Atallah, M. J. Words are not enough: Sentence level natural language watermarking. In *Proceedings of the 4th ACM International Workshop on Contents Protection and Security*, pp. 37–46. Association for Computing Machinery, 2006. doi: 10.1145/1178766.1178777.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023. *URL https://arxiv.org/abs/2307.09288*, 2023.

Tu, S., Sun, Y., Bai, Y., Yu, J., Hou, L., and Li, J. Waterbench: Towards holistic evaluation of watermarks for large language models. *arXiv preprint arXiv:2311.07138*, 2023.

Wang, B. and Komatsuzaki, A. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`, May 2021.

Wang, L., Yang, W., Chen, D., Zhou, H., Lin, Y., Meng, F., Zhou, J., and Sun, X. Towards codable text watermarking for large language models. *arXiv preprint arXiv:2307.15992*, 2023.

Wu, Y., Hu, Z., Zhang, H., and Huang, H. Dipmark: A stealthy, efficient and resilient watermark for large language models. *arXiv preprint arXiv:2310.07710*, 2023.

Yang, X., Zhang, J., Chen, K., Zhang, W., Ma, Z., Wang, F., and Yu, N. Tracing text provenance via context-aware lexical substitution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 11613–11621, 2022.

Yang, X., Chen, K., Zhang, W., Liu, C., Qi, Y., Zhang, J., Fang, H., and Yu, N. Watermarking text generated by black-box language models. *arXiv preprint arXiv:2305.08883*, 2023.

Yoo, K., Ahn, W., Jang, J., and Kwak, N. Robust multi-bit natural language watermarking through invariant features. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2092–2115, 2023. doi: 10.18653/v1/2023.acl-long.117.

Zhang, H., Edelman, B. L., Francati, D., Venturi, D., Ateniese, G., and Barak, B. Watermarks in the sand: Impossibility of strong watermarking for generative models. *arXiv preprint arXiv:2311.04378*, 2023a.

Zhang, R., Hussain, S. S., Neekhara, P., and Koushanfar, F. Remark-llm: A robust and efficient watermarking framework for generative large language models. *arXiv preprint arXiv:2310.12362*, 2023b.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Zhao, X., Ananth, P., Li, L., and Wang, Y.-X. Provable robust watermarking for ai-generated text. *arXiv preprint arXiv:2306.17439*, 2023.

Zhao, X., Li, L., and Wang, Y.-X. Permute-and-flip: An optimally robust and watermarkable decoder for llms. *arXiv preprint arXiv:2402.05864*, 2024.

A. Semantic Mapping Model Implementation Details

The Semantic Mapping Model is a neural network that starts and ends with linear layers and contains two residual blocks in the middle. For the loss function provided in Section 3.2, the weight of each term is set as 1. We train the model with the standard stochastic gradient descent with batch size 128.

As discussed in Section 3.2 , we rescale Euclidean distance between two sentence embeddings $d = D(u(S), u(S'))$ to a wider range. The original range $d \in [L, U]$, where $L = 0$ and U is the maximum distance. To enlarge this range, we apply a linear transformation $(T : \mathbb{R} \to \mathbb{R})$ to the original distance d, resulting in a new distance $d' = T(d) \in [L', U']$ where $L' < L$ and $U' > U$. The linear transformation is expressed as

$$T(d) = \left(\frac{d - L}{U - L}\right) \cdot (U' - L') + L'. \tag{6}$$

In our experiment, we set the $L' = -2$ and $U' = 4$.

B. Additional Evaluation Results

B.1. Experiment Results on Additional LLM

To further showcase the performance of our watermarking algorithm, we conduct experiments on a more powerful LLM, Mistral-7B, using the C4 dataset. Moreover, we use a more powerful LLM, GPT-3, to calculate the perplexity. As presented in Table 5 and Figure 5, our watermarking method not only demonstrates superior robustness performance but also maintains lower text perplexity compared to other methods.

Table 5: The comparison of robustness performance across various watermarking methods, conducted on Mistral-7B with C4 dataset

Language Model	Evaluation	KGW-0	KGW-1	KGW-2	EXP-edit	**Ours**
Mistral-7B	ROC-AUC	0.952	0.854	0.792	0.872	0.961
	Best F1 Score	0.887	0.785	0.747	0.833	0.902

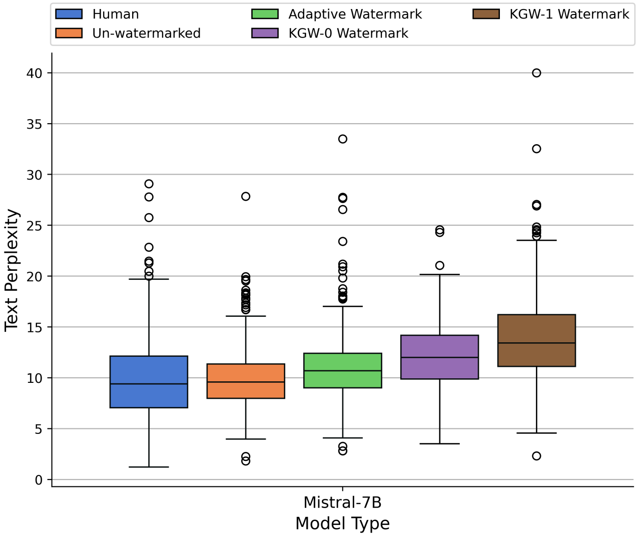


Figure 5: The comparison of text perplexity across various watermarking methods, conducted on Mistral-7B with C4 dataset, with the perplexity calculated using GPT-3.

B.2. Experiment Results on Additional Dataset

Besides utilizing C4, which generates subsequent sentences from the first two as prompts, we also consider the long-form question-answering dataset ELI5 to reflect real-world LLM use cases. We conduct experiments with the ELI5 dataset using the OPT-6.7B model to assess robustness performance and text perplexity. In these experiments, watermarked texts were paraphrased by GPT-3, and text perplexity was also calculated using GPT-3. Table 6 indicates that our robustness performance is on par with KGW-0, while Figure 6 demonstrates that the text perplexity associated with our watermarking technique is significantly lower compared to other methods.

Table 6: The comparison of robustness performance across various watermarking methods, conducted on OPT-6.7 with ELI5 dataset.

Language Model	Evaluation	KGW-0	KGW-1	KGW-2	EXP-edit	**Ours**
OPT-6.7	ROC-AUC	0.971	0.781	0.642	0.886	0.967
	Best F1 Score	0.913	0.733	0.668	0.857	0.899

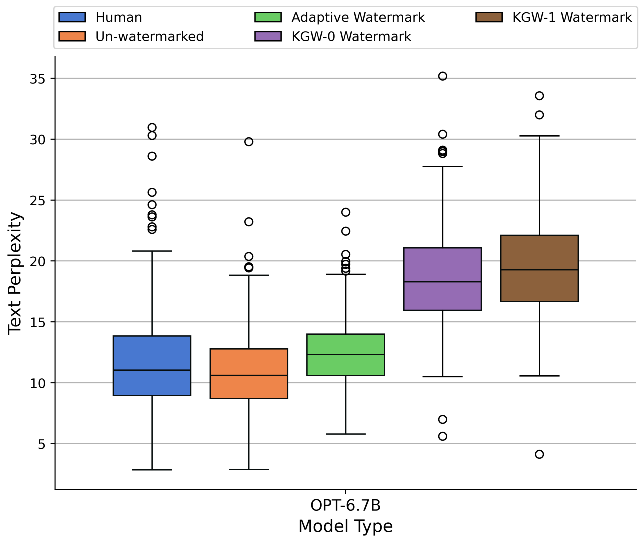


Figure 6: The comparison of text perplexity across various watermarking methods, conducted on OPT-6.7 with ELI5 dataset, with the perplexity calculated using GPT-3.

B.3. ROC Curves

Figure 7 illustrates the ROC curves and the corresponding AUC values when text is watermarked by different watermarking methods under paraphrasing attack. It is evident that our watermark method exhibits better robustness against paraphrase attacks compared to the other watermarking methods.

B.4. TPR at Fixed Low FPR

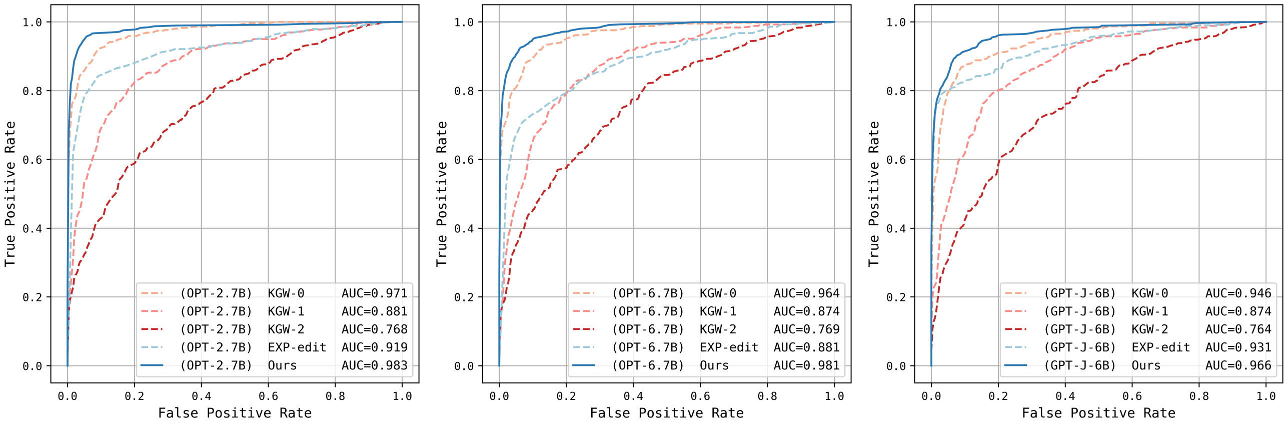
In practical applications, it is essential to monitor the TPR at a consistently low FPR to guarantee that un-watermarked texts are not mistakenly identified as watermarked. We showcase the TPR scores at fixed FPR thresholds of 1% and 10%, respectively, to demonstrate robustness performance under OPT-6.7B, as depicted in the Table 7.

Table 7: Robustness performance at fixed low FPR.

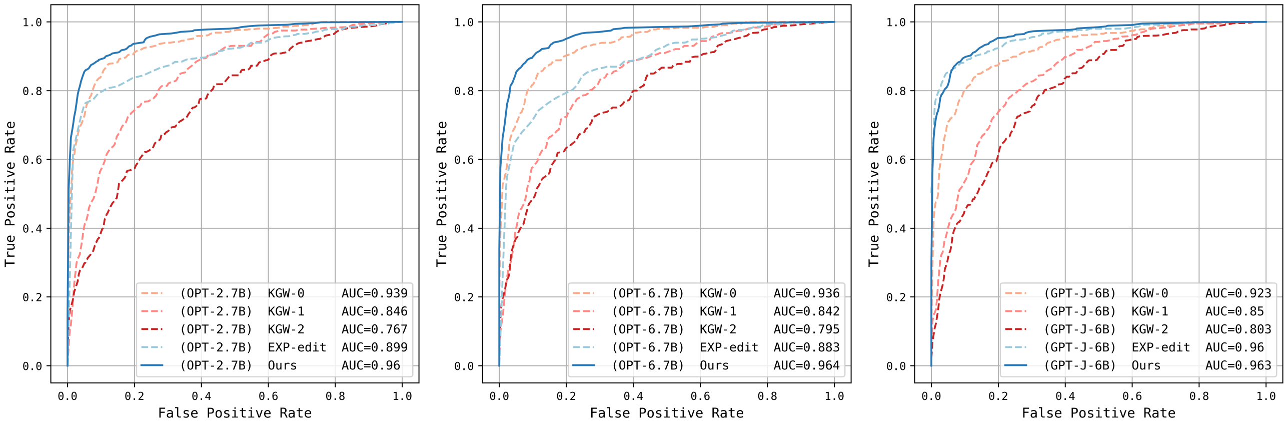
Language Model	Methods	TPR@1%FPR	TPR@1%FPR	ROC-AUC	Best F1 Score
OPT-6.7B	KGW-0	0.670	0.896	0.964	0.904
	KGW-1	0.198	0.653	0.874	0.815
	KGW-2	0.179	0.455	0.769	0.728
	EXP-edit	0.257	0.731	0.881	0.801
	Ours	0.779	0.949	0.981	0.932

B.5. Repetition Rate

To examine the repetition problem of watermarked text, we compute the different N-gram repetition rates for the watermarked text generated by different watermarking methods, setting N to 1, 2, and 3 separately. Table 8 shows that the repetition rate of our method is similar to that of the EXP-edit, which is a distortion-free watermarking technique.



(a) Paraphrasing attack using GPT-3.5.



(b) Paraphrasing attack using DIPPER.

Figure 7: Comparisons of ROC curves of different watermark methods applied to various language models against two different paraphrasing attacks.

Table 8: Repetition rate for watermarked text generated by different watermarking methods.

Language Model	Methods	1-gram	2-gram	3-gram	4-gram
OPT-6.7B	KGW-0	0.219	0.046	0.019	0.012
	KGW-1	0.199	0.037	0.013	0.007
	KGW-2	0.199	0.036	0.015	0.010
	EXP-edit	0.239	0.062	0.026	0.018
	Ours	0.241	0.066	0.026	0.014

C. Comparison with Additional Baselines

We note that Lee et al. (2023) also discuss the strategy of perturbing only the high-entropy distributions during generation. However, their method requires access to both the original LLMs and prompts. Accessing the original prompt during the detection phase can be impractical, and employing the source watermarked LLM for detection is not only time-consuming but also incurs substantial computational costs.

Moreover, Liu et al. (2023b) propose a semantic-based watermarking method, which also relies on the original prompts for detection. While this method enhances the security of the watermark, there remains a security risk, as attackers might fix the semantics of prompts to access the corresponding green token list. Our algorithm has a more stable and robust Semantic Mapping Model to handle the perturbation of semantics after modification (e.g., paraphrase), further improving our watermark's robustness. Moreover, our proposed AWTI further strengthens the security and improves the quality of watermarked text.

We conduct experiments to evaluate the robustness performance (after paraphrasing by GPT-3) and the perplexity of watermarked text using our algorithm (calculated by GPT-3), comparing it against both SWEET (Lee et al., 2023) and SIR (Liu et al., 2023b). All experiments are based on the OPT-6.7B model. The results of robustness performance are presented in Table 9, and the results of text perplexity are presented in Figure 8. It is worth noting that the comparison is not completely fair, as both SWEET and SIR are not model-agnostic. This implies that their detection processes involve using either the original prompt or the source watermarked LLM. The results show that we achieve better robustness performance and lower perplexity.

Table 9: Robustness performance comparison among Adaptive Watermark, SWEET, and SIR methods.

Language Model	Methods	ROC-AUC	Best F1 Score
OPT-6.7	SWEET	0.847	0.787
	SIR	0.963	0.903
	Ours	0.981	0.932

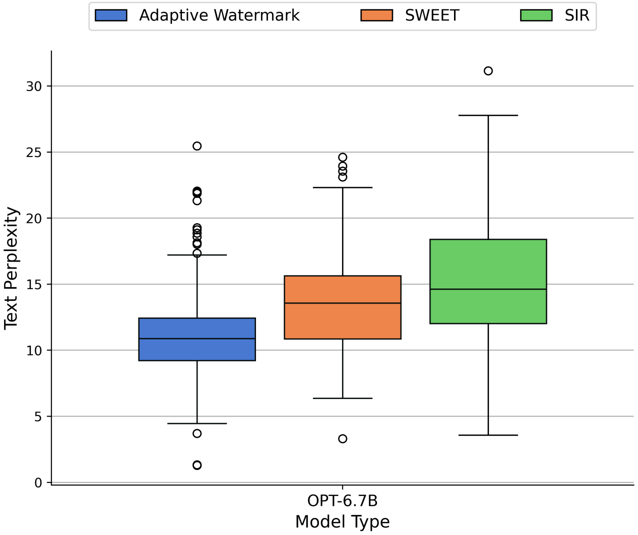


Figure 8: The comparison of text perplexity among Adaptive Watermark, SWEET, and SIR methods, conducted on OPT-6.7, with the perplexity being calculated using GPT-3.

D. Security Discussion

The security of a watermarking algorithm here refers to the difficulty an attacker has in statistically analyzing and decrypting the watermarking rules using the watermarked texts with the aim of forging the watermarks. For previous approaches such as KGW-k, the rule of green/red list can be simply recovered by analyzing token frequency after a specific fixed token, as the green/red list is purely determined using k previous tokens. For our semantic mapping model, the input sentence embedding space is continuous, and it depends on all the previously generated tokens, which requires the attackers to analyze token frequency based on fixed semantics. However, it is hard in practice, as even with the same prompt, the semantics of the generated text could vary widely. Consequently, for attackers aiming to identify the green token list associated with a particular semantic meaning, more effort or considerably larger sample size is required for our algorithm to conduct a meaningful analysis.

In general, we believe that a watermarking algorithm with more complex procedures will necessitate greater effort or a larger dataset for attackers to forge watermarking. To see this, we consider the examples of KGW-0 and KGW-k. KGW-0 employs a globally fixed green/red token list for generating watermarked text, making it the most straightforward to decrypt, i.e., simply analyzing overall token frequency. In contrast, KGW-k enhances security by utilizing previously generated tokens to formulate the green/red token list, requiring attackers to analyze token frequency in relation to specific tokens, thereby

Table 10: Green token percentage for different methods.

Methods	Green token percentage	ROC-AUC
KGW-0	0.649	0.964
Ours ($\frac{\lvert G\rvert}{\lvert U\cup W\rvert}$)	0.499	–
Ours ($\frac{\lvert \hat{G}\rvert}{\lvert W\rvert}$)	0.706	0.981

complicating the decryption process. Our watermarking algorithm further advances the process by deriving the green/red token list from the semantics of previously generated text and adaptively applying it to tokens with high entropy. As the attackers cannot accurately identify watermarked tokens (high entropy tokens measured using the specific measurement model), this adds another layer of difficulty in statistically analyzing the token frequency, even within a specific semantic context. Therefore, the complexity of our watermarking algorithm makes it more challenging for attackers to decrypt the watermarking rules, thereby enhancing the security of the watermarking algorithm.

Specifically, the implementation of AWTI results in fewer green tokens (positive entries in logits scaling vector $v^{(t)}$) within the generated text compared to other methods like KGW-0/KGW-1/KGW-2. However, the power of our watermark will not be affected. As outlined in Section 4 of our detection process, we prioritize filtering out potential un-watermarked tokens (U) and focus on the proportion of green tokens ($\hat{G}$) among potential watermarked tokens (W), rather than the proportion of green tokens (G) among all tokens ($U \cup W$) in the generated text, as done in KGW-0/KGW-1/KGW-2. Consequently, while the overall ratio of green tokens to all tokens in the generated text ($\frac{|G|}{|U \cup W|}$) may drop, the proportion of green tokens among potential watermarked tokens ($\frac{|\hat{G}|}{|W|}$) might not necessarily decrease. To validate this, we analyzed paraphrased watermarked texts, calculating the average percentage of green tokens across 500 texts generated by KGW-0 and comparing it with the average percentage of green tokens ($\frac{|\hat{G}|}{|W|}$) across 500 texts produced by our watermarking algorithm. The findings, detailed in Table 10, reveal that although the overall percentage of green tokens in the generated text ($\frac{|G|}{|U \cup W|}$) stands at 0.499, the percentage of green tokens relative to potential watermarked tokens ($\frac{|\hat{G}|}{|W|}$) reaches 0.705, surpassing that of KGW-0. Furthermore, since the overall percentage of green tokens in the generated text is only about 0.5, this indicates that our generated texts do not exhibit a noticeable bias towards a specific group of tokens. This characteristic makes it more challenging for attackers to decrypt the watermarked texts by analyzing token frequency.

E. Additional Ablation Study

Measurement Models. We investigate how different Measurement Models (MMs) impact the robustness of our watermarks and the quality of the generated text. For this purpose, we employ MMs with different sizes and types, such as GPT2 (124M), GPT2-large (774M), OPT-125M, and OPT-350M, in our evaluation. For all models, we set the α and δ as 2 and 1.5, respectively. As shown in Table 11, there is only a minor influence on the robustness of the watermark when different MMs are adopted.

Table 11: Robustness performance for different Measurement Models.

Language Model	Evaluation	GPT2	GPT2-large	OPT-125M	OPT-350M
OPT-6.7	ROC-AUC	0.985	0.981	0.987	0.982
	Best F1 Score	0.951	0.932	0.947	0.938
	AWR	0.778	0.707	0.755	0.718

However, the effect on text quality is more nuanced. As indicated in Figure 9, models like GPT2-large and OPT-350M will result in lower perplexity compared to the GPT2 and OPT-125M. One primary reason is that different MMs produce different entropy estimates, which results in different AWRs even with the same entropy threshold. Generally, a higher AWR implies a larger number of watermarked tokens, which can slightly impact text quality.

Effectiveness of Each Component in Adaptive Watermark. Our proposed watermarking method mainly consists of three key components: Adaptive Watermark Token Identification (AWTI), Semantic-based Logits Scaling Vector Extraction (SLSVE), and Adaptive Watermark Temperature Scaling (AWTS). Here, we use KGW-0 as our baseline model and analyze the robustness and perplexity of watermarked text by sequentially adding our three components. Table 12 and Figure 10 exhibit the robustness and text perplexity performance under the following four different settings. `KGW-0+AWTI` represents

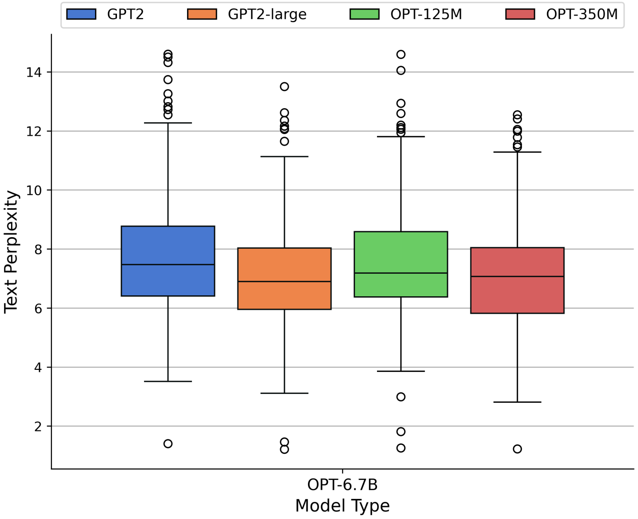


Figure 9: Text perplexity comparison between different Measurement Models.

applying AWTI to the KGW-0, aiming to adaptively perturb the distributions with low entropy. `KGW-0+AWTI+SLSVE` indicates that both AWTI and SLSVE are applied to KGW-0. `KGW-0+AWTI+SLSVE+AWTS` represents our complete Adaptive Watermark, where all our three components are applied.

Table 12: Robustness performance comparison after the removal of different components.

Language Model	Settings	ROC-AUC	Best F1 Score
OPT-6.7	KGW-0	0.964	0.904
	KGW-0 + AWTI	0.962	0.904
	KGW-0 + AWTI + SLSVE	0.960	0.893
	KGW-0 + AWTI + SLSVE + AWTS	0.981	0.932

Table 12 and Figure 10 demonstrate that the application of AWTI reduces the perplexity of the watermarked text without compromising robustness performance. The introduction of SLSVE enhances the security of the watermark but comes at the cost of increased perplexity in the watermarked text. Finally, when all three components are applied, our watermark method excels in both robustness and perplexity performance compared to existing baselines.

F. Watermarked Text Generation time

In this section, we compare the watermarked text generation time and the watermark detection time for different approaches. For each method, we generate and detect 100 instances of watermarked text, each containing roughly 200 tokens. Subsequently, we compute the average time taken for both the generation and detection of each instance of watermarked text. For our method, we set the value of the entropy threshold $\alpha = 2$ and the watermark strength $\delta = 1.5$. We evaluate the generation and detection time of the proposed adaptive watermarking with two different Measurement models.

Table 13 presents all the results. For generation time, despite employing three additional models in our watermarked text generation process, our average generation time remains faster than that of KGW-0 and KGW-1. Our method is slightly slower than EXP-edit, primarily because EXP-edit uses a sampling-based watermarking approach that does not require logit computation. Specifically, using a Measurement model with smaller parameters can lead to faster generation times. Regarding the time required for detection, our method is slightly slower than KGW-0 and KGW-1. However, it is significantly faster than EXP-edit, which necessitates computing the alignment between the watermarked text and the watermark key sequence.

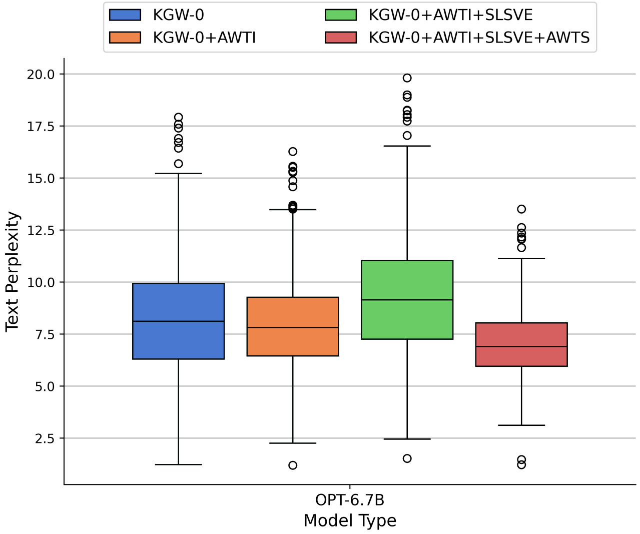


Figure 10: Text perplexity comparison after the removal of different components.

Table 13: Watermarked text generation and detection time across various watermarking methods.

Language Model	Watermarking Method	Settings	Average Generation Time	Average Detection Time
OPT-6.7	KGW-0	-	81.64s	0.01s
	KGW-1	-	85.15s	1.73s
	EXP-edit	-	25.89s	163.19s
	Ours	GPT2-large	33.16s	5.57s
	Ours	OPT-125M	31.15s	3.63

G. Watermarked Text Example

We present some examples of watermarked text using our proposed method. Table 14 shows the un-watermarked text and watermarked text under different entropy thresholds (α). The watermarked texts are generated using the OPT-6.7B model with the watermark strength $\delta = 1.5$. The watermarked tokens are highlighted in yellow. As the value of α increases, it is evident that the proportion of watermarked tokens noticeably decreases.

Table 14: Examples of watermarked text using our method with various entropy thresholds. In these examples, tokens highlighted in yellow represent watermarked tokens, indicating that these tokens were sampled from a perturbed distribution.

Prompt	It's not always easy to keep mentally focused during running. Focus requires discipline and a certain level of mental toughness, meaning that you have developed the capacity to perform to the best of your ability despite external conditions or internal distractions.
Un-watermarked text	While it's not impossible to train for a specific event and keep mentally focused at all times, it's challenging. Here are three things you can do to help yourself. Plan Ahead The best time to think about race strategy is during warm-ups before any race. Once your legs get moving, they will tell you what they need—you just have to listen to them! It's also a good idea to write down in detail the training plan for your event and make sure that you understand it completely. Plan your warm-up so that there is an emphasis on building speed without injury risk. Get Out of Your Head There are lots of other things to focus your attention on besides running. If you're running around town, take a walk break every quarter mile or so. This allows you to stretch, cool off and rest, while keeping your muscles engaged.

Watermarked text ($\alpha = 1$)	The ability and desire for discipline are developed through practice. The first step in developing discipline during running, especially if a runner wants a higher level performance, begins with setting goals and working towards meeting these goals. Goal setting allows runners an avenue to create the mental toughness needed to achieve their best results during training runs or races. Setting realistic expectations will also allow runners to work within their abilities instead of trying to perform beyond what they are currently capable of. Setting realistic expectations can also serve as motivation for runners to continue training despite the obstacles that life has to offer during the course of training or race day. It can also serve as motivation to continue training even when injuries or other problems make training challenging or downright unpleasant. For runners who want to set realistic expectations for themselves during training or race day, the below checklist can serve as guidelines: Mentally prepared - Have enough sleep the night before the race or training runs Stress free environment - Remove negative people from the environment Eat right - Eat healthy meals the day before the race or training runs
Watermarked text ($\alpha = 2$)	The best runners are often described in terms like, focused, determined, and driven. You may have read that somewhere on a running magazine cover, but did anyone actually write that down? I was recently asked if I knew any specific exercises to help people stay mentally focused during training runs. My response was ”No,” but the question got me thinking more about how to develop mental toughness for runners. Staying mentally prepared is an important part of the training process that can be just as important as physical preparation, but many runners don’t give it much thought. It’s easy to see how physical preparation is important for race day performance, but what exactly does it mean when you talk about mental preparation? Here are five things you need to do to stay mentally prepared while running: 1. Have realistic goals and expectations. Some people set unrealistic expectations for themselves or others regarding the number of miles they’ll run each week, their times for certain races, or even their overall goal for their training program.
Watermarked text ($\alpha = 3$)	The best runners are focused on a specific task, which allows for a level playing field. The same applies in any aspect in which concentration and discipline are necessary. In order for a runner, cyclist, tennis star, golffer, etc. to perform their best, they must have the ability to stay focused at all times. To achieve this state of focus or mental toughness requires training (mental conditioning) and practice. You do not have to run every day for hours, but just like anything else it’s about putting in the time and practicing until you get better over time. So how does this apply to your life? The best runners know that running is an important part of their lives, as well as their training routine and their overall health. The best runners also understand what is required to keep them mentally strong during the race. They know how to train themselves so that they will always be ready and prepared for the next race, regardless of external conditions or the obstacles within themselves.