Semantic Self-Guided Watermarking with Enhanced Text Quality for Large Language Models

Anonymous ACL submission

Abstract

The rapid development of large language models (LLMs) has created an urgent need for identifying machine-generated texts, and text watermarking technology has proven to be an effective solution. However, current watermarking methods, while demonstrating strong detectability, significantly degrade text quality due to the introduction of unnatural tokens. The main reason lies in the fact that these methods ignore the importance of semantic information in the watermarking process. To address this issue, we note that the logit vector produced by LLMs encodes both semantic understanding of input texts and prediction confidence across different tokens. Therefore, we propose a novel Semantic Self-Guided Watermarking (SSGW) framework that leverages the LLM itself to generate a guidance logic vector that assists in watermarking while producing the original one concurrently. Subsequently, we design a transform module to analyze these two vectors comprehensively and then transform them into adaptive watermark logits for different candidate tokens, thereby reducing the possibility of selecting inappropriate tokens. Experimental results confirm the effectiveness of our method in achieving superior performance in both watermark detectability and text quality preservation. The source code will be made publicly available upon acceptance.

1 Introduction

In recent years, the rapid advancement of large language models (LLMs) has ushered in a new era of natural language processing (Ouyang et al., 2022; Touvron et al., 2023b; OpenAI, 2023a). They can generate coherent and contextually relevant texts that often rival human-written content in terms of quality and fluency. However, this technological leap forward has also caused a multitude of ethical and moral concerns in various domains. Specifically, the misuse of LLM-generated essays can lead to academic dishonesty (Stokel-Walker, 2022), and the fabrication of fake news can even provoke social panic (Augenstein et al., 2024). Therefore, distinguishing text generated by LLMs from that written by humans has become an essential task.

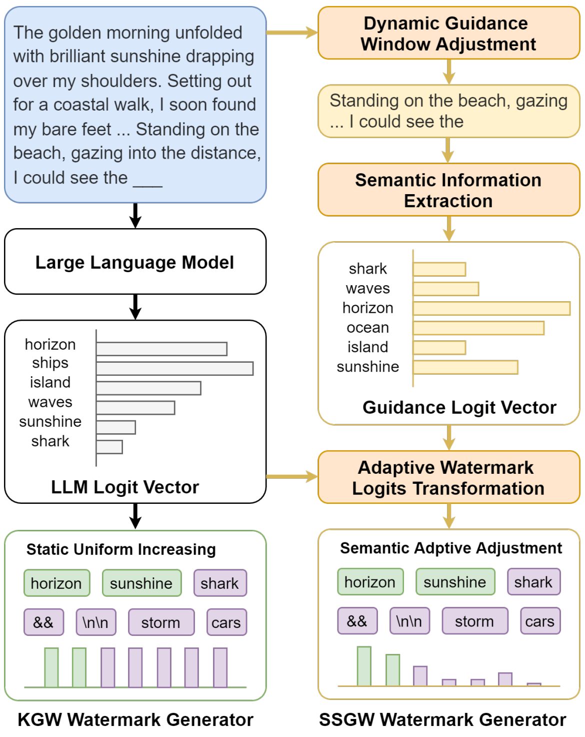
The golden morning unfolded with brilliant sunshine drapping over my shoulders. Setting out for a coastal walk, I soon found my bare feet ... Standing on the beach, gazing into the distance, I could see the ___
Large Language Model
horizon
ships
island
waves
sunshine
shark
LLM Logit Vector
Static Uniform Increasing
horizon
sunshine
shark
&&
\n\n
storm
cars
KGW Watermark Generator
Dynamic Guidance Window Adjustment
Standing on the beach, gazing ... I could see the
Semantic Information Extraction
shark
waves
horizon
ocean
island
sunshine
Guidance Logit Vector
Adaptive Watermark Logits Transformation
Semantic Adptive Adjustment
horizon
sunshine
shark
&&
\n\n
storm
cars
SSGW Watermark Generator

Figure 1: Current watermarking methods (represented by KGW (Kirchenbauer et al., 2023)) often degrade text quality by introducing unnatural tokens. In this case, it is obvious that "cars", "storm", or even "&&" are not suitable for contextual semantics. However, KGW assigns these tokens the same watermark logit as appropriate ones. In contrast, our SSGW method (right) designs a semantic self-guided watermarking framework to adaptively assign reasonable watermark logits to different candidate tokens, effectively reducing the possibility of inappropriate token replacement.

As a solution, the text watermarking technique effectively alleviates these issues by embedding hidden patterns into LLM-generated texts. These patterns are invisible to humans but can be detected by the corresponding algorithm. Kirchenbauer et al.

(2023) design a watermarking algorithm (KGW) based on logits modification. This method partitions the vocabulary into a red list and a green list using a hash function depending on the preceding token at each generation step. Then, a positive constant (termed the watermark logit) is added to the logit value of green list tokens, increasing their probability of being sampled. As a result, a text will be considered watermarked by the detector if it contains more than a certain number of green list tokens. The design of KGW can easily improve the detectability of watermarks by increasing the constant value. However, this approach often significantly degrades the quality of the generated text due to the introduction of a large number of unnatural tokens. As shown in Figure 1, the uniform logits modification strategy treats all tokens assigned to the green list as the same, while most of them are actually inappropriate.

To mitigate this issue, some works attempt to optimize the logits modification strategy instead of a positive constant used in KGW. Liu and Bu (2024) proposes an adaptive text watermarking method (ATW), which proportionally scales up the original logits instead of uniformly increasing. Wu et al. (2024) introduces a theoretically unbiased watermarking method (DIP) that discards tokens with probabilities below α and doubles those above $1 - \alpha$. However, these mathematical adjustments cannot guarantee a stable improvement in text quality. In addition, Lee et al. (2023) proposes a selective watermarking strategy (SWEET) that only applies watermarks to tokens with entropy higher than a certain threshold. Although SWEET is effective for low-entropy tasks such as code generation, this simple threshold-based strategy has little improvement in most scenarios.

Different from the aforementioned methods, our work emphasizes the importance of semantic information in the process of logits modification at each generation step. When modifying the logit value of a single green list token, both its semantic information and the semantics of its preceding text need to be taken into account. Therefore, we introduce a **S**emantic **S**elf-**G**uided **W**atermarking (SSGW) method (Figure 1, right). Specifically, we set a dynamically adjusted guidance window during text generation, and then the LLM itself is applied to the corresponding text. The obtained output, which we call the guidance logit vector, encodes both semantic understanding of the guidance window input and prediction confidence across different candidate tokens. Furthermore, in order to embed the semantic guidance message as a watermark, we design an algorithm that transforms the guidance logit vector into a watermark logit list, which has the same length as the vocabulary.

Properties	SWEET	ATW	DIP	SSGW
watermark detectability	✓	✓	✗	✓
text coherence and fluency	✗	✓	✗	✓
semantic preservation	✓	✗	✓	✓

Table 1: Experimental results in Section 5 show that existing watermarking techniques do not adhere to all three key properties.

During the evaluation phase, we systematically break down text quality into two key aspects: the semantic preservation ability and the fluency and coherence of the generated text. Correspondingly, we measure the former by calculating the semantic similarity between the watermarked text and the unwatermarked one, and the latter using the perplexity metric. Furthermore, if detectability is added, these three key properties will form a three-dimensional trade-off relationship.

To the best of our knowledge, there is no watermarking technique that adheres to all three key properties simultaneously, and the experimental results in Section 5 confirm this point as presented in Table 1. In summary, the contributions of our work are summarized as follows:

- We propose an innovative method called **SSGW**, which can make full use of semantic information in the preceding text to assist in watermarking through a self-guided approach.
- We are the first to find and verify a three-dimensional trade-off relationship in the evaluation system for enhancing watermarked text quality through extensive experiments.
- Experimental results show that SSGW effectively outperforms existing methods, achieving a simultaneous improvement of watermark detectability and text quality, especially in text coherence and fluency.

2 Related Work

The rapid advancement of LLMs has significantly narrowed the distinction between human-written and LLM-generated text, raising critical concerns about content authenticity and attribution (Radford et al., 2019; Brown et al., 2020). This blurring boundary has created an urgent need for reliable text authentication mechanisms, as traditional classification-based detection methods (Mitchell et al., 2023; OpenAI, 2023b) struggle to identify synthetic content accurately (Sadasivan et al., 2023; Chakraborty et al., 2023). In response, researchers have renewed interest in watermarking techniques adapted for modern AI systems.

Text watermarking, the practice of embedding imperceptible identifiers in textual content, has historically served as a cornerstone of copyright protection (Atallah et al., 2001). Conventional approaches typically relied on lexical substitution or syntactic pattern manipulation (Topkara et al., 2005; Meral et al., 2009), which have evolved significantly in the era of LLMs.

Recent breakthroughs have integrated watermarking directly into LLM generation processes, fully leveraging the advanced understanding of language semantics and contextual awareness inherent in LLMs. The most representative is the LLM watermarking technology based on logits modification proposed by Kirchenbauer et al. (2023). As a pioneer, although this method has demonstrated excellent watermark detectability, it still needs further optimization to be applied in the real world, especially in two key aspects: improving robustness against attacks and mitigating impact on text quality (Liu et al., 2024b).

A number of works have been proposed to enhance the robustness of watermarks. Based on KGW, Zhao et al. (2023) proves that a fixed partition of red and green lists contributes to stronger robustness against various attacks. Moreover, some studies attempt to integrate semantic information into their watermarking algorithm (Ren et al., 2023; Liu et al., 2024a; He et al., 2024; Liu and Bu, 2024), considering that most watermark removal attacks tend to preserve the semantics of the original text.

In addition, some works focus on improving the quality of watermarked texts. Hu et al. (2024) proposes a theoretically unbiased method using inverse sampling and reweighting technology to preserve the original text distribution. Similarly, Wu et al. (2024) extends this idea and proposes the α-reweight method with more general parameter settings. However, unbiased distribution does not imply lossless text quality, and these methods have shown poor detection performance. Lee et al. (2023) employs a more practical strategy that selectively applies watermarks to tokens with entropy higher than the threshold since lower entropy means less suitable tokens. However, this strategy is almost no different from KGW in most scenarios. Liu and Bu (2024) designs an adaptive watermark temperature scaling module, allocating higher watermark logits to tokens with higher probability. The disadvantage of this multiplicative method lies in that some tokens will be assigned excessively high watermark logits compared to those unmodified tokens. To address these shortcomings, our work designs a framework that constructs auxiliary guidance logit vectors to assist in adding watermarks, emphasizing the importance of utilizing semantic information.

3 Preliminaries

The watermark algorithm consists of a watermark generator $\mathcal{G}$ and a watermark detector $\mathcal{D}$. At each generation step t, the watermark generator can introduce subtle modifications to the logit vector $l^{(t)}$ obtained by the given LLM $\mathcal{M}$ over the vocabulary $\mathcal{V}$ based on the prompt $x_{-m:0}$ and the preceding generated text $x_{1:t-1}$. Then, the LLM will sample the next token based on the modified logit vector after performing softmax. This procedure is presented in Eq. 1.

$$\begin{aligned} l^{(t)} &= \mathcal{M}(x_{-m:0}, x_{1:t-1}) \\ x_t &\sim softmax(\mathcal{G}(l^{(t)})) \end{aligned} \tag{1}$$

As introduced in Kirchenbauer et al. (2023), we employ a similar red-green list strategy in this paper. Given the logit vector $l^{(t)}$, a predetermined constant γ is used to partition the vocabulary into a green list G_t of size $\gamma|V|$ and a red list R_t of size $(1-\gamma)|V|$, where $|V|$ is the size of the vocabulary. Then, we adjust the original logit vector with a watermark logit list $\delta^{(t)}_{|V|}$, using Eq.2. Specially, $\delta^{(t)}_{|V|}$ is a constant list in KGW.

$$\mathcal{G}(l^{(t)}[k]) = l^{(t)}[k] + \delta^{(t)}_{|V|}[k], \quad k \in G_t \tag{2}$$

The method of calculating the red-green list in the detection process is the same as during the generation. Given a token sequence $\boldsymbol{s} = \{s_1, s_2, ..., s_T\}$, let $|\boldsymbol{s}|_G$ denote the number of green list tokens in $\boldsymbol{s}$. Our detection is carried out through the one-sided z-test, specifically by calculating the z-score using Eq.3. If the z-score is higher than a given threshold, $\boldsymbol{s}$ will be considered watermarked.

$$z = \frac{|\boldsymbol{s}|_G - \gamma T}{\sqrt{T\gamma(1-\gamma)}} \tag{3}$$

4 Method

In this section, we will introduce our Semantic Self-Guided Watermarking (SSGW) method that simultaneously improves both watermark detectability and text quality. This method operates through three key modules: (1) Guidance Logit Vector Generation, (2) Adaptive Watermark Logits Transformation, and (3) Dynamic Guidance Window Adjustment.

4.1 Guidance Logit Vector Generation

The design of our Semantic Self-Guided Watermarking (SSGW) method is driven by two fundamental observations. First, an effective watermark generator should analyze the semantic information of the preceding text to evaluate token substitutability at each generation step. This requirement stems from the fact that arbitrary token replacements disrupt the semantic coherence of the generated text. Second, the logit vector produced by LLMs inherently reflects both semantic understanding of the given input and relative preference scores for different tokens in the vocabulary.

Consequently, it is natural to take into account the scheme of constructing an auxiliary guidance logit vector $\hat{l}^{(t)}$ different from the original $l^{(t)}$ during text generation and embedding this logit vector as a watermark. **Specifically, if these two logit vectors have different prediction results, we prefer to use the result of $\hat{l}^{(t)}$ as the next token by modifying $l^{(t)}$, achieving the effect of reducing semantic disturbance while altering the original selection of the LLM.**

Therefore, our SSGW method employs a strategy of setting a guidance window and then applying the LLM to the corresponding text. In this way, the obtained guidance logit vector can not only assist in selecting tokens that are suitable for contextual semantics but also have a stable difference in prediction results to enhance the watermark detectability. Specifically, the following computations will be performed at each generation step t:

$$\begin{aligned} l^{(t)} &= \mathcal{M}(x_{-m:0}, x_{1:t-1}) \\ \hat{l}^{(t)} &= \mathcal{M}(x_{t-L:t-1}) \end{aligned} \tag{4}$$

where $l^{(t)}$ incorporates the complete historical context from prompt $x_{-m:0}$ to previously generated tokens $x_{1:t-1}$, while $\hat{l}^{(t)}$ is obtained from the guidance window text $x_{t-L:t-1}$ of length L. This architecture ensures that $\hat{l}^{(t)}$ preserves semantic coherence while introducing controlled divergence.

4.2 Adaptive Watermark Logits Transformation

After obtaining the guidance logit vector, our logits modification strategy diverges from that of KGW. Our approach adaptively biases different candidate tokens in the vocabulary with varying watermark logits based on their semantics. Algorithm 1 describes the procedure of Adaptive Watermark Logits Transformation (AWLT).

Algorithm 1 Adaptive Watermark Logits Transformation (AWLT)

```
Input: Original logit vector $l^{(t)}$, guidance logit vector $\hat{l}^{(t)}$, initial suppression coefficient $\lambda_0$, guidance watermark strength $\delta_{\text{gui}}$, low entropy threshold $\theta$
 1: Convert the input logit vectors into probability distributions: $p^{(t)} = \text{softmax}\left(l^{(t)}\right)$, $\hat{p}^{(t)} = \text{softmax}\left(\hat{l}^{(t)}\right)$
 2: Obtain the entropy and maximum index of the original probability distribution: $H_{\text{ori}} = H(p^{(t)})$, $I_{\text{ori}} = \text{argmax}\left(p^{(t)}\right)$
 3: Obtain the entropy and maximum index of the guidance probability distribution: $H_{\text{gui}} = H(\hat{p}^{(t)})$, $I_{\text{gui}} = \text{argmax}\left(\hat{p}^{(t)}\right)$
 4: if $I_{\text{gui}} = I_{\text{ori}}$ then
 5:     $\lambda \Leftarrow \lambda_0 \cdot \text{Sigmod}\left(H_{\text{ori}} + H_{\text{gui}}\right)$
 6:     $\hat{p}^{(t)}[I_{\text{gui}}] \Leftarrow \frac{\hat{p}^{(t)}[I_{\text{gui}}]}{\lambda}$
 7:     if continuous $H_{\text{ori}} + H_{\text{gui}} < \theta$ then
 8:         update $\lambda_0 \Leftarrow \lambda_0 + 1$
 9:     end if
10: end if
11: $\delta^{(t)}_{|V|} = \delta_{\text{gui}} \cdot \frac{\hat{p}^{(t)}}{\hat{p}^{(t)}[I_{\text{gui}}]}$
Output: $\delta^{(t)}_{|V|}$
```

The AWLT procedure begins by converting both logit vectors into probability distributions $p^{(t)}$ and $\hat{p}^{(t)}$ through softmax normalization. We then analyze their prediction behaviors through two key metrics:

- **Prediction consensus**: The argmax function is applied to both probability distributions to obtain their maximum probability indices I_{ori} and I_{gui}. The comparison result between them shows whether both distributions agree on the most probable token.

- **Prediction uncertainty**: Shannon Entropy (Shannon, 1948), which is defined as $H = -\sum p_k \log p_k$, measures the uncertainty of a discrete probability distribution. We denote H_{ori} as the entropy of $p^{(t)}$ and H_{gui} as that of $\hat{p}^{(t)}$.

There are two possible outcomes for the comparison of prediction consensus, and we will handle them separately.

Case 1: Divergent predictions ($I_{\text{gui}} \neq I_{\text{ori}}$). When the guidance distribution suggests a different optimal token, we consider this a natural watermarking opportunity. The watermark logit list $\delta_{|V|}^{(t)}$ is calculated proportionally to the guidance distribution $\hat{p}^{(t)}$, scaled by the watermark strength parameter δ_{gui}.

Case 2: Consensus predictions ($I_{\text{gui}} = I_{\text{ori}}$). To improve the watermark detectability, we implement an entropy-adaptive Maximum Probability Suppression (MPS) mechanism using the following equation.

$$\lambda = \lambda_0 \cdot \text{Sigmod}\left(H_{\text{ori}} + H_{\text{gui}}\right) \tag{5}$$

where λ_0 is the predefined initial suppression coefficient. We use the sum of both entropy $H_{\text{sum}} = H_{\text{ori}} + H_{\text{gui}}$ as a basis for the coefficient adjustment. A very low H_{sum} indicates that both $p^{(t)}$ and $\hat{p}^{(t)}$ are concentrated on specific tokens. In this case, the watermark strength should be relatively weak, corresponding to our algorithm's use of a smaller suppression coefficient. Conversely, when H_{sum} is higher, it suggests that the distributions perceive multiple reasonable choices, allowing for a stronger watermark.

It is worth mentioning that if both $p^{(t)}$ and $\hat{p}^{(t)}$ have extremely low entropy under a predefined entropy threshold θ at the same time and $I_{\text{gui}} = I_{\text{ori}}$, it indicates a failure of watermark injection at this time step. If this extreme situation continuously occurs during the watermarking process, we will employ the Dynamic Suppression Coefficient Update (DSCU) mechanism to increase λ_0.

We note that both Lee et al. (2023) and Liu and Bu (2024) discuss the use of entropy for watermarking in the text generation process. However, their strategy is limited to setting a threshold, treating all tokens with entropy higher than the predefined threshold as the same. Their experimental results have proved the effectiveness of using entropy for watermarking. However, a more in-depth and reasonable approach should be adjusting the watermark strength adaptively according to different entropy scenarios, rather than a simple binary classification.

4.3 Dynamic Guidance Window Adjustment

To further improve the generated text quality, we design a dynamic guidance window adjustment framework, as detailed in Algorithm 2.

Algorithm 2 Dynamic Guidance Window Adjustment (DGWA)

```
Input: prompt $x_{-m:0}$, initial window length $L_0$, similarity threshold $\alpha$, initial watermark strength $\delta_{\text{ini}}$
1: Initialize $Start = False$, $L = L_0$
2: for $t = 1, 2, ...$ do
3:     Apply LLM to the full input $x_{-m:t-1}$ to get a logit vector $l^{(t)}$.
4:     Apply LLM to the window input $x_{t-L:t-1}$ to get an guidance logit vector $\hat{l}^{(t)}$.
5:     Utilize $l^{(t)}$ and $\hat{l}^{(t)}$ to obtain their cosine similarity $C_s = \cos_{\text{sim}}\left(l^{(t)}, \hat{l}^{(t)}\right)$
6:     if $Start \neq True$ then
7:         if $C_s > \alpha$ then
8:             $\delta_{|V|}^{(t)} \Leftarrow [\delta_{\text{ini}}] \times V$
9:         else
10:            $\delta_{|V|}^{(t)} = \text{AWLT}\left(l^{(t)}, \hat{l}^{(t)}\right)$
11:            update $L \Leftarrow L + 1$
12:            update $Start \Leftarrow True$
13:        end if
14:    else
15:        $\delta_{|V|}^{(t)} = \text{AWLT}\left(l^{(t)}, \hat{l}^{(t)}\right)$
16:        update $L \Leftarrow L + 1$
17:    end if
18:    if continuous $C_s > \alpha$ then
19:        update $L \Leftarrow L_0$
20:    end if
21: end for
```

At the beginning of text generation, we use a fixed window length $L = L_0$ to determine the starting point. At this stage, the guidance window text $x_{t-L:t-1}$ accounts for a high proportion of the full input $x_{-m:t-1}$, which can easily lead to a high similarity between $l^{(t)}$ and $\hat{l}^{(t)}$. To ensure enough difference between these two logit vectors, we calculate their cosine similarity C_s using the following equation.

$$C_s = \cos_{\text{sim}}\left(l^{(t)}, \hat{l}^{(t)}\right) = \frac{l^{(t)} \cdot \hat{l}^{(t)}}{\left\|l^{(t)}\right\| \times \left\|\hat{l}^{(t)}\right\|} \quad (6)$$

Cosine similarity is a measure of the cosine of the angle between two non-zero vectors in multi-dimensional space. It assesses the degree of similarity between these vectors independently of their magnitude. In our method, only when C_s is less than the predefined threshold α will we begin to carry out continuous guidance. Otherwise, we will use the uniform logits modification strategy as in KGW temporarily.

It is worth mentioning that the guidance logit vector will gradually move closer to the original one with the dynamic increase of L. Therefore, we design a Window Length Reset (WLR) mechanism to address scenarios where the cosine similarity between $l^{(t)}$ and $\hat{l}^{(t)}$ continuously exceeds the similarity threshold.

5 Experiments

5.1 Experiment Settings

Baselines and Models. Our method is compared with four methods, including KGW (Kirchenbauer et al., 2023), SWEET (Lee et al., 2023), ATW (Liu and Bu, 2024) and DIP (Wu et al., 2024). All experiments are conducted on four different models: OPT-6.7B (Zhang et al., 2022), GPT-J-6B (Wang and Komatsuzaki, 2021), LLAMA2-7B (Touvron et al., 2023a), and Qwen2.5-14B (Bai et al., 2023) (in Appendix A).

Evaluation Metrics. Excellent watermark detectability requires algorithms to correctly identify watermarked text while not mistakenly recognizing human text as watermarked text. We report the true positive rate (TPR) at a fixed 0% FPR for each method. In terms of generated text quality, we use perplexity (PPL) as the main metric, which is calculated using LLAMA2-13B (Touvron et al., 2023a). For further analysis, we compute the semantic similarity (SS) between the watermarked text and the un-watermarked one using a pre-trained sentence transformer (all-MiniLM-L12-v2) (Reimers and Gurevych, 2019). In Appendix B, we also use two more powerful embedding models, bge-large-en-v1.5 (Xiao et al., 2023) and m3e-large (Wang Yuxin, 2023), to calculate the semantic similarity.

Dataset and Prompt. We select two different datasets, C4 (Raffel et al., 2020) and Essays (Schuhmann, 2023), to validate the effectiveness of our method. For the C4 dataset, we use the first three sentences of each text as a prompt to continue the news report generation. For the Essays dataset, we employ the instructions to guide LLMs in essay composition. For each dataset, we generate 500 samples of 200 tokens using the LLMs and the corresponding prompts. We use the first 200 tokens that follow the prompt in the C4 dataset and the reference answers from the Essays dataset as human-generated texts. In Appendix C, further experiments are conducted on longer outputs.

Hardware. All experiments are conducted using an 80GB A800 GPU. More details of the hyperparameter settings can be found in Appendix E.

5.2 Main Results

Table 2 presents the performance of various methods on the specified model and dataset, with the top-performing results for each metric bolded. Our proposed SSGW method significantly improves the detectability of watermarks and the quality of the generated text compared to the baselines across nearly all models and datasets. To further assess the influence of different watermarking techniques on the text quality, Figure 2 and Figure 3 illustrate the distribution of perplexity and semantic similarity compared to un-watermarked text, respectively. Based on the experimental results, the following will delve into a comparative analysis between our method and various baselines.

Comparison with KGW and SWEET: Both KGW and SWEET employ a fixed predefined δ as the watermark strength. Although SWEET improves upon KGW through entropy-based token filtering, the experimental results indicate little improvement. In contrast, our method leverages semantic information to adjust the watermark logits for different tokens dynamically. This adaptive mechanism not only enhances watermark detectability but also achieves 30% lower perplexity than KGW. In terms of semantic similarity, our method matches the performance of KGW on the C4 dataset and demonstrates superior results on the Essays dataset.

Comparison with ATW: ATW employs temperature scaling to amplify the original logits propor-

Model	Dataset	TPR@0%					Perplexity				
		KGW	SWEET	ATW	DIP	**Ours**	KGW	SWEET	ATW	DIP	**Ours**
OPT-6.7B	C4	0.988	0.998	0.998	0.988	**0.998**	9.045	8.774	7.499	7.866	**6.774**
	Essays	0.994	0.996	1.000	0.978	**1.000**	10.277	9.933	7.673	10.351	**6.822**
GPT-J-6B	C4	0.998	0.996	1.000	0.992	**1.000**	11.336	10.906	8.343	9.313	**7.404**
	Essays	0.996	0.988	1.000	0.992	**1.000**	10.915	10.599	8.026	9.581	**6.364**
LLama2-7B	C4	0.994	1.000	**1.000**	0.980	0.996	8.270	7.901	10.513	7.329	**5.798**
	Essays	0.988	0.992	1.000	0.964	**1.000**	9.014	8.559	8.776	7.459	**5.434**

Table 2: Main results of comparing different watermarking strategies across various datasets and models.

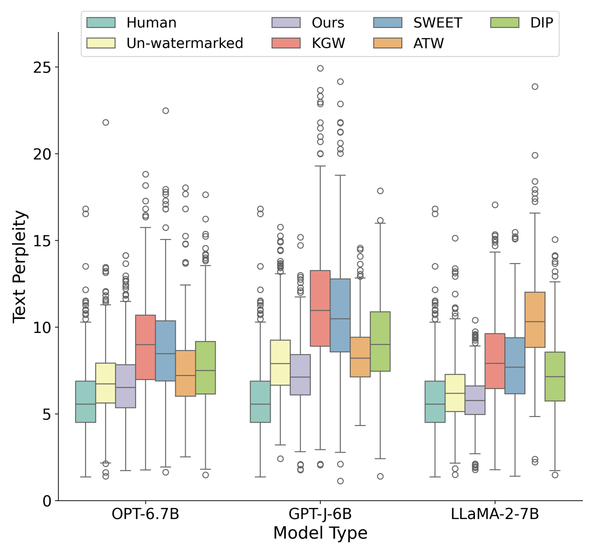

Figure 2: Comparison of text perplexity among human-written text, un-watermarked text, and texts using various watermarking methods conducted on different LLMs for C4 dataset.

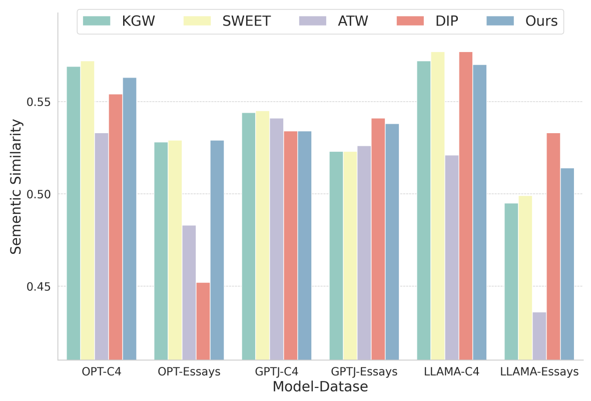

Figure 3: Comparison of semantic similarity between un-watermarked text and texts using various watermarking methods conducted on different LLMs and datasets.

tionally. Although this multiplicative approach contributes to filtering out inappropriate tokens, it also amplifies the impact of watermarked tokens, causing significant interference to the original sampling process. Experimental results reveal that although ATW achieves detection performance comparable to that of our method, it causes catastrophic semantic distortion, as shown in Figure 3. In contrast, our solution addresses this imbalance through semantic guidance that simultaneously improves watermark detectability and text quality.

Comparison with DIP: DIP proposed the α-reweight method, which is theoretically unbiased. However, this unbiased watermarking method requires sacrificing detectability, with DIP consistently exhibiting the lowest true positive rates across all model-dataset configurations. Furthermore, this mathematical property of DIP cannot guarantee a stable improvement in text quality, particularly evident in its performance on OPT-6.7B for Essays. In contrast, our method maintains excellent and stable performance on both detectability and text quality.

5.3 Three-dimensional Trade-off Relationship

In many previous studies that are committed to improving text quality while maintaining detectability, PPL and semantic similarity have been considered as alternative indicators to measure text quality that can be chosen one over the other. However, experimental results in this paper show that there is a deep three-dimensional trade-off relationship among these three key properties. To further explore this trade-off, we perform an in-depth analysis of our method on the C4 dataset.

As presented in Figure 4, an increase in δ_{gui} improves the performance of perplexity, resulting in a corresponding decrease in semantic similarity. This phenomenon arises because higher guidance watermark strength tends to bias the LLM towards selecting tokens consistent with the guidance window text, thereby ignoring important information earlier. Consequently, watermarked texts generated with stronger guidance exhibit lower semantic

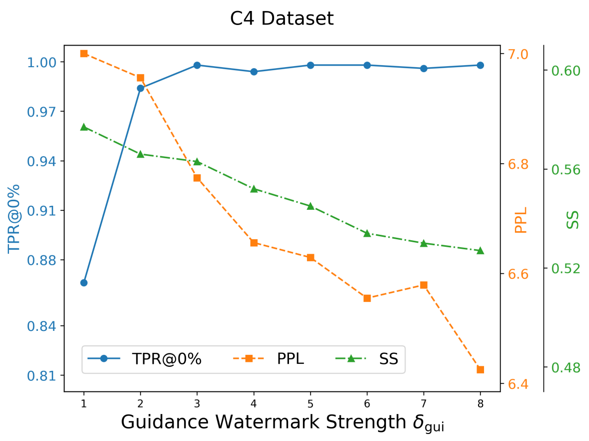

Figure 4: Performance trade-offs of OPT-6.7B on C4 dataset at different guidance watermark strengths (δ_{gui}) with δ_{ini} fixed at 2.0.

Dataset	Metric	SSGW	Removed Mechanisms		
			WLR	DSCU	MPS
C4	TPR@0%	**0.998**	0.998	0.996	0.962
	PPL	6.774	**6.759**	7.224	7.750
	SS	**0.563**	0.560	0.545	0.544
Essays	TPR@0%	**1.000**	0.994	0.990	0.966
	PPL	6.822	**6.703**	7.260	8.329
	SS	**0.529**	0.524	0.504	0.506

Table 3: Main results of the individual impact of different mechanisms, conducted on OPT-6.7B.

similarity to their un-watermarked counterparts.

5.4 Ablation Study

In our approach, we implement three key mechanisms to maintain a stable difference between $l^{(t)}$ and $\hat{l}^{(t)}$ during text generation: Window Length Reset (WLR), Dynamic Suppression Coefficient Update (DSCU), and Maximum Probability Suppression (MPS). To validate their effectiveness, we conduct a comprehensive ablation study by systematically removing each of them. The experimental results, presented in Table 3, clearly demonstrate that removing either the DSCU (which controls suppression strength) or MPS (which removes suppression entirely) causes severe detection failure. The removal of these two mechanisms will directly result in the invalidation of the watermark injection when the guidance vector and the original one have a consensus prediction.

Moreover, the WLR mechanism also proves essential since we observe a decline in performance in terms of detectability and semantic similarity when we discard it. This is primarily because, as the guiding length continues to update, the guiding logit vector will gradually move closer to the original one, thereby losing its guiding significance. Regarding the little improvement in perplexity, we believe the main reason is that our WLR mechanism employs a simple reset approach, which may introduce incoherence at the reset points.

Dataset	Attack	TPR@0%		TPR@1%		TPR@5%	
		KGW	Ours	KGW	Ours	KGW	Ours
C4	CP	**0.934**	0.914	**0.980**	0.972	0.994	**0.994**
	Dipper	0.274	**0.348**	0.504	**0.572**	0.730	**0.760**
Essays	CP	**0.970**	0.930	0.988	**0.990**	0.996	**0.998**
	Dipper	0.434	**0.470**	0.674	**0.704**	0.844	**0.884**

Table 4: Robustness performance against Copy-Paste and Dipper paraphrase attacks.

5.5 Robustness Against Attacks

Considering that the watermarked text is often edited before detection, we evaluate the robustness of our method against two prevalent attack types: Copy-Paste Attack (Kirchenbauer et al., 2023) and Dipper Attack (Krishna et al., 2023). For the Copy-Paste attack, we randomly embed three watermarked text fragments with a length of 20% inside a surrounding un-watermarked text. For the Dipper attack, we use the DIPPER paraphrase model to rewrite the text with the lex diversity set to 60. As shown in Table 4, we report TPR at varying FPR levels, specifically at 0%, 1%, and 5%. Our method performs similarly to KGW under the Copy-Paste attack and demonstrates superior detection accuracy under the Dipper attack, which can be attributed to the fact that paraphrasing tends to preserve semantic information.

6 Conclusion

In this work, we present a novel watermarking method for LLMs called SSGW, which achieves a simultaneous improvement of watermark detectability and text quality through a semantic self-guided approach. Experimental results demonstrate that our method outperforms existing baselines across various models and datasets, especially on text coherence and fluency. Furthermore, a thorough analysis reveals that there is a deep three-dimensional trade-off relationship among different key properties, which provides a new direction for future research.

Limitations

Our method mainly includes two limitations. First, the calculation to obtain a guidance logit vector at each time step during text generation significantly increases computational overhead, resulting in nearly double the generation time (see Appendix D). This makes our method less suitable for real-time applications where efficiency is critical. Second, the experimental results in this paper show that there is a three-dimensional trade-off relationship among different key properties for evaluation, highlighting the need for future investigations. Despite these limitations, we believe that our work contributes positively to the development of high-quality watermarking technology since merely improving watermark detectability is insufficient to address the multifaceted demands of practical applications.

Ethics Statement

Watermarking methods are designed to mitigate the abuse of large language models. However, if the specific watermarking mechanism is leaked, malicious users may use it to escape detection. Therefore, we recommend that all users avoid disclosing specific details to others when using the watermarking method, such as the hash key used in many methods.

References

Mikhail J Atallah, Victor Raskin, Michael Crogan, Christian Hempelmann, Florian Kerschbaum, Dina Mohamed, and Sanket Naik. 2001. Natural language watermarking: Design, analysis, and a proof-of-concept implementation. In *Information Hiding: 4th International Workshop, IH 2001 Pittsburgh, PA, USA, April 25–27, 2001 Proceedings 4*, pages 185–200. Springer.

Isabelle Augenstein, Timothy Baldwin, Meeyoung Cha, Tanmoy Chakraborty, Giovanni Luca Ciampaglia, David Corney, Renee DiResta, Emilio Ferrara, Scott Hale, Alon Halevy, et al. 2024. Factuality challenges in the era of large language models and opportunities for fact-checking. *Nature Machine Intelligence*, 6(8):852–863.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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

Zhiwei He, Binglin Zhou, Hongkun Hao, Aiwei Liu, Xing Wang, Zhaopeng Tu, Zhuosheng Zhang, and Rui Wang. 2024. Can watermarks survive translation? on the cross-lingual consistency of text watermark for large language models. *arXiv preprint arXiv:2402.14007*.

Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. 2024. Unbiased watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

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

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

Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. 2023. Who wrote this code? watermarking for code generation. *arXiv preprint arXiv:2305.15060*.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. 2024a. A semantic invariant robust watermark for large language models.

Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip Yu. 2024b. A survey of text watermarking in the era of large language models. *ACM Computing Surveys*, 57(2):1–36.

Yepeng Liu and Yuheng Bu. 2024. Adaptive text watermark for large language models. *arXiv preprint arXiv:2401.13927*.

Hasan Mesut Meral, Bülent Sankur, A Sumru Özsoy, Tunga Güngör, and Emre Sevinç. 2009. Natural language watermarking via morphosyntactic alterations. *Computer Speech & Language*, 23(1):107–125.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *International Conference on Machine Learning*, pages 24950–24962. PMLR.

OpenAI. 2023a. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

OpenAI. 2023b. New ai classifier for indicating ai-written text. *OpenAI blog*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. 2022. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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. *J. Mach. Learn. Res.*, 21(1).

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Jie Ren, Han Xu, Yiding Liu, Yingqian Cui, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. 2023. A robust semantics-based watermark for large language model against paraphrasing. *arXiv preprint arXiv:2311.08721*.

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*.

Christoph Schuhmann. 2023. essays-with-instructions. `https://huggingface.co/datasets/ChristophSchuhmann/essays-with-instructions`.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Chris Stokel-Walker. 2022. Ai bot chatgpt writes smart essays - should professors worry? *Nature*.

Mercan Topkara, Cuneyt M Taskiran, and Edward J Delp III. 2005. Natural language watermarking. In *Security, Steganography, and Watermarking of Multimedia Contents VII*, volume 5681, pages 441–452. SPIE.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023a. Llama 2: Open foundation and fine-tuned chat models, 2023. *URL https://arxiv.org/abs/2307.09288*.

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

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

He sicheng Wang Yuxin, Sun Qingxuan. 2023. M3e: Moka massive mixed embedding model.

Yihan Wu, Zhengmian Hu, Junfeng Guo, Hongyang Zhang, and Heng Huang. 2024. A resilient and accessible distribution-preserving watermark for large language models. In *Forty-first International Conference on Machine Learning*.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

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

Method	Dataset					
	C4			Essays		
	TPR@0	SS	PPL	TPR@0	SS	PPL
KGW	0.996	0.581	12.14	0.996	0.741	7.572
SWEET	0.988	0.582	11.39	0.992	0.746	7.027
ATW	1.000	0.554	10.71	1.000	0.699	10.180
DIP	0.992	0.570	10.07	0.980	0.741	5.631
OURS	1.000	0.581	7.41	0.996	0.737	5.242

Table 5: Additional experimental results of comparing different watermarking strategies on Qwen2.5-14b, with ppl calculated by qwen2.5-32b.

Embedding Model	Method				
	KGW	SWEET	ATW	DIP	OURS
all-MiniLM-L12-v2	0.581	0.582	0.554	0.570	0.581
bge-large-en-v1.5	0.759	0.760	0.746	0.751	0.756
m3e-large	0.686	0.687	0.667	0.681	0.687

Table 6: Further analysis of the semantic similarity evaluation results using different embedding models. The underlined results represent the worst performance measured by each model.

Method	TPR@0	PPL	SS
KGW	0.992	8.305	0.710
SWEET	1.000	8.106	0.712
ATW	1.000	6.716	0.686
DIP	0.982	7.937	0.679
OURS	1.000	6.440	0.707

Table 7: Performance of generating long outputs of 500 tokens conducted on OPT-6.7B and C4 dataset.

Method	Generation Time	Detection Time
KGW	3.773s	0.054s
SWEET	3.801s	0.073s
ATW	8.608s	4.012s
DIP	3.420s	0.070s
Ours	7.892s	0.050s

Table 8: This table shows the average time taken to generate 200 tokens as well as the average time taken for detection using different methods on OPT-6.7b, measured in seconds.

A Experiments on Qwen2.5-14b

To further showcase the performance of our watermarking algorithm, we conduct experiments on a more powerful LLM, Qwen2.5-14b. Correspondingly, we use Qwen2.5-32b (Bai et al., 2023) for the evaluation of perplexity. As shown in Table 5, our watermarking method significantly improves the perplexity of the generated text while ensuring detectability and semantic similarity compared to other methods.

B Impact of Different Embedding Models

In order to eliminate the influence of different embedding models on the evaluation results of semantic similarity, we selected two additional prominent models, bge-large-en-v1.5 (Xiao et al., 2023) and m3e-large (Wang Yuxin, 2023), for experimentation. The experimental results presented in Table 6 indicate that despite the varying absolute values of semantic similarity measured by different embedding models, the comparisons among various methods exhibit significant consistency in ranking.

C Experiments of Longer Outputs

In the previous experiments, we uniformly set the generation length to 200 tokens, which is consistent with the settings of prior works. Considering that longer generations may occur in real-world applications, we conduct an additional experiment with a fixed generation length of 500 tokens. Note that since the length of 500 exceeds the maximum input length of all-MiniLM-L12-v2, we use m3e-large for SS calculation here. The results presented in Table 7 show that SSGW maintains consistent excellent performance in generating long outputs, which can be attributed to the DGWA mechanism. This adaptive approach ensures that our method remains responsive to the evolving context and maintains its effectiveness.

D Efficiency

In this section, we evaluate the efficiency performance of different watermarking methods. Specifically, for each method, we generate and detect 500 samples of watermarked text, with each sample comprising 200 tokens, and then calculate the average time consumption. Table 8 indicates that our method requires additional time consumption during generation due to the use of the self-guided approach. During the detection process, our method is consistent with KGW and does not require loading additional models like ATW.

E Implement Details

In this section, we will provide the hyperparameters used in the experiments. For KGW and SWEET, we set $\gamma = 0.25$ and $\delta = 2.0$, respectively. The entropy threshold in SWEET is set to 0.9. For ATW, δ is set to 1.5. For DIP, we set $\alpha = 0.45$. The above parameters are derived from the officially recommended default settings. It is worth mentioning that for a fair comparison, we set the prompt to be invisible in SWEET during detection. In addition, we use multinomial sampling with a Top-k of 50 and a Top-p of 1.0. For our method, the default hyperparameters are set as follows: $L_0 = 50$, $\alpha = 0.95$, $\gamma = 0.25$. Before the starting point is identified, we set δ_{ini} the same as used in KGW. In the process of AWLT, we set $\delta_{\text{gui}} = 3.0$.