CEFW: A Comprehensive Evaluation Framework for Watermark in Large Language Models

Shuhao Zhang[1], Bo Cheng[1†], Jiale Han[2], Yuli Chen[1], Zhixuan Wu[1], Changbao Li[3], Pingli Gu[3]
[1]*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications*
[2]*Hong Kong University of Science and Technology*
[3]*BigData R&D Center, North China Institute of Computing Technology*
{2020111429,chengbo,chenyuli,wzxmogu}@bupt.edu.cn, jialehan@ust.hk, {lichangbao_1, gpl98}@163.com

***Abstract*—Text watermarking provides an effective solution for identifying synthetic text generated by large language models. However, existing techniques often focus on satisfying specific criteria while ignoring other key aspects, lacking a unified evaluation. To fill this gap, we propose the Comprehensive Evaluation Framework for Watermark (CEFW), a unified framework that comprehensively evaluates watermarking methods across five key dimensions: ease of detection, fidelity of text quality, minimal embedding cost, robustness to adversarial attacks, and imperceptibility to prevent imitation or forgery. By assessing watermarks according to all these key criteria, CEFW offers a thorough evaluation of their practicality and effectiveness. Moreover, we introduce a simple and effective watermarking method called Balanced Watermark (BW), which guarantees robustness and imperceptibility through balancing the way watermark information is added. Extensive experiments show that BW outperforms existing methods in overall performance across all evaluation dimensions. We release our code to the community for future research[1].**

***Index Terms*—Watermark for LLMs, Watermark Evaluation Framework, Large Language Model**

I. Introduction

With the rapid development of large language models (LLMs) [1], [2], text generated by LLMs increasingly resembles human-generated text and gradually fills all parts of our lives. This trend presents several potential threats, including hallucinations [3], misinformation generation [4], and malicious use [1], [5]. Therefore, detecting text generated by LLMs has become an emerging and critical issue.

Digital watermark [6], [7] is a promising approach for detecting LLM-generated text. The approach embeds watermark information into the text and determines whether the text is generated by the LLM by detecting the watermark information. As shown in Fig. 1, an effective watermark should meet both applicability and security requirements, demonstrating the following five characteristics: (1) **Detectability**: The ability to accurately distinguish between watermarked and non-watermarked text; (2) **Text Quality**: The quality of watermarked texts should not significantly be compromised; (3) **Usability**: The time and resource consumption incurred by the addition and detection of a watermark should be acceptable; (4) **Robustness**: The watermark should remain detectable even when the watermarked text is subjected to scrubbing attacks; (5) **Imperceptibility**: The watermark should not be cracked by spoofing attacks, ensuring that attackers cannot generate legitimate watermark text.

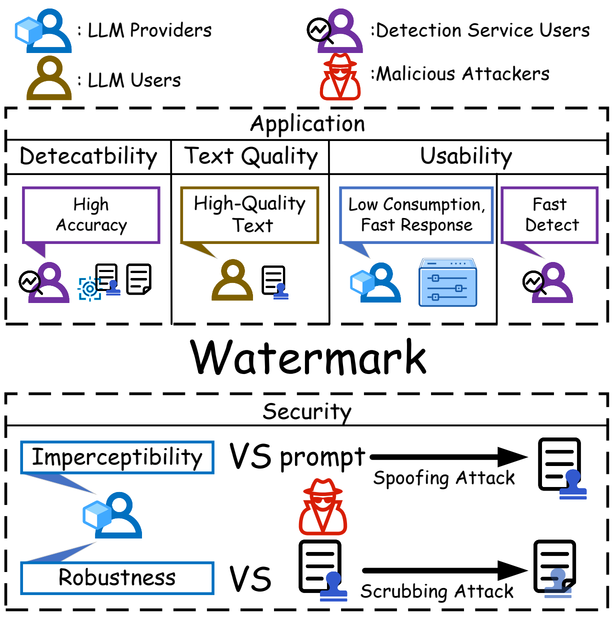


Fig. 1. Watermarks have three distinct audiences: LLM providers, LLM users, and detection service users. They necessitate watermarks to have detectability, text quality, and usability in terms of application. Concurrently, LLM providers must ensure the imperceptibility and robustness of watermarks in terms of security to counteract scrubbing attacks and spoofing attacks by malicious attackers.

Previous watermark studies [7]–[13] focus only on part of the necessary watermark characteristics without considering the whole picture for their watermarks. KGW [7] emphasizes detectability and robustness, but neglects usability and imperceptibility when evaluation. Unigram Watermark (UNIW) [9] aims to enhance robustness, however, the imperceptibility neglected by UNIW is an obviously shortcoming. Although these watermarks are proven to be effective for the characteristics they focus on, the lack of evaluation of the other characteristics makes them unreliable, thereby hindering their practicality. The "barrel effect" illustrates that the capacity of a barrel is determined by its shortest plank, highlighting the idea that

† Corresponding Author.
[1]https://github.com/DrankXs/BalancedWatermark

the weakest component often limits the overall performance or potential of a system. Therefore, a uniform and complete watermark evaluation framework is highly important.

In this paper, we propose the Comprehensive Evaluation Framework for Watermark in LLMs (CEFW) to obtain a comprehensive evaluation for the watermark. CEFW selects watermark metrics for each characteristic and calculates them to obtain the corresponding score. By weighting five characteristic scores, CEFW offers a comprehensive score for the evaluated watermark. We also notice that the way for watermark information adding is over-dynamic in KGW, and over-static in UNIW. Therefore, we propose Balanced Watermark (BW), it is designed to balance the way watermark information is added. BW integrates the advantages of both dynamic and static aspects to obtain a higher comprehensive score. Our experiment proves that BW not only has the best comprehensive performance, but also has no obvious shortcomings.

Our main contributions are as follows:

- We propose the Comprehensive Evaluation Framework (CEFW) for Watermark in LLM from five key dimensions, which first standardizes the watermark evaluation process and provides a comprehensive score for watermark.
- We introduce the Balanced Watermark (BW), which balances the way of adding watermark information to avoid being over-dynamic or over-static, offering a well-rounded and practical solution.
- We carry out extensive experiments to obtain the comprehensive score of different watermarks using the CEFW framework, and experimental results demonstrate that BW achieves the highest comprehensive performance.

II. Related Work

A. LLM-Generated Text Watermark

In order to distinguish between texts generated by models and those composed in natural language, some scholars try to find a more accurate detector [14], [15], while others decided to tackle the problem at the source by adding watermarks to the LLM-generated text. Watermarks for LLM-generated text can be categorized into three types based on different embedding phrases: A-prior watermarks modify parameters of LLM [16], [17]; generating watermarks change the output probabilities of LLM [7]–[9], [13], [18]; plain text watermarks edit existing texts [19], [20]. Generating watermarks emerge as a focal point of current watermark research [12], [21], [22]. It does not require fine-tuning LLM, and effectively preserves the original inference capabilities of the LLM.

B. Watermark Evaluation

Most researchers, when proposing a new watermark, only evaluate detectability, text quality, and robustness [7]–[9]. The measurement of detectability typically employs binary classification metrics, while text quality is represented by perplexity and rouge score. For robustness, various researchers employ different scrubbing attacks to remove the watermark from the text and analyze changes in detectability to assess robustness. The most effective approach is DIPPER [23]. Liu et al. [21] take into account the usability and imperceptibility required by the watermark. They evaluate usability by comparing changes in generation speed, but overlook the additional memory consumption caused by the watermark. Regarding imperceptibility (referred to as security robustness), Liu et al. [21] apply the spoofing attack proposed by [24] and test the success rate of the attack to evaluate imperceptibility.

III. Comprehensive Evaluation Framework for Watermark

In this section, we first analyze why the five characteristics are necessary for watermark evaluation. Subsequently, we present the overall process of the Comprehensive Evaluation Framework for Watermark in LLMs (CEFW).

A. Role Analysis

The five characteristics are necessary for watermark evaluation due to the requirements of actual LLM service.

Actual LLM service with the watermark primarily involves four roles: LLM provider, LLM user, detection service user, and malicious attacker.

At the application level, LLM providers should consider the impact of watermarks on memory consumption and response speed during deployment, which refers to **usability**. When using LLM, LLM users expect more accurate responses, signifying higher **text quality**. Detection service users desire more accurate and swift detection of watermarked text, necessitating better **detectability** and **usability** of the watermark.

From a security perspective, malicious attackers attempt two types of attack method: scrubbing attack and spoofing attack. Scrubbing attacks modify the watermarked text with the aim of removing the watermark while preserving the meaning of the text generated by LLM. LLM providers require watermarks with higher **robustness** to defend against scrubbing attacks. Spoofing attacks involve unauthorized cracking of the watermark and simulating watermarked text, which can undermine the credibility of target watermark. To resist spoofing attacks, watermarks are required to demonstrate greater **imperceptibility**.

B. Overall Process

We first define a few symbols: Y is the text generated by LLM without watermark; $\hat{Y}$ is the watermarked text; X is the prompt for generating Y and $\hat{Y}$; $A(\cdot)$ is the text generated by the attack algorithm; $M(\cdot)$ is the metric function to calculate the metric value; S is the score obtained after normalization.

a) Detectability: We select Area Under the Receiver Operating Characteristic Curve (AUCROC) for detectability evaluation. AUCROC integrates the information of True Positive Rate (TPR) and False Positive Rate (FPR) to comprehensively evaluate the detectability of watermark. The worst AUCROC value is 0.5, which indicates a random classification of texts, while the best is 1. CEFW normalizes it at the following:

$$S_D = \frac{M_{ACUROC}(Y, \hat{Y}) - 0.5}{1 - 0.5} \tag{1}$$

S_D is the detectability score.

b) Text Quality: We utilize perplexity (PPL) as the metric for text quality, which is the most popular in watermark evaluation. The advantage of PPL is versatile. PPL does not lose its evaluation function as ROUGE does due to changes in the generation task. CEFW obtains text quality score S_T by the following:

$$S_T = \frac{M_{PPL}(\hat{Y}) - 2M_{PPL}(Y)}{M_{PPL}(Y) - 2M_{PPL}(Y)} \tag{2}$$

We believe that evaluating the impact of watermark on text quality depends on the degree of degradation it brings to LLM. Eq. (2) represents the best watermark will not degrade the LLM, while the worst one degrades it by up to twofold, indicating twice $M_{PPL}(Y)$.

c) Usability: Usability refers to the impact of the watermark on the normal use of LLM services and the efficiency of watermark detection. In terms of time, we evaluate the Generate Time and Detect Time. As for memory aspect, we evaluate the Memory Cost after loading LLM and watermark. Like PPL, Memory Cost and Generate Time need to compare the change of metrics before and after adding the watermark to evaluate the impact of the watermark, they can utilize the same normalization method as Eq. (2). Scores S_{MC} and S_{GT} are obtained by normalizing the metric values of Memory Cost and Generate Time, respectively. CEFW regards Detect Time as the time to detect a piece of text and sets the upper bound to 0 second and the lower bound to 1 second. We define the Detect Time score S_{DT} as:

$$S_{DT} = \frac{M_{DetectTime}(\hat{Y}) - 1}{0 - 1} \tag{3}$$

We average S_{MC}, S_{GT} and S_{DT} to obtain the usability score S_U.

d) Robustness: Robustness is the ability of a watermark to remain detectable following a scrubbing attack. When the watermark has the best robustness, it will maintain the original detectability after a scrubbing attack. We compare the detectability metric value of watermarked text before and after scrubbing attack. CEFW selects AUCROC as the detectability metric and DIPPER as the scrubbing attack method. DIPPER is a language model to avoid watermark detection by paraphrasing text. The final robustness score S_R is formalized as:

$$S_R = \frac{M_{AUCROC}(Y, A_{DIPPER}(\hat{Y})) - 0.5}{M_{AUCROC}(Y, \hat{Y}) - 0.5} \tag{4}$$

e) Imperceptibility: Imperceptibility is evaluated by detecting the existence of a watermark in the text generated by spoofing attacks. When spoofing attacks are most effective, they generate watermarked text that can be perfectly detected. We utilize AUCROC as the detectability metric and STEAL as the spoofing attack. STEAL score S_{STEAL} is defined as:

$$S_{STEAL} = \frac{M_{AUCROC}(Y, A_{STEAL}(X)) - 1}{0.5 - 1} \tag{5}$$

STEAL statistically analyzes the n-gram frequency differences between non-watermarked text and watermarked texts to crack watermark. To achieve a more effective attack, CEFW performs four STEAL attacks by varying the value of n from 1 to 4 in the n-gram frequency analysis, as the watermark is unknown to the attackers. After implementing the spoofing attack procedure, CEFW can obtain four STEAL scores. To obtain more realistic imperceptibility scores through four STEAL scores, we conceive two scenarios for spoofing attack: 1) Non-Authorization (N.A.): Malicious attacker lacks authorization to access the watermark detection service provided by LLM providers. 2) Authorization (A.): Malicious attacker has unrestricted access to detect watermarked texts.

Under N.A., malicious attacker lacks the capability to discern which spoofing attack is the most successful, thus can only make a random selection. Under A., malicious attacker can easily select the most effective type of spoofing text generation models.

CEFW selects scenario A., where the imperceptibility score S_I is determined as the minimum among the four S_{STEAL}.

f) Comprehensive: CEFW calculates a comprehensive score by weighting all five characteristic scores. The weights assigned are $\frac{1}{6}$ for detectability, $\frac{1}{6}$ for text quality, $\frac{1}{6}$ for usability, $\frac{1}{4}$ for robustness, $\frac{1}{4}$ for imperceptibility, Our weight assignment criterion is that applicability and security are equally important, and each characteristic under them has equal importance.

C. Flexible Design

CEFW allows for the customization of metrics and their corresponding weights for each characteristic. The available watermark metrics are detailed in the Appendix C.

IV. Balanced Watermark

A. Original Watermark

Give a prompt $X = \{x_1, x_2, ..., x_{|X|}\}$, an LLM with parameters θ can generate a response $Y = \{y_1, y_2, ..., y_{|Y|}\}$. We can formula the probability distribution of the i-th token y_i as:

$$P(y_i) = P_\theta(y_i | X, Y_{<i}) \tag{6}$$

To obtain $P(y_i)$, the LLM generates a logit value $l_k^{(i)}$ for each token k in the vocabulary $\mathcal{V}$ at generation time.

Both KGW and UNIW add watermark information by modifying $l_k^{(i)}$ at each generation step. They utilize corresponding **partition functions** to partition $\mathcal{V}$ into a green list $\mathcal{G}$ and a red list $\mathcal{R}$, and the logits of green tokens are increased by a positive constant δ. The watermark probability $p_k^{(i)}$ can be formalized as follows:

$$p_k^{(i)} = \begin{cases} \frac{exp(l_k^{(i)}+\delta)}{\sum_{j\in\mathcal{R}} exp(l_j^{(i)})+\sum_{j\in\mathcal{G}} exp(l_j^{(i)}+\delta)}, & k \in \mathcal{G} \\ \frac{exp(l_k^{(i)})}{\sum_{j\in\mathcal{R}} exp(l_i^{(t)})+\sum_{j\in\mathcal{G}} exp(l_j^{(i)}+\delta)}, & k \in \mathcal{R} \end{cases} \tag{7}$$

This results in an increased number of green tokens in watermarked text. We can calculate the number of green tokens in the given sentence, and utilize the z-statistic as the criterion to determine whether the watermark exists.

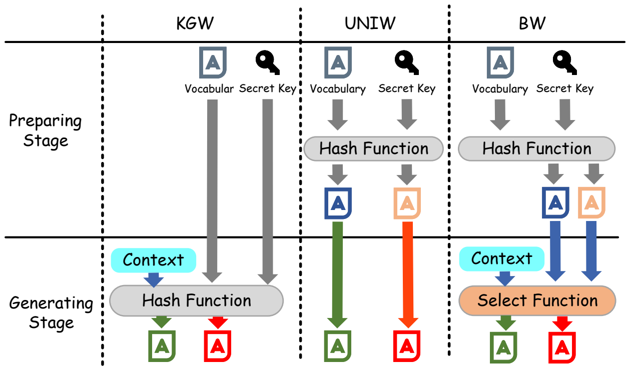


Fig. 2. Difference of partition functions in KGW, UNIW, and BW. Preparing Stage is before entering the prompt to LLM. Generating Stage is at when LLM generates a response.

B. Partition Function

We show the difference in the partition function between KGW, UNIW, and BW in Fig. 2.

KGW partition vocabulary at generating stage, the introduction of context makes $\mathcal{G}$ and $\mathcal{R}$ change at each generation step. UNIW partition vocabulary at the preparing stage, $\mathcal{G}$ and $\mathcal{R}$ are constant during generation. In the UNIW study, Zhao et al. [9] prove that fixed $\mathcal{G}$ and $\mathcal{R}$ will bring higher robustness. However, fixed $\mathcal{G}$ also introduces some tokens with an exceptionally high frequency, which in turn makes spoofing attacks more feasible to execute.

Addressing the limitation of UNIW, BW designs a Select Function to dynamically confirm $\mathcal{G}$ and $\mathcal{R}$. At preparing stage, BW, like UNIW, utilizes fixed list partition to obtain two subsets $\mathcal{A}$ and $\mathcal{B}$ by dividing the vocabulary to. At the generating stage, BW inputs context into Select Function, which then determines $\mathcal{G}$ by choosing between $\mathcal{A}$ and $\mathcal{B}$. $\mathcal{G}$ is dynamic for BW, just as for KGW.

When generating the token y_i, the context used for the watermark is $\{y_{i-w}, y_{i-w+1}, ..., y_{i-1}\}$, w denotes the length of the context window, which can be seen as watermark complexity. The Select Function F_S maps the context to a signal and will determine which of $\mathcal{A}$ or $\mathcal{B}$ is determined as $\mathcal{G}$. The process of selecting $\mathcal{G}$ can be formalized as follows:

$$\mathcal{G} = \begin{cases} \mathcal{A}, & F_S(\{y_{i-w}, y_{i-w+1}, ..., y_{i-1}\}) = 1 \\ \mathcal{B}, & F_S(\{y_{i-w}, y_{i-w+1}, ..., y_{i-1}\}) = 0 \end{cases} \tag{8}$$

To avoid the unusually high frequency in tokens, the probability that $\mathcal{A}$ and $\mathcal{B}$ are selected as $\mathcal{G}$ should be approximately equal. BW utilizes the first token y_{i-w} of context to determine $\mathcal{G}$ like KGW. Therefore, we introduce token frequency statistics to make the mapping of F_S as balanced as possible. BW counts the frequency of each token t in $\mathcal{V}$ across a large set of non-watermarked texts generated by the LLM. Based on the token frequency, we form an ordered list $\{t_1, t_2, ..., t_{|V|}\}$. Following this ordered list, when t_j is y_{i-w}, the first token of context , F_S can be formalized as:

$$F_S(t_j) = \begin{cases} 1, & j\%2 = 0 \\ 0, & j\%2 \neq 0 \end{cases} \tag{9}$$

We present the overall process for BW in Appendix A1.

V. Experiments

A. Experiment Setup

a) Dataset: We employ C4 [25][2] for our text generation task, and Quora-QA [26][3] to simulate the LLM service scenario. For input, we extract the first 30 tokens from each text in C4 and utilize the question part in Quora-QA.

b) Language Model: We select two language models. One is OPT-2.7b in the OPT family [27], which is widely utilized for watermark evaluation. Another is Llama3-8b [2], which is a latest large language model. During each generation, we employ sampling as the decoding strategy and generate a maximum of 200 tokens.

c) Evaluated Watermarks: We compare KGW [7] and UNIW [9] with BW. For each watermark, we equally partition vocabulary and set δ as 2. We define KGWw for KGW with watermark complexity w and set watermark complexity 1 to 4 for both KGW and BW.

B. Comprehensive Analysis

By comparing different watermarks, BW4 is the most recommended watermark. Table I presents five characteristic scores and the comprehensive score for all watermarks.

The imperceptibility score of UNIW is always 0, which means STEAL can completely crack it. KGW with high watermark complexity does not exhibit any significant drawbacks. However, in the majority of instances, BW consistently demonstrates superior text quality and robustness compared to KGW when both employ the same watermark complexity. In the remaining cases, the scores for the corresponding characteristics are approximately equal. In the comparison of comprehensive scores, BW4 achieves the highest comprehensive score in three settings, and BW1 achieves in one. However, BW4 has no obvious shortcomings with all characteristic scores higher than 0.4, while BW1 has significant imperceptibility disadvantages.

In our design, BW achieves a high comprehensive score for two reasons: 1) **Select Function brings higher imperceptibility than UNIW**; 2) **Fixed list partition brings higher robustness than KGW**.

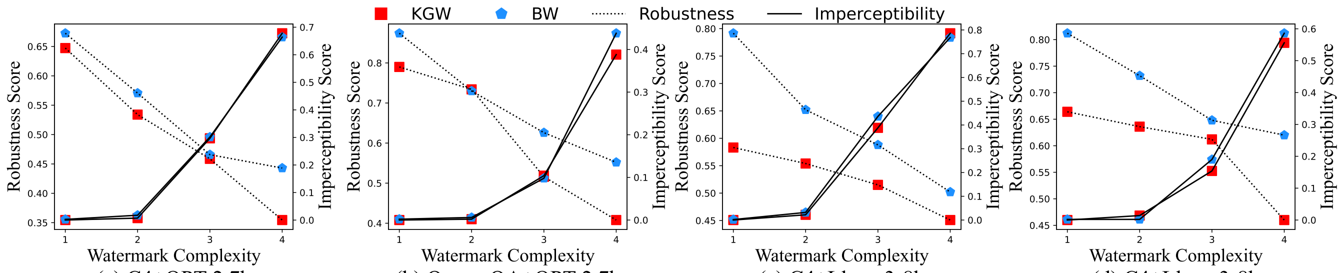
Select Function enables BW to regulate its watermark complexity. As shown in Table I, watermark complexity significantly affects the imperceptibility. Watermarks with high watermark complexity (KGW3, KGW4, BW3, BW4) have imperceptibility scores higher than UNIW obviously. By increasing watermark complexity, BW can achieve a higher imperceptibility score compared to the UNIW.

According to the result of Zhao et al. [9], fixed list partition and selection will bring the watermarks higher robustness. Fixed list partition makes the watermark information of BW more stubborn. Thus, BW exhibits robustness that is superior to or equivalent to that of KGW.

[2]https://huggingface.co/datasets/allenai/c4
[3]https://huggingface.co/datasets/toughdata/quora-question-answer-dataset

TABLE I
THE RESULT OF DIFFERENT WATERMARKS UNDER CEFW. S_D, S_T, S_U, S_I, AND S_R ARE RESPECTIVELY THE SCORES OF DETECTABILITY, TEXT QUALITY, USABILITY, IMPERCEPTIBILITY, AND ROBUSTNESS AFTER NORMALIZATION UNDER CEFW. S_{CEFW} REPRESENTS THE COMPREHENSIVE SCORE OF WATERMARK UNDER CEFW. DARKER COLORS INDICATE BETTER RELATIVE RANKINGS.

			UNIW	KGW1	KGW2	KGW3	KGW4	BW1	BW2	BW3	BW4
OPT-2.7b	C4	S_D	0.998	1.000	1.000	0.999	0.998	0.999	0.998	1.000	0.999
		S_T	0.571	0.454	0.403	0.322	0.287	0.527	0.510	0.466	0.443
		S_U	0.953	0.937	0.944	0.925	0.925	0.948	0.944	0.953	0.948
		S_I	0.000	0.000	0.007	0.297	0.678	0.003	0.018	0.301	0.664
		S_R	0.986	0.647	0.534	0.459	0.355	0.673	0.571	0.466	0.443
		S_{CEFW}	0.667	0.560	0.526	0.563	0.626	0.581	0.556	0.595	**0.675**
	Quora-QA	S_D	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
		S_T	0.345	0.385	0.388	0.457	0.453	0.671	0.528	0.479	0.477
		S_U	0.992	0.980	0.980	0.982	0.980	0.995	0.989	0.991	0.995
		S_I	0.000	0.000	0.002	0.104	0.388	0.002	0.006	0.098	0.438
		S_R	0.930	0.790	0.734	0.516	0.408	0.874	0.730	0.626	0.552
		S_{CEFW}	0.622	0.592	0.579	0.561	0.604	**0.663**	0.603	0.593	0.659
Llama3-8b	C4	S_D	0.997	0.997	0.997	0.995	0.999	0.998	0.999	0.999	0.998
		S_T	0.694	0.602	0.564	0.512	0.525	0.582	0.576	0.524	0.537
		S_U	0.998	0.980	0.979	0.982	0.981	0.992	0.992	0.993	0.992
		S_I	0.002	0.000	0.022	0.388	0.786	0.002	0.032	0.436	0.768
		S_R	0.772	0.583	0.554	0.515	0.451	0.792	0.652	0.588	0.502
		S_{CEFW}	0.641	0.576	0.567	0.641	0.727	0.627	0.599	0.675	**0.739**
	Quora-QA	S_D	0.998	1.000	0.999	1.000	1.000	0.999	1.000	1.000	1.000
		S_T	0.781	0.651	0.589	0.534	0.453	0.660	0.625	0.578	0.545
		S_U	0.996	0.982	0.983	0.983	0.983	0.981	0.981	0.980	0.952
		S_I	0.000	0.000	0.014	0.154	0.556	0.002	0.002	0.190	0.586
		S_R	0.784	0.664	0.636	0.612	0.460	0.812	0.732	0.648	0.620
		S_{CEFW}	0.659	0.605	0.591	0.611	0.660	0.643	0.618	0.636	**0.718**



(a) C4+OPT-2.7b (b) Quora-QA+OPT-2.7b (c) C4+Llama3-8b (d) C4+Llama3-8b

Fig. 3. Watermark Complexity Analysis.

For other characteristics, the three watermarks achieve scores greater than 0.9 in detectability and usability. In text quality, BW shows a more stable advantage over KGW, with scores that are either higher or equivalent. In contrast, the text quality of UNIW may suffer a significant decline due to an unreasonable list partition.

C. Watermark Complexity Analysis

Randomness brings watermark higher imperceptibility but lower robustness.

Tokens that are closer in natural language are generally considered to have stronger correlations. Higher watermark complexity makes the token selected in the context and the token to be generated have a lower correlation, resulting in a more random green list.

As shown in Fig. 3, the additional randomness introduced by the increase in watermark complexity improves the imperceptibility of BW and KGW at the cost of reduced robustness.

D. STEAL Analysis

Select Function brings close imperceptibility as KGW for BW. To further validate it, we present the AUCROC values of texts generated by 4 STEAL attacks in Table II.

We observe that a fixed configuration STEAL, when attacking BW and KGW with the same complexity, generates spoofing texts with AUCROC scores that mostly differ by no more than 0.05, exhibiting closely similar AUCROC scores. As a result, no matter how the watermark setting or attack setting changes, the imperceptibility brought about by Select Function for BW is always similar to that of KGW.

VI. Conclusion

In this paper, we propose a Comprehensive Evaluation Framework for Watermark in LLMs (CEFW). CEFW standardizes the rational evaluation process for watermarks by considering five necessary characteristics. We also propose a novel watermark, termed Balanced Watermark (BW), which balances the method to add watermark information. In the ex-

TABLE II
THE AUCROC VALUE OF ALL WATERMARKS FOR IMPERCEPTIBILITY EVALUATION. STEALn REPRESENTS THE n-GRAM FREQUENCY ANALYSIS FOR THIS STEAL ATTACK. A. REPRESENTS AUTHORIZATION, AND N.A. REPRESENTS NON-AUTHORIZATION.

			UNIW	KGW1	KGW2	KGW3	KGW4	BW1	BW2	BW3	BW4
OPT-2.7b	C4	STEAL1	1.000	1.000	0.874	0.594	0.551	0.998	0.891	0.611	0.546
		STEAL2	0.989	0.991	0.996	0.788	0.639	0.990	0.991	0.797	0.657
		STEAL3	0.762	0.838	0.826	0.852	0.661	0.814	0.837	0.849	0.668
		STEAL4	0.574	0.629	0.597	0.637	0.655	0.614	0.608	0.647	0.644
		N.A.	0.832	0.865	0.823	0.718	0.626	0.854	0.832	0.726	0.629
		A.	1.000	1.000	0.996	0.852	0.661	0.998	0.991	0.849	0.668
	Quora-QA	STEAL1	1.000	1.000	0.832	0.661	0.543	0.999	0.829	0.669	0.571
		STEAL2	0.995	0.999	0.999	0.828	0.637	0.997	0.997	0.787	0.655
		STEAL3	0.903	0.829	0.930	0.948	0.730	0.924	0.936	0.951	0.683
		STEAL4	0.700	0.733	0.735	0.779	0.806	0.726	0.756	0.748	0.781
		N.A.	0.900	0.890	0.874	0.804	0.679	0.912	0.880	0.789	0.673
		A.	1.000	1.000	0.999	0.948	0.806	0.999	0.997	0.951	0.781
Llama3-8b	C4	STEAL1	0.999	1.000	0.768	0.603	0.563	0.999	0.676	0.581	0.533
		STEAL2	0.968	0.992	0.989	0.733	0.604	0.985	0.984	0.712	0.616
		STEAL3	0.674	0.786	0.774	0.806	0.607	0.726	0.756	0.782	0.583
		STEAL4	0.550	0.568	0.563	0.553	0.576	0.588	0.569	0.575	0.583
		N.A.	0.798	0.837	0.774	0.674	0.588	0.825	0.746	0.663	0.579
		A.	0.999	1.000	0.989	0.806	0.607	0.999	0.984	0.782	0.616
	Quora-QA	STEAL1	1.000	1.000	0.791	0.584	0.564	0.999	0.750	0.626	0.536
		STEAL2	0.987	0.998	0.993	0.828	0.645	0.996	0.999	0.791	0.639
		STEAL3	0.821	0.910	0.914	0.923	0.695	0.875	0.901	0.905	0.663
		STEAL4	0.623	0.678	0.659	0.676	0.722	0.656	0.672	0.683	0.707
		N.A.	0.858	0.897	0.839	0.753	0.657	0.882	0.831	0.751	0.636
		A.	1.000	1.000	0.993	0.923	0.722	0.999	0.999	0.905	0.707

perimental part, we prove that BW has the best comprehensive performance among the three watermarks.

ACKNOWLEDGMENT

This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFF0902701; in part by the National Natural Science Foundation of China under Grants U21A20468, 62372058, U22A2026.

REFERENCES

[1] OpenAI, "GPT-4 technical report," *CoRR*, vol. abs/2303.08774, 2023.
[2] AI@Meta, "Llama 3 model card," 2024.
[3] Hussam Alkaissi and Samy I McFarlane, "Artificial hallucinations in chatgpt: implications in scientific writing," *Cureus*, vol. 15, no. 2, 2023.
[4] Yizhou Zhang, Karishma Sharma, Lun Du, and Yan Liu, "Toward mitigating misinformation and social media manipulation in LLM era," in *WWW*, 2024, pp. 1302–1305.
[5] N Editorials, "Tools such as chatgpt threaten transparent science; here are our ground rules for their use," *Nature*, vol. 613, no. 7945, pp. 612, 2023.
[6] Mikhail J. Atallah, Victor Raskin, et al., "Natural language watermarking: Design, analysis, and a proof-of-concept implementation," in *Information Hiding, 4th International Workshop, IHW*. 2001, vol. 2137 of *Lecture Notes in Computer Science*, pp. 185–199, Springer.
[7] John Kirchenbauer, Jonas Geiping, et al., "A watermark for large language models," in *ICML*, 2023, pp. 17061–17084.
[8] Yu Fu, Deyi Xiong, and Yue Dong, "Watermarking conditional text generation for AI detection: Unveiling challenges and a semantic-aware watermark remedy," in *AAAI*, 2024, pp. 18003–18011.
[9] Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang, "Provable robust watermarking for AI-generated text," in *ICLR*, 2024.
[10] Xuandong Zhao, Yu-Xiang Wang, and Lei Li, "Protecting language generation models via invisible watermarking," in *ICML*, 2023, pp. 42187–42199.
[11] Abe Bohan Hou, Jingyu Zhang, et al., "Semstamp: A semantic watermark with paraphrastic robustness for text generation," in *NAACL*, 2024, pp. 4067–4082.
[12] Yepeng Liu and Yuheng Bu, "Adaptive text watermark for large language models," in *ICML*, 2024.
[13] Yijian Lu, Aiwei Liu, Dianzhi Yu, Jingjing Li, and Irwin King, "An entropy-based text watermarking detection method," in *ACL*, 2024, pp. 11724–11735.
[14] Sebastian Gehrmann, Hendrik Strobelt, and Alexander M. Rush, "GLTR: statistical detection and visualization of generated text," in *ACL*, 2019, pp. 111–116.
[15] Eric Mitchell, Yoonho Lee, et al., "Detectgpt: Zero-shot machine-generated text detection using probability curvature," in *ICML*, 2023, pp. 24950–24962.
[16] Yossi Adi, Carsten Baum, et al., "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *USENIX Security 18*, 2018, pp. 1615–1631.
[17] Wenjun Peng, Jingwei Yi, et al., "Are you copying my model? protecting the copyright of large language models for eaas via backdoor watermark," in *ACL*, 2023, pp. 7653–7668.
[18] Zhengmian Hu, Lichang Chen, et al., "Unbiased watermark for large language models," in *ICLR*, 2024.
[19] Xi Yang, Jie Zhang, et al., "Tracing text provenance via context-aware lexical substitution," in *AAAI*, 2022, pp. 11613–11621.
[20] Yuhang Li, Yihan Wang, Zhouxing Shi, and Cho-Jui Hsieh, "Improving the generation quality of watermarked large language models via word importance scoring," *CoRR*, vol. abs/2311.09668, 2023.
[21] Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen, "A semantic invariant robust watermark for large language models," in *ICLR*, 2024.
[22] Aiwei Liu, Leyi Pan, et al., "An unforgeable publicly verifiable watermark for large language models," in *ICLR*. 2024, OpenReview.net.
[23] Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer, "Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense," in *NeurIPS*, 2023.
[24] Vinu Sankar Sadasivan, Aounon Kumar, et al., "Can ai-generated text be reliably detected?," *CoRR*, vol. abs/2303.11156, 2023.
[25] Colin Raffel, Noam Shazeer, et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, pp. 140:1–140:67, 2020.
[26] Zhiguo Wang, Wael Hamza, and Radu Florian, "Bilateral multi-perspective matching for natural language sentences," in *IJCAI*, 2017, pp. 4144–4150.
[27] Susan Zhang, Stephen Roller, et al., "OPT: open pre-trained transformer language models," *CoRR*, vol. abs/2205.01068, 2022.

[28] Nikola Jovanovic, Robin Staab, and Martin T. Vechev, "Watermark stealing in large language models," in *ICML*. 2024, OpenReview.net.

[29] Chenchen Gu, Xiang Lisa Li, Percy Liang, and Tatsunori Hashimoto, "On the learnability of watermarks for language models," in *ICLR*. 2024, OpenReview.net.

Appendix

A. Algorithm

1) Balanced Watermark: We summarize Section IV and give the overall flow algorithm of BW in Algorithm 1.

Algorithm 1 Balanced Watermark

```
Require: Prompt $Y = \{y_1, y_2, ...y_{|Y|}\}$, Large Language Model $LLM$, secret key $\mathcal{K}$, logits bias $\delta > 0$, watermark complexity $w$.
Ensure: watermarked text
 1: Count token frequencies from large amounts of text generated by $LLM$;
 2: Sort tokens on $\mathcal{V}$ by token frequencies;
 3: Construct Select Function $F_S$ according to the order tokens list;
 4: Apply $\mathcal{K}$ as a random seed, randomly and uniformly partition the vocabulary $\mathcal{V}$ into lists $\mathcal{A}$ and $\mathcal{B}$;
 5: for $i \leftarrow |Y| + 1$ to ... do
 6:     Based on prompt $y_{<i}$, LLM get a logits distribution $l^{(i)}$ on the vocabulary $\mathcal{V}$;
 7:     if $F_S(y_{i-w}) = 1$ then
 8:         $\mathcal{G} = \mathcal{A}$, $\mathcal{R} = \mathcal{B}$
 9:     else if $F_S(y_{i-w}) = 0$ then
10:         $\mathcal{G} = \mathcal{B}$, $\mathcal{R} = \mathcal{A}$
11:     end if
12:     Add a fixed bias value $\delta$ to all green tokens logits, then obtain a new probability distribution $p_w^{(i)}$ over the vocabulary $\mathcal{V}$ through softmax;
13:     Sample the next token $y_i$ from $p_w^{(i)}$.
14: end for
```

2) KGW: In this paper, we evaluate the most commonly used Soft Watermark of KGW. To be consistent with BW, we set the green list ratio to 0.5. The algorithm is shown in Algorithm 2.

3) UNIW: Like KGW, UNIW also sets the green list ratio at 0.5. We show its process in Algorithm 3.

4) DIPPER: DIPPER [23] is a language model for the scrubbing attack. In robustness evaluation, it rewrites the watermark text to avoid watermark detection while maintaining the meaning of the text. Typically, the watermark has high robustness if the watermark texts have high detectability after DIPPER.

5) STEAL: We provide a simple introduction for the spoofing attack, STEAL [28]. We only introduce one configuration from multiple attack modes of STEAL, which considers only the n-gram scenario, it conducts a spoofing attack by statistically analyzing the n-gram frequency differences between non-watermarked text and watermarked texts.

Algorithm 2 KGW

```
Require: Prompt $Y = \{y_1, y_2, ...y_{|Y|}\}$, Large Language Model $LLM$, secret key $\mathcal{K}$, logits bias $\delta > 0$, watermark complexity $w$.
Ensure: watermarked text.
1: for $i \leftarrow |Y| + 1$ to ... do
2:     Based on prompt $y_{<i}$, LLM get a logits distribution $l^{(i)}$ on the vocabulary $\mathcal{V}$;
3:     Compute a hash of token $y_{i-w}$, and use it to seed a random number generator;
4:     Using this random number generator to randomly and uniformly partition the vocabulary;
5:     Add a fixed bias value $\delta$ to all green tokens logits, then obtain a new probability distribution $p_w^{(i)}$ over the vocabulary $\mathcal{V}$ through softmax;
6:     Sample the next token $y_i$ from $p_w^{(i)}$.
7: end for
```

Algorithm 3 UNIW

```
Require: Prompt $Y = \{y_1, y_2, ...y_{|Y|}\}$, Large Language Model $LLM$, secret key $\mathcal{K}$, logits bias $\delta > 0$, watermark complexity $w$.
Ensure: watermarked text
1: Use secret key $\mathcal{K}$ to seed a random generator;
2: Using this random number generator to randomly and uniformly partition the vocabulary;
3: for $i \leftarrow |Y| + 1$ to ... do
4:     Based on prompt $y_{<i}$, LLM get a logits distribution $l^{(i)}$ on the vocabulary $\mathcal{V}$;
5:     Add a fixed bias value $\delta$ to all green tokens logits, then obtain a new probability distribution $p_w^{(i)}$ over the vocabulary $\mathcal{V}$ through softmax;
6:     Sample the next token $y_i$ from $p_w^{(i)}$.
7: end for
```

To record n-gram frequency statistic, we tokenize both non-watermarked and watermarked texts. After tokenizing the texts, we record the token sequences in a dictionary. For an n-gram spoofing attack, the dictionary counts $(T_{i-n}, T_{i-n+1}, ..., T_{i-1}, T_i)$ as the key, with the corresponding count as the value. T_i represents the current token. To obtain the final target dictionary, we convert the frequency count of the dictionary into actual frequencies. Subsequently, the attacker will construct two dictionaries for both non-watermarked and watermarked texts.

Due to the increased complexity of the spoofing attack, we can observe that the number of keys required for statistical analysis increases exponentially. In the worst-case scenario, the number of keys that n-gram spoofing attack needs to count is $|V|^n$, where $|V|$ is the size of the vocabulary.

We then leverage the dictionary to influence the generation. In this step, we need to determine which token needs to increase the probability when the LLM generates tokens. The spoofing attack influences this step in a manner similar to

TABLE III
THE AUCROC VALUE OF ALL WATERMARKS FOR DETECTABILITY EVALUATION.

	OPT-2.7b		Llama3-8b	
	C4	Quora-QA	C4	Quora-QA
UNIW	0.998	1.000	0.997	0.998
KGW1	1.000	1.000	0.997	1.000
KGW2	1.000	1.000	0.997	0.999
KGW3	0.999	1.000	0.995	1.000
KGW4	0.998	1.000	0.999	1.000
BW1	0.999	1.000	0.998	0.999
BW2	0.998	1.000	0.999	1.000
BW3	1.000	1.000	0.999	1.000
BW4	0.999	1.000	0.998	1.000

TABLE IV
THE PERPLEXITY VALUE OF ALL WATERMARKS FOR TEXT QUALITY EVALUATION. ORIGINAL MEANS WHEN GENERATING WITHOUT ADDING ANY WATERMARK.

	OPT-2.7b		Llama3-8b	
	C4	Quora-QA	C4	Quora-QA
Original	4.388	5.054	3.065	2.668
UNIW	6.273	8.362	4.004	3.251
KGW1	6.784	8.164	4.284	3.598
KGW2	7.007	8.146	4.401	3.764
KGW3	7.363	7.799	4.561	3.911
KGW4	7.519	7.82	4.521	3.966
BW1	6.464	6.715	4.345	3.576
BW2	6.539	7.441	4.365	3.669
BW3	6.732	7.686	4.524	3.794
BW4	6.832	7.698	4.483	3.882

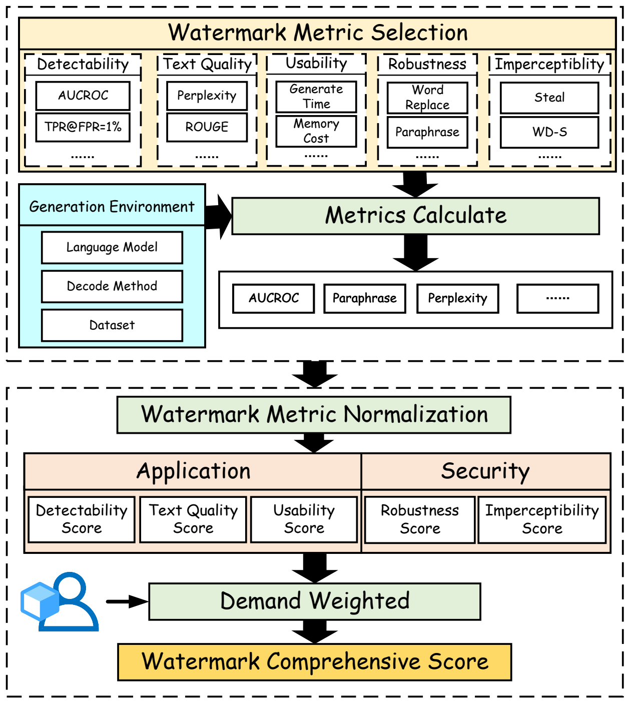


Fig. 4. An overview of CEFW. The upper area delimited by dotted lines is traditional watermark evaluation process, which is used by current watermark study. The bottom area shows the work of CEFW. CEFW combs the current watermark metrics to give five necessary characteristics score. Subsequently, CEFW introduces the Demand Weighted from LLM service providers to get a watermark comprehensive score.

KGW, by adding a certain bias to specific tokens to make them more likely to be generated. Unlike KGW, the spoofing attack determines how to add bias by comparing the two dictionaries.

Given the context ctx, A computes a score for the token T:

$$s(T, ctx) = \begin{cases} \frac{1}{2}min(\frac{\hat{p}_w(T|ctx)}{\hat{p}_b(T|ctx)}, 2), & \text{if } \frac{\hat{p}_w(T|ctx)}{\hat{p}_b(T|ctx)} \geq 1, \\ 0, & \text{otherwise.} \end{cases} \quad (10)$$

$\hat{p}_b(T|ctx)$ represents the frequency of the non-watermarked text, while $\hat{p}_w(T|ctx)$ represents the frequency of the watermarked text. This formula obtains the score for outlier high-frequency tokens, enabling the spoofing attack to determine which tokens are worth adding bias to and how much bias should be added.

In the final step, the spoofing attack multiplies the attack intensity by the score to obtain the final bias vector that will be added to the logits distribution.

B. *Experimental Result*

a) Detectability: For four generation environments, we generate 5000 watermarked texts and 5000 non-watermarked texts and each text has at most 200 max new tokens. For details of detectability evaluation, we show the AUCROC values in Table III.

b) Text Quality: Perplexity is selected as the text quality evaluation metric in CEFW, and Llama-2-13b[4] is the auxiliary language model. We utilize the same batch of data as for the detectability evaluation in the text quality evaluation and show the perplexity value of all watermarks in Table IV.

[4]https://huggingface.co/meta-llama/Llama-2-13b-hf

c) Usability: Three metrics are selected for usability. For time aspect, two metrics are the time needed to detect 5000 texts and the time to generate 5000 texts. For memory aspect, we sum the size of the language model and the size of the additional variable introduced by the watermark as the metric of Memory Cost. For example, OPT-2.7b is 5057.7MB and logits bias vector from KGW is 0.191MB. Therefore, $M_{MemoryCost}(Y)$ is 5057.7 and $M_{MemoryCost}(\hat{Y})$ is 5057.891. The result of three metrics for usability is shown in Table V.

d) Robustness: We show the AUCROC values before and after the DIPPER attack in Table VI. The AUCROC value without attack is different from the value in detectability, because we only select 500 texts to attack in robustness, but 5000 for detectability.

e) Imperceptibility: For each watermark, we generate 5000 watermarked texts and 5000 non-watermarked texts for n-gram frequency statistics. After learning the watermark, each STEAL attack generates 500 spoofing texts. In the end, we calculate their AUCROC to evaluate imperceptibility.

We show our result of the imperceptibility evaluation in Table II.

TABLE V
THE METRIC VALUES OF ALL WATERMARKS FOR USABILITY EVALUATION. t_g IS THE TIME TO GENERATE 5000 TEXTS; t_d IS THE TIME TO DETECT 5000 TEXTS; Mem IS THE METRIC OF MEMORY COST. ORIGINAL MEANS WHEN GENERATING WITHOUT ADDING ANY WATERMARK.

			Original	UNIW	KGW1	KGW2	KGW3	KGW4	BW1	BW2	BW3	BW4
OPT-2.7b	C4	t_g	18779	21379	22186	21488	20952	22038	21356	21515	21026	21256
		t_d	-	20.79	216.05	231.77	263.40	252.01	90.04	105.07	112.97	115.15
		Mem	5057.7	5057.9	5057.9	5057.9	5057.9	5057.9	5058.3	5058.3	5058.3	5058.3
	Quora-QA	t_g	20464	20862	20825	20861	20768	20878	20420	20811	20699	20298
		t_d	-	19.48	204.41	200.07	196.57	198.58	82.08	81.62	81.60	80.31
		Mem	5057.7	5057.9	5057.9	5057.9	5057.9	5057.9	5058.3	5058.3	5058.3	5058.3
Llama3-8b	C4	t_g	46342	45975	45611	45655	45724	46157	45648	46232	45803	45611
		t_d	-	33.80	299.45	309.03	275.47	281.09	124.27	123.29	110.08	117.67
		Mem	15316.5	15317.0	15317.0	15317.0	15317.0	15317.0	15318.1	15318.1	15318.1	15318.1
	Quora-QA	t_g	45603	45978	46197	46063	45987	45973	47330	47304	47418	51258
		t_d	-	18.85	209.41	206.15	208.32	208.67	96.69	98.32	98.41	94.84
		Mem	15316.5	15317.0	15317.0	15317.0	15317.0	15317.0	15318.1	15318.1	15318.1	15318.1

TABLE VI
THE AUCROC VALUE OF ALL WATERMARKS FOR ROBUSTNESS EVALUATION. NO ATTACK REPRESENTS THAT WATERMARK DO NOT SUFFER ANY SCRUBBING ATTACK. DIPPER REPRESENTS THAT THE WATERMARK HAS BEEN ATTACKED BY DIPPER.

			UNIW	KGW1	KGW2	KGW3	KGW4	BW1	BW2	BW3	BW4
OPT-2.7b	C4	No Attack	0.999	0.985	0.983	0.990	0.992	0.978	0.996	0.997	0.993
		DIPPER	0.992	0.814	0.758	0.720	0.674	0.822	0.783	0.729	0.719
	Quora-QA	No Attack	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
		DIPPER	0.965	0.895	0.867	0.758	0.704	0.937	0.865	0.813	0.776
Llama3-8b	C4	No Attack	0.999	0.999	1.000	0.999	0.999	0.999	1.000	1.000	1.000
		DIPPER	0.885	0.791	0.777	0.757	0.725	0.895	0.826	0.794	0.751
	Quora-QA	No Attack	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
		DIPPER	0.892	0.832	0.818	0.806	0.730	0.906	0.866	0.824	0.810

TABLE VII
WATERMARK EVALUATION METRICS

Detectability	TPR/TNR/FPR/FNR; AUCROC; Accuracy; TPR@FPR=X%; Bit Accuracy; Bit Error Rate.
Usability	Generate Time; Detect Time; Memory Cost.
Text Quality	Perplexity; BLEU; ROUGE; BERT Score; Entailment Score; Log Diversity.
Robustness	Emoji Prompt; Word (Insert/Delete/Exchange); Replace-Synonym; Replace-Context; HELM Perturbation; Human Modify; Paraphrase; Bigram-Paraphrase [11]; Back-translate; Re-watermark.
Imperceptibility	Steal [28]; 181-greenList [24]; WD-S [29].

C. Flexible Design

As shown in Fig. 4, CEFW can flexibly set the selection of watermark metrics and demand weights to better provide a suitable evaluation score for LLM service providers. CEFW first conducts a traditional watermark evaluation process to obtain some available metric values. Subsequently, CEFW classifies the corresponding characteristic and normalizes all metric values to ensure that each characteristic has one final score. Finally, CEFW utilizes simple weighting to bring in the actual needs of the LLM service provider.

We give some existing watermark metrics in Table VII and associate them with the corresponding watermark characteristics.

Although we can associate metrics with the corresponding watermark characteristics, LLM service providers cannot integrate different metrics due to their great difference in numerical feature. To obtain a comprehensive score, CEFW **designs some normalization functions for watermark metrics** to convert them into values in the range of 0 to 1. The consistent numerical features enable CEFW to integrate different metrics. Therefore, a comprehensive score of the watermark becomes computable.

The key to normalization is to set the upper and lower bounds. Give the upper bound V_u and lower bound V_l, for a metric value V, the normalization result $\hat{V}$ is:

$$\hat{V} = min(0, max(1, \frac{V - V_l}{V_u - V_l})) \tag{11}$$

Therefore, CEFW achieves different normalization methods by setting upper and lower bounds according to the following three principles:

a) Original Bounds: These metrics have their own upper and lower bounds, CEFW can directly normalize them. For example, AUCROC has upper bound 1 and lower bound 0.5. In watermark evaluation, these metrics are usually binary classification metrics for detectability. It is worth noting that the evaluation of imperceptibility is achieved by detecting cracked text generated by spoofing attacks, therefore the metric of imperceptibility can also utilize them.

b) Preset Bounds: Preset Bounds sets the upper and lower bounds for metrics by given values from LLM service providers or other evaluators. Theoretically, each metric supports Preset Bounds to achieve normalization. We take an example Detect Time, it is a metric that can only utilize Preset Bounds. CEFW regards it as the time to detect a piece of text and sets the upper bound to 0 seconds and the lower bound to 1 second.

c) Comparison Bounds: The upper and lower bounds of metrics suitable for Comparison Bounds are usually obtained by calculating the metric value of two target texts to be compared.

Robustness is a classic example. We calculate some detectability metrics to evaluate the original watermarked text and the watermarked text after scrubbing attack for robustness. The upper bound is set to the detectability metric value of the original watermark text, while the lower bound is the same as the lower bound of the metric.

Sometimes, the metric to compare has no lower bound, such as Perplexity, Generate Time, and Memory Cost. They only have upper bounds, which are the metric values of the generated text without watermark. Since the generation environment affects their upper bound values, it is unreasonable to assume a fixed lower bound for them. CEWF sets the lower bound as twice their upper bound value, indicating that the addition of the watermark causes the worst deterioration to be twice as bad.