Context-aware Watermark with Semantic Balanced Green-red Lists for Large Language Models

Yuxuan Guo[1], Zhiliang Tian[1]*, Yiping Song[1], Tianlun Liu[1], Liang Ding[2], Dongsheng Li[1]*

[1]National University of Defense Technology
[2]Zhejiang University

Abstract

Watermarking enables people to determine whether the text is generated by a specific model. It injects a unique signature based on the "green-red" list that can be tracked during detection, where the words in green lists are encouraged to be generated. Recent researchers propose to fix the green/red lists or increase the proportion of green tokens to defend against paraphrasing attacks. However, these methods cause degradation of text quality due to semantic disparities between the watermarked text and the unwatermarked text. In this paper, we propose a semantic-aware watermark method that considers contexts to generate a semantic-aware key to split a semantically balanced green/red list for watermark injection. The semantic balanced list reduces the performance drop due to adding bias on green lists. To defend against paraphrasing attacks, we generate the watermark key considering the semantics of contexts via locally sensitive hashing. To improve the text quality, we propose to split green/red lists considering semantics to enable the green list to cover almost all semantics. We also dynamically adapt the bias to balance text quality and robustness. The experiments show our advantages in both robustness and text quality comparable to existing baselines.

1 Introduction

Large Language Models (LLMs) show their power on text generations but their formidable power may be used for unethical purposes such as plagiarism (Augenstein et al., 2023). Current research injects watermarks into LLMs' generated texts, thereby enabling people to distinguish between LLM-generated text and human-written text. Recent watermark methods inject a unique signature into LLM-generated text, which can only be perceived by watermarking methods, facilitating the detection of whether a text was generated by LLMs.

*Corresponding author

Current watermark methods mainly inject the signature according to "green-red list" (Ren et al., 2023): they divide vocabulary into green/red lists, regard words in green lists as the unique signature, and encourage to generate green tokens, which is easy to be recognized. The methods can be divided into two categories: (1) *Token sampling biasing*-based watermark forces LLMs to select only green tokens during generation. EXP-Edit (Kuditipudi et al., 2023) intervenes in the sampling process of each token. However, forcing LLMs to sample green tokens restricts the semantic richness of LLM-generated text, thus undermining its text quality. (2) To improve the generation quality, researchers further propose *token probability biasing*-based watermark, which enriches the semantics of watermarked texts by introducing a bias to the probability distribution to softly encourage generating green tokens instead of restricting to select green tokens. Takezawa et al. (2023) proposed NS-Mark to constrain the frequency of biasing. Wu et al. (2023) introduced DiPMark to approximate the biased probability distribution to the original one. These methods mitigate the impact of biasing on text quality and ensure superior text quality.

The above methods narrow the gap in text quality between the watermarked text and the unwatermarked text but lack robustness against paraphrasing attacks. Paraphrasing attacks (Krishna et al., 2024) use language models to modify the watermarked text to evade the unique signature of the watermarked text. Specifically, first, the paraphrasing attacks make it difficult for current watermarking methods to match the green/red lists used in watermark injection during the process of watermark detection, causing incorrectly determining whether a token is in the green list; second, the paraphrasing of words turned many green tokens into red tokens, greatly reducing the proportion of green tokens in the text. Paraphrasing attacks make the proportion of green tokens in the attacked watermarked text similar to that of the unwatermarked text, destroying the unique signature of the watermarked text, and leading to detection errors.

Researchers propose to fix the green/red lists (Zhao et al., 2024) in response to the aforementioned problem of mismatched green/red lists caused by paraphrasing attacks: ensuring that the green/red lists used for watermark detection are always consistently aligned with those used for watermark injection even if the watermarked text suffered paraphrasing attacks. This alignment enables accurate identification of tokens within the green list. However, the watermark method introduces the same bias to the probability distribution during each generation step due to the fixed green tokens, restricting the diversity of watermarked text. Researchers have discovered that increasing the proportion of green tokens in the watermarked text can maintain a sufficiently high ratio of green tokens even when paraphrasing attacks reduce their count, thus ensuring the detection of the unique signature, which effectively mitigates the problem described above of reducing the number of green tokens caused by paraphrasing attacks. Current works propose to cause greater bias to the probability distribution to increase the sampling probability of green tokens during watermark injection, enlarging the proportion of green tokens (Kirchenbauer et al., 2023a). However, a greater bias leads to more significant disparities between the perturbed probability distribution and the original one, thereby resulting in a degradation of text quality.

In this paper, to balance text quality and robustness against paraphrasing attacks, we propose a LLM-based semantic-aware watermark method that considers contexts to generate a semantic-aware key to split a semantic balanced green/red lists for watermark injection. Those green/red lists ensure the semantic distribution of green tokens to be very similar to the distribution of the whole vocabulary, which highly reduces the performance drop due to introducing biases to encourage the green tokens. Specifically, to improve the robustness against paraphrasing attacks, we propose the context-aware semantic-based watermark key generator with local sensitive hashing (LSH). It prevents the paraphrasing attack from maliciously replacing tokens and thus changing the watermark key to mislead the watermark detection. To improve the text quality, we propose a semantic-based green-red lists split method, which enables the green lists to cover almost all semantics and ensures the distribution among green and red lists is balanced. It avoids the bias on green lists and reduces the text quality. To balance the text quality and robustness, we propose an entropy-based dynamic bias adaptation module, dynamically adjusting the bias during generation. The experimental results validate our method's effectiveness in the robustness against paraphrasing attacks and text quality.

Our contributions are: (1) We propose a semantic-based watermark method for LLMs to enhance the text quality and robustness against paraphrasing attacks. (2) We obtain the green/red lists based on semantics, making green lists cover almost all semantic spaces and obtain a balanced semantics distribution green-red list. (3) Experiments show our method outperforms baselines on text quality, watermarked text detection, and robustness against paraphrasing attacks.

2 Related Works

2.1 Watermarking on Generated Text

Watermarking safeguards textual content inconspicuously with stable embedding (Kamaruddin et al., 2018). Some researchers used the watermark techniques to protect the privacy of user data (Song et al., 2024, Tian et al., 2022). In this paper, we mainly discuss using watermarks to help people distinguish between LLM-generated text and human-written text. Initially, the focus was on integrating watermarks into existing texts. Current watermark methods for generated text consist of: (1) Format-based watermark methods, that integrate a watermark within the text format. Al-maweri et al. (2016) proposed embedding watermarks with Unicode extended characters using a predefined encoding table. Alotaibi and Elrefaei (2018) inserted pseudo spaces within Arabic texts for watermark embedding. Por et al. (2012) selectively inserted Unicode spaces for encoding external information. (2) Lexical-based watermark methods, which replace the tokens with watermarked tokens with similar semantics. Topkara et al. (2006b) proposed token substitution with prioritized synonyms based on resilience criteria in the generated text. Yang et al. (2022) introduced a scheme of using BERT for context-aware lexical substitution. He et al. (2022) proposed optimizing word selection variability to mitigate watermark vulnerability. (3) Syntactic-based watermark methods, which embed the watermark into the generated text's syntax.

Atallah et al. (2001) used syntax transformations to embed the watermark. Topkara et al. (2006a) enhanced the previous method with additional syntax transformations.

2.2 Watermarking for LLMs' generation

There are two ways to inject the watermark into the text generated by LLMs. (1) **Token sampling biasing** refers to forcing the model to sample green tokens. Christ et al. (2023) devised an undetectable watermark detectable only with the key. Hou et al. (2023) introduced sentence-level watermarking during sampling. Giboulot and Teddy (2024) introduced watermark into token chunks, encouraging the model to sample watermarked text satisfying text quality. (2) **Token probability biasing** refers to increasing the probability of the model sampling green tokens. Kirchenbauer et al. (2023a) propose splitting green-red lists for tokens pre-generation, softly prompting green token use during sampling for watermark injection. Zhao et al. (2024) developed prior methods by fixing green/red lists, and injecting watermarks into the next token's probability distribution at each generation step. Hu et al. (2023) introduced an unbiased reweighting method for watermarking without altering token probability distribution. Lee et al. (2023) devised a selective watermarking method, thereby alleviating the degradation of LLM-generated code. Takezawa et al. (2023) represented text quality degradation due to watermarking as a constrained optimization problem by adjusting green token proportions in the generated text. Yoo et al. (2024) introduced a multi-bit watermark method using positional allocation to inject traceable information. Fernandez et al. (2023) employed cyclic shifts and a shared watermark key to generate multiple watermark versions, each representing distinct watermark messages.

2.3 Paraphrasing Attacks on Watermark

Paraphrasing attacks modify the watermarked text, disrupting its unique signature and causing misclassification as unwatermarked. Early paraphrasing attacks relied on round-trip translation (Yang et al., 2023), translating the text into another language and back. Ueoka et al. (2021) employed Masked Language Models (MLMs) to replace words while maintaining text quality, enhancing paraphrasing attack effectiveness. Researchers found that text summarization models simplify the text, potentially enhancing attack effectiveness (Hou et al., 2023). Krishna et al. (2024) finetuned a paraphrasing model, undermining the watermark's effectiveness.

To enhance robustness against paraphrasing attacks, several strategies have been proposed: Zhao et al. (2024) expanded existing methods by employing a fixed green/red lists strategy. Ren et al. (2023) introduced a technique of splitting the green/red list based on the discretized result from continuous semantic spaces. Kuditipudi et al. (2023) developed a watermarking method biasing token sampling using edit distance during watermark detection.

3 Method

3.1 Overview

The general framework of watermark methods includes two stages: watermark injection and watermark detection. During every generation step, most watermark methods first get a watermark key and then partition the vocabulary to get the green/red lists based on the watermark key. They introduce a bias into the probability to encourage the generation of tokens from the green list. Following this framework, we propose (1) **context-aware semantic-based watermark key generator** (Sec. 3.2), which generates watermark key considering semantics in the contexts to improve the robustness; (2) **semantic-aware green/red lists split** (Sec. 3.3), which splits vocabulary into green or red lists based on the semantic, ensuring the diversity of the green list; (3) **entropy-based dynamic bias adaptation** (Sec. 3.4), which adaptively adjusts the bias.

Our framework first employs the watermark key generator (Sec. 3.2) to obtain semantic-based watermark keys for splitting the green/red lists (Sec. 3.3). Then, we conduct bias based on the green/red lists. We dynamically add adaptive bias (Sec. 3.4) for perturbations and filter some tokens hard to conduct bias (see App. A due to page limitation).

3.2 Context-aware Semantic-based Watermark Key Generator

To generate a suitable watermark key to defend against paraphrasing attacks, we propose a *Context-aware Semantic-based Watermark Key Generator*. It utilizes the semantics of the context to generate the watermark key.

Current watermark methods generate the watermark key by feeding the context tokens for hashing without considering the semantics, thus similar words can not share the same key. Paraphrasing attacks change the contexts by replacing tokens with semantically similar tokens, resulting in

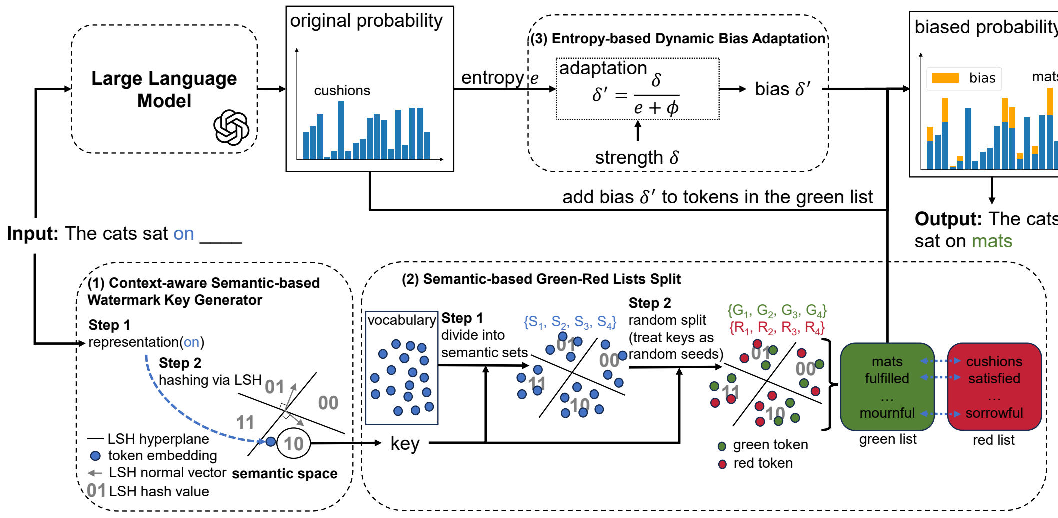

Figure 1: An overview of our method. At each generation step, the (1) Key Generator (lower branch) applies LSH to hash tokens in the vocabulary into hash key according to the semantics of "on"; the (2) Green-Red List Split splits green-red list for each divided semantic set. In the upper branch of each generation step, the LLM generates as usual, then the (3) Bias Adaptation dynamically obtains bias according to the entropy. Finally, the model adds the bias on the generation distributions of green list tokens and then generates the next token "mats".

changing the watermark key. Changing watermark keys causes the change of green/red list, which further misleads the watermarked detection (judging the watermarked text as an unwatermarked one) [1].

Hence, we construct a semantic-based watermark key to assign the same watermark key to tokens with similar semantics via local sensitive hashing (LSH) (Indyk and Motwani, 1998). This ensures replacing with similar tokens in paraphrasing attacks may not result in the change of watermarked keys, since similar tokens may have the same key. Particularly, at each generation step, the processing consists of two steps as the bottom left corner of Fig. 1: (1) **Representation**. We represent the context semantic with an embedding: we treat the last token as the context and feed it into the embedding layer of the LLM to obtain the last token embedding. (2) **Hashing via LSH**. We obtain the hash value of current step according to the last token embedding via LSH, which hashes similar inputs into the same value, serving as the watermark key corresponding to each token.

LSH hashes similar textual inputs into the same hash value, making it viable to get the watermark key from the semantic. We follow the cosine-preserving method (Weir et al., 2020, Guu et al., 2018 and Charikar, 2002). This method uses d random hyperplanes to split the semantic space, which specifies d hyperplanes represented by corresponding normal vector $r^{(i)}$ that is randomly drawn from the Gaussian distribution with the same dimension as the token embedding v [2]. For the i-th hyperplane, we get the dot product between the token embedding and its normal vector $r^{(i)}$ and use an indicator function $\mathbb{1}(\cdot)$ to get the result that represents the side of the hyperplane that the token embedding falls to.

$$LSH_i(v) = \mathbb{1}(r^i \cdot v \geq 0) \qquad (1)$$

After projection on d hyperplanes, we get a d-bit binary value, which represents the hash value. At each generation, we use the hash value of the previous token as the watermark key for splitting the green/red lists of the next word, which can be obtained from the text itself to reproduce the split results of each generation step.

After obtaining the watermark keys of all tokens, we construct the mapping from tokens to watermark keys and store this mapping. In watermark injection and detection processes, we directly retrieve the watermark key from the mapping given the context tokens to avoid the practical issues of calling the watermarked LLM during detection.

[1] Using different green/red lists to determine the next token will randomize the detection result, causing the number of green tokens in the watermarked text similar to that in the unwatermarked text, resulting in missing detection.

[2] Normal vector $r^{(i)}$ signifies the hyperplane that is perpendicular to $r^{(i)}$ and pass through the origin.

3.3 Semantic-based Green-Red Lists Split

To ensure the green lists cover almost all semantics, we propose *Semantic-based Green-Red Lists Split* to split green/red lists based on the sets of tokens with similar semantics.

Current methods directly partition the vocabulary randomly into green/red lists seeded by the watermark key. The arbitrary partitioning over the vocabulary ensures that each word has an equal probability of being selected as a green token. However, this split method cannot guarantee that tokens with similar semantics are balanced distributed between the green list and the red list at every generation step. That imbalanced distribution among similar tokens makes it difficult for the green tokens to cover almost all semantics, which makes it hard for the model to select desired tokens from green lists to express the desired semantics thus degrading the quality of generated texts.

Hence, we get the green/red lists of the vocabulary by splitting the green/red lists from the sets of tokens with similar semantics and merging these lists, which achieves a balanced distribution of the tokens with similar semantics in the green/red lists for every generation step. The processing consists of three steps: (1) **Divide into semantic sets**. Based on the analysis of LSH in Sec. 3.2, tokens with the same hash value can be regarded as tokens with similar semantics. At each generation step, we divide the vocabulary into semantic sets based on the hash value and get all semantic sets of tokens with similar semantics $\{S_1, S_2, ..., S_n\}$.(2) **Randomly split into green/red lists**. At each generation step, for the i-th semantic set S_i, we randomly split the set to get the green list G_i and the red list R_i, where we treat the semantic-based key from Sec. 3.2 as the seed of pseudo-random function. Employing a semantic-based key as the seed is crucial since watermark algorithm requires the detection, and injection with the same contexts should share a same green/red list and the watermark key relying on context semantics ensures detection and injection can get the same key to obtain a same green/red list. (3) **Gather to obtain whole green/red lists**. Now we merge green lists $\{G_i\}$ from the semantic sets to get a whole green list for the vocabulary G through $G = G \cup G_i$ and red lists $\{R_i\}$ into a whole red list R by $R = R \cup R_i$. The merged green/red G and R list will be used in adding biases into the generation (mentioned in Sec. 3.5).

This approach guarantees that similar tokens are balanced distributed on the green/red lists and makes green lists cover all lived semantics of the semantic space (i.e. vocabulary), which is aligned with the LSH's semantic space [3]. It means that the gap between the semantic distribution of green lists and that of the entire vocabulary is quite small. It results in adding a bias to obtain green tokens does not lead to a large semantic shift, guaranteeing the semantic coherence of the generated text when sampling green tokens.

3.4 Entropy-based Dynamic Bias Adaptation

To balance text quality and robustness against paraphrasing attacks, we propose *Entropy-based Dynamic Bias Adaptation* to modify bias dynamically according to the entropy for each generation step.

Current watermark methods inject a bias into the probability distribution. A large bias improves robustness against paraphrasing attacks but causes a low text quality due to the drastic impact. A low bias introduces a minor impact on the distribution but can improve the text quality. Current methods mostly use a fixed bias and lack dynamic adjustment for the bias to influence the biasing effect according to the requirements. The fixed bias cannot meet the changing need for each generation step, making it difficult for watermark methods to improve robustness against paraphrasing attacks while preserving text quality.

We introduce a dynamic adaptation mechanism for the bias that scales the bias dynamically based on the entropy of generated tokens. Following Kirchenbauer et al. (2023a), we use spike entropy to quantify the uncertainty of the distribution, which reflects the ease of sampling green tokens.

We use the reciprocal function for entropy to form an inverse relationship with entropy. To reduce the bias when the entropy is extremely high, we then introduce a scalar ϕ as a balance factor to control the maximal value of reciprocal value, which will cause a reduction of the bias δ when the entropy is high.

$$\delta'(s) = \delta \cdot \frac{1}{entropy(s) + \phi} \tag{2}$$

We adjust the bias dynamically adaptation according to the entropy: at low entropy, a low bias aimed at preserving text quality fails to sustain

[3]LSH has processed the token embeddings of all tokens, making the hash value can reflect the semantic similarity of the tokens in the semantic space.

the sampling probability of green tokens. Consequently, we use the dynamical adaption to increase the bias to elevate the sampling probability; when the entropy is high, the large bias used to preserve the sampling probability of green tokens will cause a severe impact on the probability distribution. Thus, we reduce the bias dynamically to mitigate the impact of the bias. Our method solves the inability to adapt to bias requirements in high and low entropy environments due to the fixed bias.

3.5 Workflow of Watermark Injection and Detection

For watermark injection, at each generation step, we first use the Context-aware Semantic-based Watermark Key Generator (Sec. 3.2) to generate watermark key based on the semantics of the context. Then, we employ Semantic-aware green/red lists Split (Sec. 3.3) to get the green/red list. Before conducting bias, we get the entropy from the next token's probability distribution, and use Entropy-based Dynamic Bias Adaptation (Sec. 3.4) to adjust the bias. The procedure of injection can be found in Algorithm 1 in appendix. We introduce Entropy-based Token Filter module in App. A.

For watermark detection, given a text, for each token, we obtain a watermark key from the context and split the green/red lists based on the key following the process of injection to determine whether the token falls into the green list. We count the number of green tokens T, and calculate z-score as:

$$z = \frac{T - \gamma N}{\gamma(1-\gamma)N} \tag{3}$$

where γ is the percentage of green list in entire vocabulary and N is the number of tokens. Higher z-score provides more confidence in determining whether the text is watermarked. We expand the explanation of watermark detection in App. B. The detection procedure is in Algorithm 2 in appendix.

4 Experiments

4.1 Experiment Settings

Dataset. Following previous works (Hou et al., 2023, Kirchenbauer et al., 2023a, Kuditipudi et al., 2023), we randomly sampled 500 samples from the RealNews subset of the C4 dataset (Raffel et al., 2020), which contains a variety of news articles.

Baselines. Our baselines consist of the following watermark methods: (1) KGW / KGW-Large (Kirchenbauer et al., 2023a), which split the green/red lists based on the watermark key hashed from the previous token to inject the watermark; (2) Unigram watermark (Zhao et al., 2024), which use a fixed green/red lists to improve the robustness against paraphrasing attacks; (3) SWEET (Lee et al., 2023), which reduce the number of bias to improve the text quality; (4) EWD (Lu et al., 2024), which gives weights to tokens based on their entropy to improve the robustness against paraphrasing attacks; (5) EXP-Edit (Kuditipudi et al., 2023), which bias the token sampling process to improve the robustness against paraphrasing attacks (See implication details in App. C).

Evaluation Metrics. Following the previous works (Liu and Bu, 2024, Ren et al., 2023), our methods consist of: (1) Area Under the Receiver Operating Characteristic curve (AUROC). AUROC evaluates the performance of classification results based on the True Positive Rate (TPR) and the False Positive Rate (FPR) at various thresholds; (2) TPR@5%FPR, which represents the ratio of watermarked text that is detected correctly when 5% of unwatermarked texts are misclassified as watermarked text. (3) Best F1 score, which represents the F1 score provided with the optimal TPR and FPR during detection; (4) Perplexity. we use the perplexity to measure the quality of the generated texts. We use OPT-2.7B (Zhang et al., 2022) to calculate the perplexity of the text.

Paraphrasing attacks setup. Following Zhao et al. (2024), we test the detectability of the paraphrased watermarked text since people tend to use the generated text after paraphrasing it rather than directly using it. We use two types of paraphrasing attacks to modify the watermarked text, including Pegasus (Zhang et al., 2020) and Dipper (Krishna et al., 2024). Pegasus is a language model that simplifies the watermarked text. Dipper is a model with 11B parameters fine-tuned for paraphrasing, causing a significant modification of the text. For Pegasus, we paraphrase the watermarked text through beam search with 25 beams. For Dipper, we follow the same parameter setting in Kirchenbauer et al. (2023b), with the lex diversity of 60.

4.2 Overall Performance on Detectability

Table 1 shows the detectability of the original watermarked text (**No Attack**) and robustness against different paraphrasing attacks (**Pegasus Attack** and **Dipper Attack**) in various watermark methods. The detectability of the original watermarked texts among various watermark methods (**No Attack** in

Method	No Attack			Pegasus Attack			Dipper Attack		
	TPR@5%FPR (↑)	Best F1 (↑)	AUROC (↑)	TPR@5%FPR (↑)	Best F1 (↑)	AUROC (↑)	TPR@5%FPR (↑)	Best F1 (↑)	AUROC (↑)
KGW	0.9960	0.9940	0.9993	0.8480	0.9021	0.9298	0.5380	0.7947	0.8693
KGW-Large	0.9980	0.9950	0.9969	0.8980	0.8980	0.9486	0.5380	0.8230	0.9045
Unigram	0.9920	0.9970	0.9989	0.9120	0.9460	0.9743	0.6600	0.8379	0.9115
SWEET	0.9840	0.9889	0.9975	0.9220	0.9228	0.9695	0.5360	0.7907	0.8593
EWD	0.9960	0.9950	0.9943	0.9140	0.8891	0.9189	0.5060	0.7773	0.8492
EXP-Edit	0.9980	0.9947	0.9968	0.8860	0.9216	0.9452	0.5460	0.8407	0.8986
Ours	**0.9980**	**0.9980**	**0.9998**	**0.9380**	**0.9545**	**0.9773**	**0.7880**	**0.8742**	**0.9188**

Table 1: Performance comparison on different methods, including cases with no attack and two paraphrasing attacks. The detectability of the cases with two paraphrasing attacks represents the performance of robustness.

Settings	No Attack			Paraphrasing Attack			Text Quality
	TPR@5%FPR (↑)	Best F1 (↑)	AUROC (↑)	TPR@5%FPR (↑)	Best F1 (↑)	AUROC (↑)	Perplexity (↓)
Ours (Full Model)	**0.9980**	**0.9980**	**0.9998**	0.9380	**0.9545**	**0.9773**	6.1880
w/o Watermark Key	0.9940	0.9960	0.9992	0.9100	0.9326	0.9564	6.2432
w/o Green-Red Lists	0.9960	0.9960	0.9982	**0.9402**	0.9482	0.9738	6.7048
w/o Dynamic Bias	0.9920	0.9869	0.9976	0.9200	0.9431	0.9600	**6.0649**

Table 2: Performance comparison of robustness and text quality after the removal of different components.

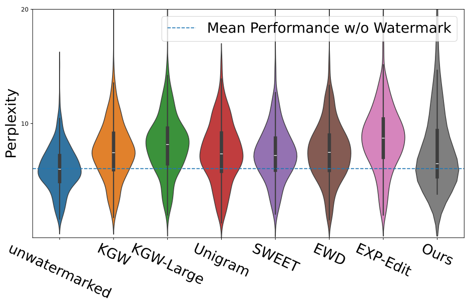

Figure 2: Violin plot of Text PPL over all methods.

Table 1) proves the effectiveness of current watermark methods in watermark detection since all watermark methods demonstrate effective performance. Robustness against paraphrasing attacks is represented by the detectabilities for the watermarked texts under two paraphrasing attacks in **Pegasus Attack** and **Dipper Attack** rows of Table 1. The results demonstrate our method still keeps a relatively high detectability while the detectability of most baselines significantly deteriorated, which indicates the outperformance of our method in robustness against paraphrasing attacks since our method obtained the watermark key based on the semantics of the context and increased the number of green tokens in the original watermark text. We test the time consumption among different watermark methods during watermark injection and detection in App. D.

4.3 Overall Performance on Text Quality

In Fig. 2, we compare the quality of generated text by calculating text perplexity (PPL) on different watermark methods. We observe that our method obtains similar perplexity to that of the unwatermarked text, which shows our watermark has almost no influence on the generated quality. This performance can be attributed to our semantic-based green/red lists allowing the model to sample the desired tokens in the green list, which narrows the gap in the semantics between the watermarked text and the unwatermarked text.

4.4 Ablation Studies

In Table 2, we conduct ablation studies by removing the proposed modules one by one to verify their effectiveness. We use Pegasus Attack as a typical example of paraphrasing attacks. The row 2 demonstrates that the deletion of the Semantic-based Watermark Key Generator (Sec. 3.2) worsens the robustness performance, which indicates that the semantic-aware key plays an important role in improving the robustness against paraphrasing attacks. The removal of the Semantic-based Green-Red Lists Split (Sec. 3.3) increases perplexity in the watermarked text, which proves the semantic-based green/red lists help our method have a better performance in text quality. We also find that removing the semantic-based green/red lists worsens the robustness against paraphrasing attacks since our green/red lists have a more uniform distribution of the semantically similar tokens, resulting in paraphrased tokens being more likely to fall on the green list, maintaining the proportion of green tokens. After removing the Dynamic Bias Adaptation (Sec. 3.4), text quality increases slightly but robustness against paraphrasing attacks drops much, which implies that the dynamic bias balances the text quality and robustness against paraphrasing attacks. Our method performs worse perplexity

Method	No Attack			Pegasus Attack			Dipper Attack			Text Quality
	TPR@5%FPR (↑)	Best F1 (↑)	AUROC (↑)	TPR@5%FPR (↑)	Best F1 (↑)	AUROC (↑)	TPR@5%FPR (↑)	Best F1 (↑)	AUROC (↑)	Perplexity (↓)
KGW	0.9940	0.9940	0.9995	0.7460	0.8731	0.9389	0.3080	0.7233	0.7777	4.2932
KGW-Large	0.9980	0.9970	0.9998	0.9140	0.9429	0.9503	0.4100	0.7712	0.8379	4.6295
Unigram	0.9780	0.9982	0.9927	0.9040	0.9261	0.9180	0.6040	0.8074	0.8885	4.2104
EXP-Edit	1.0000	0.9990	1.0000	0.8923	0.9431	0.9482	0.3768	0.7675	0.8328	4.9284
SWEET	1.0000	0.9980	1.0000	0.7273	0.7692	0.9407	0.3490	0.7599	0.8060	4.2156
EWD	0.9980	0.9940	0.9998	0.7273	0.5556	0.9298	0.3440	0.7472	0.7987	4.2330
Ours	**1.0000**	**0.9990**	**1.0000**	**0.9260**	**0.9454**	**0.9522**	**0.7040**	**0.8316**	**0.9113**	**4.0928**

Table 3: Performance comparison on different methods based on Llama-2-7b, including cases with no attack and two paraphrasing attacks. Perplexity is calculated by Llama-2-13b.

compared to the method removing dynamic bias due to our dynamic bias module focusing on improving robustness against paraphrasing attacks while slightly decreasing text quality.

We also conducted the performance comparison on different methods with Llama-2-7b (Touvron et al., 2023) as the backbone model. The result can be found in Table 3 with analysis in App. E.

4.5 Semantic Coverage of Green List

To validate that our semantic-based green/red lists provide more comprehensive semantics, we conducted an experiment comparing the average semantic similarity of the Top-K green tokens between our method (semantic-based green/red lists) and the KGW method (traditional green/red lists).

Specifically, for each step, we random sample a token from the vocabulary and find the Top-K green tokens that have the highest semantic similarity with the sampled token, and compute their average similarity[4]. The higher metric reflects the better semantic coverage of the green list. Table 4 shows that our method (semantic-based green list) has higher similarities than the KGW method (traditional green list) across different K values. This reflects that our green list is more semantically comprehensive. For a given token, a semantic-based green list can provide highly semantically similar tokens, indicating rich semantic coverage. This demonstrates that our green list covers almost all semantic space of the vocabulary, performing more comprehensive coverage of the semantics.

4.6 Semantic Distribution between Green and Red Lists

To confirm that the distribution of semantically similar tokens in our semantic-based green/red lists is more uniform, we conducted an experiment to compare the distribution of our method (semantic-based green/red lists) to that of the KGW method (traditional green/red lists). The experimental settings are identical to those in Section 4.5.

[4]The percentage of green tokens is set to 0.25. To evade randomness, we set test times to 200 and K to 5, 10, 20 or 50. We use the cosine similarity of token embeddings to measure the semantic similarity between tokens.

Settings	Method	Semantic Similarity (↑)
K=5	KGW	0.529
	Ours	**0.542 (+2.40%)**
K=10	KGW	0.528
	Ours	**0.535 (+1.31%)**
K=20	KGW	0.505
	Ours	**0.507 (+0.39%)**
K=50	KGW	0.484
	Ours	**0.485 (+0.21%)**

Table 4: Comparison of semantic comprehensiveness. Higher Similarity indicates comprehensiveness.

Settings	Method	Standard Deviation (↓)
K=5	KGW	0.1997
	Ours	**0.1887 (-5.83%)**
K=10	KGW	0.1456
	Ours	**0.1357 (-7.29%)**
K=20	KGW	0.0937
	Ours	**0.0935 (-0.21%)**
K=50	KGW	0.0621
	Ours	**0.0576 (-7.81%)**

Table 5: Comparison of semantic distribution. Lower Standard Deviation indicates more uniform distribution.

Specifically, for each step, we randomly split the green/red lists and find the Top-K tokens with the highest semantic similarity with a fixed token and determine the proportion of green tokens among these K tokens. We then calculate the standard deviation to analyze the distribution. Table 5 presents that our method (semantic-based green/red lists) achieves lower standard deviations compared to the KGW method (traditional green/red lists) across different K values, which suggests a more uniform distribution of semantically similar tokens in our semantic-based green/red lists. This uniform distribution implies that the frequency of tokens from various semantics appearing more evenly in our semantic-based green/red lists. Consequently, the watermarked text based on our approach is semantically closer to the unwatermarked text.

We also present the robustness against paraphrasing attacks of different numbers of LSH hyperplanes in Table 9 with analysis in App. F.

5 Conclusion

In this paper, we propose a semantic-based watermark method for LLMs that balances text quality and robustness against paraphrasing attacks. Our approach effectively retrieves the semantic watermark key and ensures coverage among semantically similar tokens in the green list while reducing the semantic distribution gap between the green list and the entire vocabulary. This allows the model to sample desired text in the green list, enhancing text quality. Dynamic bias adaption addresses fixed bias adaptation limitations. The experiments show our method excels in robustness against paraphrasing attacks and significantly improves text quality.

Limitations

Locality-Sensitive Hashing (LSH) algorithm is a relatively old-fashioned method for gathering semantically similar tokens. Some advanced methods can perform better in splitting the sets of semantically similar tokens. Nevertheless, our method regards the LSH method as a module that can be easily replaced with other advanced methods.

The dataset and backbone model we utilized in the experiment are comparatively small. The RealNews subset from C4 dataset (Raffel et al., 2020) and OPT-1.3B (Zhang et al., 2022) are recognized as benchmark standards, widely used by numerous studies (Kirchenbauer et al., 2023a, Hou et al., 2023, Wang et al., 2023, Liu et al., 2023). Although our method is relatively independent and can easily adapt to new datasets and models, we choose these benchmarks for fair comparison.

We only tested our watermarking method in an English environment, lacking validation in multilingual contexts. However, our token-based watermark design exhibits high compatibility with various languages. We conducted our experiments on the datasets composed of English corpora to align with existing watermark methods.

Ethical Considerations

Privacy: Watermarking technology does not present ethical concerns. On the contrary, watermarking can enhance the accountability of large language model API access by tracking malicious users, without infringing on individual user privacy.

Human Resources: As our research does not involve manual annotation, there is no risk of labor exploitation, such as forcing employees to overwork or paying them below-market wages.

Watermark Application: The advanced capabilities of LLMs have greatly increased the need for detecting LLM-generated texts. We advocate for the integration of watermarking methods into models to improve the governance of LLMs. While our method demonstrates excellent robustness and text quality, the watermark remains vulnerable to paraphrasing attacks from advanced language models and thus should not be overly relied on. We remind users to pay attention to the above issue.

Acknowledgments

This work is supported by the following fundings: Young Elite Scientist Sponsorship Program by CAST (2023QNRC001) under Grant No. YESS20230367, the National Natural Science Foundation of China under Grant No. 62306330, No. 62106275, No. 62025208, No. 62421002, and the Grant of No. WDZC20235250103.

References

Nasraddin Ahmed Salem Al-maweri, Wan Azizun Wan Adnan, Abdul Rahman Ramli, Khairulmizam Samsudin, and Sharifah Mumtazah Syed Ahmad Abdul Rahman. 2016. Robust digital text watermarking algorithm based on unicode extended characters. *Indian Journal of Science and Technology*.

Reem A Alotaibi and Lamiaa A Elrefaei. 2018. Improved capacity arabic text watermarking methods based on open word space. *Journal of King Saud University-Computer and Information Sciences*, 30(2):236–248.

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, Eduard Hovy, Heng Ji, Filippo Menczer, Ruben Miguez, Preslav Nakov, Dietram Scheufele, Shivam Sharma, and Giovanni Zagni. 2023. Factuality challenges in the era of large language models. *Preprint*, arXiv:2310.05189.

Moses S Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of* *the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388.

Miranda Christ, Sam Gunn, and Or Zamir. 2023. Undetectable watermarks for language models. *arXiv preprint arXiv:2306.09194*.

Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. 2023. Three bricks to consolidate watermarks for large language models. In *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE.

Eva Giboulot and Furon Teddy. 2024. Watermax: breaking the llm watermark detectability-robustness-quality trade-off. *arXiv preprint arXiv:2403.04808*.

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

Xuanli He, Qiongkai Xu, Yi Zeng, Lingjuan Lyu, Fangzhao Wu, Jiwei Li, and Ruoxi Jia. 2022. Cater: Intellectual property protection on text generation apis via conditional watermarks. *Advances in Neural Information Processing Systems*, 35:5431–5445.

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

Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. 2023. Unbiased watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

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

Nurul Shamimi Kamaruddin, Amirrudin Kamsin, Lip Yee Por, and Hameedur Rahman. 2018. A review of text watermarking: theory, methods, and applications. *IEEE Access*, 6:8011–8028.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023a. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. 2023b. On the reliability of watermarks for large language models. In *The Twelfth International Conference on Learning Representations*.

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

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2023. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*.

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, Shuang Li, Lijie Wen, Irwin King, and S Yu Philip. 2023. A private watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

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

Yijian Lu, Aiwei Liu, Dianzhi Yu, Jingjing Li, and Irwin King. 2024. An entropy-based text watermarking detection method. *arXiv preprint arXiv:2403.13485*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Lip Yee Por, KokSheik Wong, and Kok Onn Chee. 2012. Unispach: A text-based data hiding method using unicode space characters. *Journal of Systems and Software*, 85(5):1075–1082.

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. *Journal of machine learning research*, 21(140):1–67.

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*.

Yiping Song, Juhua Zhang, Zhiliang Tian, Yuxin Yang, Minlie Huang, and Dongsheng Li. 2024. Llm-based privacy data augmentation guided by knowledge distillation with a distribution tutor for medical text classification. *Preprint*, arXiv:2402.16515.

Yuki Takezawa, Ryoma Sato, Han Bao, Kenta Niwa, and Makoto Yamada. 2023. Necessary and sufficient watermark for large language models. *Preprint*, arXiv:2310.00833.

Zhiliang Tian, Yingxiu Zhao, Ziyue Huang, Yu-Xiang Wang, Nevin L. Zhang, and He He. 2022. Seqpate: Differentially private text generation via knowledge distillation. In *Advances in Neural Information Processing Systems*, volume 35, pages 11117–11130. Curran Associates, Inc.

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

Umut Topkara, Mercan Topkara, and Mikhail J Atallah. 2006b. The hiding virtues of ambiguity: quantifiably resilient watermarking of natural language text through synonym substitutions. In *Proceedings of the 8th workshop on Multimedia and security*, pages 164–174.

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. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Honai Ueoka, Yugo Murawaki, and Sadao Kurohashi. 2021. Frustratingly easy edit-based linguistic steganography with a masked language model. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5486–5492, Online. Association for Computational Linguistics.

Lean Wang, Wenkai Yang, Deli Chen, Hao Zhou, Yankai Lin, Fandong Meng, Jie Zhou, and Xu Sun. 2023. Towards codable text watermarking for large language models. In *The Twelfth International Conference on Learning Representations*.

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

Yihan Wu, Zhengmian Hu, Hongyang Zhang, and Heng Huang. 2023. Dipmark: A stealthy, efficient and resilient watermark for large language models. *Preprint*, arXiv:2310.07710.

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

Xi Yang, Jie Zhang, Kejiang Chen, Weiming Zhang, Zehua Ma, Feng Wang, and Nenghai Yu. 2022. Tracing text provenance via context-aware lexical substitution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11613–11621.

KiYoon Yoo, Wonhyuk Ahn, and Nojun Kwak. 2024. Advancing beyond identification: Multi-bit watermark for large language models. *Preprint*, arXiv:2308.00221.

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

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 Vijendra Ananth, Lei Li, and Yu-Xiang Wang. 2024. Provable robust watermarking for AI-generated text. In *The Twelfth International Conference on Learning Representations*.

A Entropy-based Token Filter

To solve the problem where biasing low-entropy tokens will worsen text quality, we propose the *Entropy-based Token Filter* to ignore low-entropy tokens and not inject watermarks into them, which preserves their probability distribution and maintains the semantic, improving text quality of the watermarked text.

As described in Sec. 3.4, the probability distribution with low entropy means one or a very few tokens account for a large proportion of the probability, making it difficult to sample the green tokens if these high-probability tokens are not in the green list. It is feasible to introduce a large bias in the probability distribution and enlarge the probability of sampling green tokens. However, the drastic

Settings	No Attack			Paraphrasing Attack			Text Quality
	TPR@5%FPR (↑)	Best F1 (↑)	AUROC (↑)	TPR@5%FPR (↑)	Best F1 (↑)	AUROC (↑)	Perplexity (↓)
Ours	**0.9980**	**0.9980**	**0.9998**	0.9380	0.9545	0.9773	6.1880
w/ Entropy-based Token Filter	0.9960	0.9970	0.9994	**0.9560**	**0.9606**	**0.9870**	**5.8942**

Table 6: Performance comparison of our original method and our method with the entropy-based token filter module.

Proxy LM	No Attack			Paraphrasing Attack			Text Quality
	TPR@5%FPR (↑)	Best F1 (↑)	AUROC (↑)	TPR@5%FPR (↑)	Best F1 (↑)	AUROC (↑)	Perplexity (↓)
OPT-125M	0.9960	0.9970	**0.9994**	0.9560	0.9606	0.9870	5.8942
GPT2	0.9980	0.9920	0.9993	0.9573	0.9629	0.9884	5.7403
OPT-350M	**0.9980**	**0.9940**	0.9979	**0.9590**	**0.9682**	**0.9938**	**5.7365**

Table 7: Performance comparison of our method with the entropy-based token filter module with different Proxy LMs.

biasing of the probability distribution severely impacts the semantics which affects the coherence of the text, causing a decline in text quality.

Hence, we set a threshold of entropy to select these low-entropy tokens and preserve their probability distributions, which means there is no watermark injection on these low-entropy tokens. We refer to the token sampled from the original probability distribution as *unwatermarked token* and the token sampled from the biased distribution as *watermarked token*. When detecting text, we filter out these unwatermarked tokens and only calculate the proportion of green tokens among the watermarked tokens.

We follow the entropy described in Sec. 3.4. We regard the LLM using the watermark method as a watermarked LLM. We use a Proxy Model (PM) to calculate the probability distribution of the next token and estimate the entropy given previous sequences to avoid invocating the watermarked LLM during watermark detection, which will turn our method into a white-box method, lacking practicality. We employ OPT-125M (Zhang et al., 2022) as the PM, which is a smaller-scale language model compared to the watermarked LLM.

Filtering the low-entropy tokens means the watermark injection will maintain the semantics and keep the coherence of the text during generation. When detecting text, the proportion of green tokens can be preserved by filtering out these unwatermarked tokens, which maintains robustness against paraphrasing attacks.

We test the performance of this module by adding the module to our method. The result can be found in Table 6. We found our method will have a better performance with this module. We also test the performance of our method with this module while using different Proxy LMs, including OPT-125M (Zhang et al., 2022), GPT-2 (Radford et al., 2019) and OPT-350M (Zhang et al., 2022). The result in Table 7 shows that our method will exhibit even better results as the proxy LM becomes stronger. However, the outperformance comes at the cost of time.

Algorithm 1 Watermark Injection

```
Input: Large language model $LLM(\cdot)$, previous watermarked sequences $s^{(1):(t-1)}$, hash size $K$, green list ratio $\gamma$, watermark strength $\delta$, generation length $L$.
1:  for $t \leftarrow t$ to $L$ do
2:      Initialize $G$
3:      $key \leftarrow LSH(s^{(t-1)})$
4:      for $i \leftarrow 1$ to $K$ do
5:          For each set $S_i$, partition it into a green list $G_i$ of size $|S_i|$ and a red list $R_i$ of size $(1-\gamma)|S_i|$ seeded by $key$.
6:          $G \leftarrow G \cup G_i$
7:      end for
8:      logits $l \leftarrow LLM(s^{(1):(t-1)})$
9:      entropy $ent \leftarrow H(s^{(t)})$
10:     dynamic bias $\delta' = \delta \cdot \frac{1}{ent+\phi}$
11:     for $v \in G$ do
12:         $\hat{l}_v = l_v \cdot (1+\delta')$
13:     end for
14:     biased probs $\hat{p} = softmax(\hat{l})$
15:     sample a next token $s^{(t)}$ from $\hat{p}$
16: end for
Output: watermarked text $s^{(1)}, s^{(2)}, \ldots, s^{(L)}$
```

B Procedure of Watermark Injection and Detection

The process of watermark injection is as Sec. 3.5 described. The injection procedure can be found in Algorithm 1.

Algorithm 2 Watermark Detection

```
Input: Suspicious text $s^{(1)}, s^{(2)}, ..., s^{(L)}$, Proxy LM $PM(\cdot)$, hash size $K$
 1: Initialize count of detected tokens $N_D$
 2: Initialize count of green tokens $N_G$
 3: for $t \leftarrow 2$ to $L$ do
 4:     Initialize $G$
 5:     $key \leftarrow LSH(s^{(t-1)})$
 6:     for $i \leftarrow 1$ to $K$ do
 7:         For each set $S_i$, partition it into a green list $G_i$ of size $|S_i|$ and a red list $R_i$ of size $(1-\gamma)|S_i|$ seeded by $key$.
 8:         $G \leftarrow G \cup G_i$
 9:     end for
10:     if $s^{(t)} \in G$ then
11:         $N_G$ += 1
12:     end if
13:     $N_D$ += 1
14: end for
15: z-score $z = \frac{N_G - \gamma N_D}{\gamma(1-\gamma)N_D}$
Output: z-score $z$
```

During watermark detection, Kirchenbauer et al. (2023a) test the following null hypothesis through a one-proportion z-test to detect whether the text is injected into a watermark:

H_0 : *The text is generated (or written) lacking knowledge about the green/red lists.*

According to Eq. 3, the z-score indicates the difference in the number of green tokens between the suspicious text and the unwatermarked text. The null hypothesis will be rejected if the z-score used in 3 computed based on the number of green tokens in the text exceeds a threshold M.

During detection, we detect the tokens in the text one by one to count the number of green tokens. We determine the text is watermarked when z-score $z > M_r$, where M_r is located according to a given FPR r: We define watermarked as the positive class and unwatermarked as the negative class. We get $M_r = m$ where m is the selected threshold in which r percentage of unwatermarked texts are classified as watermarked falsely. The process of watermark detection is as Alogrithm 2.

C Additional Experiment Settings

Implication details. Following Kirchenbauer et al. (2023a), we utilize OPT-1.3B (Zhang et al., 2022) as the backbone model. For each sample, We use the first 20 tokens of each text as a prompt for the model. For each prompt, we expect the model to generate 200 ± 5 tokens. For KGW, Unigram watermark, SWEET, EWD, and our method, we set the green list percentage γ and the bias δ to 0.5 and 1.5 respectively. For KGW-Large, we use a larger bias $\delta = 2.0$ to test the impact of the large bias on the text quality and robustness. For EXP-Edit, we follow the same settings from the original paper. We use Pytorch (Paszke et al., 2019) during experiments. RealNews dataset uses news from Common Crawl dumps from December 2016 through March 2019 as training data and the articles published in April 2019 from the April 2019 dump as evaluation data.

Computing Infrastructure and Budget: We run sampling and paraphrase attack jobs on 2 A100 GPUs, taking up a total of around 100 GPU hours.

D Generation time of Watermarked Text

This section compares the watermarked text generation time and the watermark detection time for different watermark methods. We follow the same setting for all watermark methods in Sec. 4.1. We generate and detect 500 samples of watermarked text, each containing 200 ± 5 tokens for each method. Later, we compute the average time taken for both the generation and detection of each sample of watermarked text.

The result can be found in Table 8. For generation time, EXP-Edit has the fastest generation time because it directly biases the process of token sampling to inject the watermark and does not require the computation of biasing the probability distribution. However, during the detection time, the EXP-Edit method has the worst performance due to its requirement during watermark detection of calculating the alignment between the watermarked text and the watermark key sequence. The SWEET method and EWD method need to use the original LLM during watermark detection, which causes a time consumption. The performance of our method is very similar to these methods only using green/red lists though we add these modules. Our method with the entropy-based token filter module causes more wastage of computing and time resources compared to our basic method since we introduce a Proxy LM during generation and detection.

Method	Average Generation Time	Average Detection Time
KGW	4.53s	0.05s
Unigram	4.16s	0.04s
SWEET	4.95s	0.10s
EWD	4.56s	0.08s
EXP-Edit	1.53s	172.89s
Ours	4.37s	0.04s
Ours w/ Entropy-based Token Filter	5.62s	0.06s

Table 8: Text generation and detection time performance in different watermark methods.

d	No Attack			Paraphrasing Attack			Text Quality
	TPR@5%FPR (↑)	**Best F1 (↑)**	**AUROC (↑)**	**TPR@5%FPR (↑)**	**Best F1 (↑)**	**AUROC (↑)**	**Perplexity (↓)**
2	0.9900	0.9909	0.9964	**0.9400**	0.9467	0.9689	**5.7186**
4	0.9960	**0.9989**	0.9974	0.9360	**0.9507**	**0.9775**	6.1129
8	**0.9980**	0.9960	**0.9998**	0.9320	0.9494	0.9691	6.0039

Table 9: Performance comparison of robustness and text quality in different settings of d.

E Experiments on different watermark methods based on Llama-2

To test the effect of those watermark methods in a newer and larger LLM, we conducted experiments with a larger model, Llama2-7b (Touvron et al., 2023). We then use Llama-2-13b to calculate the text perplexity. We found the result in Table 3 is similar to the result in Table 1, where our method achieves the best performance of robustness against paraphrasing attacks and the best text quality. The result proves that our method still has a better performance compared to the baselines even if we use a larger model as the backbone model.

F Effect of Number of LSH Hyperplanes

We test the performance of our method across different numbers of the random hyperplanes d. The result can be found in Table 9. We found when d = 4, our method performs best in robustness against paraphrasing attacks since the tokens are more likely to fall into the same region after suffering paraphrasing attacks. However, robustness performance when $d = 2$ is weaker, which indicates too few hyperplanes used in our method will result in more tokens falling into the same region in semantic space, which causes an inflated number of green tokens, leading to misclassifying the unwatermarked text as watermarked text, causing bad performance in both the detectability of the watermarked text and robustness against paraphrasing attacks. Based on the above analysis, we set d to 4 in Sec. 4 to maintain robustness against paraphrasing attacks.