EntroBench: Evaluating LLM Watermarking Under Multi-Entropy Scenarios and Practical User Operations

Pengyuan Qin[1]*, Linnan Tu[1]*, Yuhan Ke[2], Hefei Ling[1]†,
[1]Huazhong University of Science and Technology, [2]Wuhan University
{qpengyuan,lntu,lhefei}@hust.edu.cn; keyuhan@whu.edu.cn

Abstract

Large language models (LLMs) watermarking has been proposed as an active approach for content provenance verification, yet existing evaluations are largely confined to fixed entropy settings. In this paper, we introduce EntroBench, a benchmark for LLM watermarking that systematically covers three entropy levels and seven representative tasks. We conducted a fair evaluation of eight watermarking methods through hyper-parameter search based on an anchored dataset. We find that current approaches struggle to perform consistently across different entropy levels. Our analysis reveals a clear trade-off between watermark detectability and downstream output quality that varies across tasks and entropy conditions. Furthermore, we assess watermark robustness under realistic user interaction scenarios and show that common, non-adversarial user behaviors can substantially degrade watermark signals. These results indicate that practical usage-driven perturbations pose a significant challenge to current watermarking techniques. EntroBench provides a unified evaluation framework for studying these issues and supports the development of more adaptive and robust LLM watermarking methods. Dataset and codes are available at https://github.com/py-qin/EntroBench.

1 Introduction

Large language models (LLMs) are demonstrating increasingly powerful text generation capabilities(Achiam et al., 2023), and their potential for misuse—such as generating fake news(Yi et al., 2025) or infringing copyright(Liu et al., 2024b)—has raised widespread concern. Verifying the provenance of LLM-generated content is therefore critical to ensuring trust, safety, and accountability.

Traditional detection methods perform passive detection based on classification(Guo et al., 2023; Sadasivan et al., 2023; Li et al., 2024; Mitchell et al., 2023; OpenAI, 2023), while active-protection approaches such as LLM watermarking, which embed imperceptible statistical signals into the LLM generation process, have emerged as a promising solution(Kirchenbauer et al., 2023; Ren et al., 2024b,a; He et al., 2024; Lee et al., 2024; Christ et al., 2024; Dathathri et al., 2024).Specifically, LLM watermarking provides theoretical guarantees for the detectability of embedded signals by performing statistical inference on the generated text and testing the null hypothesis(Kirchenbauer et al., 2023). Since watermarking modifies the original output, it is crucial to minimize its impact on text quality while preserving the detectability of the watermark(Liu and Bu, 2024; Dathathri et al., 2024).

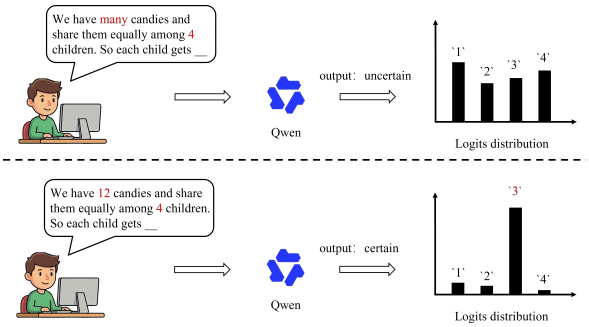


Figure 1: Illustration of next-token probability distributions under varying context constraints. The figure contrasts a high-entropy setting with a uniform-like distribution (top) against a low-entropy setting with a sharp, deterministic distribution (bottom) induced by informative context.

Existing research on LLM watermarking primarily evaluates the quality and detectability of watermarked text under single-entropy settings, overlooking the assessment of multi-entropy capabilities. In real-world scenarios, LLM watermarking must handle tasks with diverse entropy distributions, and evaluations under a single-

*Equal contribution.
†Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 42101–42118

entropy setting may not provide a comprehensive assessment. Existing approaches either focus on high-entropy scenarios by evaluating watermarks through text completion on the C4 RealNewsLike dataset(Kirchenbauer et al., 2023; Liu et al., 2024a; Raffel et al., 2020), or target only low-entropy, domain-specific settings such as code generation benchmarks(Kim et al., 2025; Lee et al., 2024). Although some methods have been evaluated on both C4 and MBPP(Austin et al., 2021), they lack a thorough analysis of how the entropy levels of different tasks affect watermark detectability and text quality.(Lu et al., 2024). Moreover, existing LLM watermarking methods have largely overlooked mixed-entropy scenarios, such as combinations of explanatory text and corresponding code.

Most studies evaluate robustness under paraphrasing and translation(Wang et al., 2025a; Kirchenbauer et al., 2023; He et al., 2024), focusing primarily on adversarial attack scenarios. However, this perspective often overlooks the more common real-world setting, where users interact with LLMs to fulfill practical needs, such as summarizing or content refinement, rather than deliberately attempting to remove or circumvent watermarks. For example, in code-assist scenarios, users typically copy only the code snippet from the LLM's response and discard the explanatory text.

Our study shows that current watermarking methods face significant challenges in multi-entropy environments. In real-world settings, when LLM users perform practical operations to extract valuable information, this non-malicious behavior can be even more detrimental than deliberate attacks, often causing existing LLM watermarks to fail. Our code and data will be made publicly available upon publication. The main contributions of this paper are as follows:

- We introduce **EntroBench**, the first systematic evaluation framework that covers diverse entropy scenarios and tasks, analyzing the risks posed to current LLM watermarking schemes through 3 major entropy scenarios and 7 core tasks.
- We present a fair evaluation of 8 LLM watermarking methods across diverse entropy levels, leveraging an anchored dataset for hyperparameter search, with a focus on watermark detectability and output quality.
- We introduce *LLM-as-a-User*, a framework that simulates real-world user behaviors, specifically *Summarize*, *Bullet Points*, and *Code Only*, to assess watermark detectability under practical conditions.

2 Related Work

2.1 LLMs Watermarking

The current mainstream LLM watermarking methods during the generation stage can be categorized into logits-based and sampling-based approaches.

Logits-based Watermarking. The pioneering KGW method(Kirchenbauer et al., 2023) uses a hash key to partition the vocabulary into red-green list, preferentially promoting green tokens during generation. To improve robustness, Zhao et al. (2023) simplifies the red-green list scheme by partitioning the vocabulary into a single global list. Ren et al. (2024b) uses prior probabilities to divide tokens into two distinct groups. For semantic preservation, Liu et al. (2023) uses the semantics of preceding tokens to determine the logits for watermarking. Wu et al. (2024b) applies a distribution-preserving reweighting function to preserve the original output distribution in expectation. Liu and Bu (2024) generates a semantics-based logits scaling vector to apply perturbations. Wang et al. (2025b) dynamically estimates the probability of green-list tokens to adjust watermark strength, minimizing perturbation to the model's natural output distribution.

Sampling-based Watermarking. For token-level watermarking, Kuditipudi et al. (2023) introduces watermark key control during the sampling through inverse sampling and Gumbel-max. Christ et al. (2024) uses pseudo-random number to guide token generation. Aaronson (2022) employs exponential minimum sampling to embed watermarks, and Dathathri et al. (2024) designs tournament sampling to maintain text quality while ensuring high detection accuracy. At the sentence level, Hou et al. (2024a) uses locality-sensitive hashing to partition the semantic space into watermarked and non-watermarked regions, and Hou et al. (2024b) simplifies this process by applying the K-means clustering(McQueen, 1967).

2.2 Entropy-Aware Watermarking

Several entropy-aware approaches have been proposed to address low-entropy scenarios. However, they primarily focus on watermarking performance in code generation settings. For instance,

LLM Watermarking Benchmark	Entropy-Stratified Datasets	Multi-Entropy Tasks Covering	User Operations
WaterBench(Tu et al., 2024)	×	√	×
CodeWMBench(Wu et al., 2024a)	×	×	×
MarkLLM(Pan et al., 2024)	×	√	×
WaterPark(Liang et al., 2025)	×	√	×
UWBENCH(Zhang et al., 2025)	×	×	×
MARKMYWORDS(Piet et al., 2025)	×	√	×
Lingual(Al Ghanim et al., 2025)	×	×	×
EntroBench (ours)	√	√	√

Table 1: Comparison with existing LLM watermarking benchmarks. The column *Multi-Entropy Tasks Covering* denotes whether it includes multiple tasks spanning different entropy scenarios. The column *User Operations* column indicates whether user behavior simulation is supported during post-processing.

Lee et al. (2024) focuses on watermarking high-entropy tokens, while Lu et al. (2024) applies entropy-weighted adjustments to tokens. To improve efficiency, Liu and Bu (2024) employs an auxiliary model to adaptively apply watermarks to high-entropy tokens, and Gu et al. (2025) uses a lightweight tagger to predict token entropy. Huang et al. (2025) focuses on strong evidence embedded in low-entropy tokens, thereby improving watermark detection accuracy in low-entropy scenarios.

2.3 LLMs Watermarking Benchmark

Recent work has attempted to establish benchmark evaluations for the performance of various watermarking methods. However, existing studies either focus solely on watermark effectiveness in single-entropy settings or offer a limited assessment of robustness under practical conditions. For example, Pan et al. (2024) provides a unified, extensible framework and a user-friendly interface for LLM watermarking. Tu et al. (2024) adjusts watermark hyper-parameters to a uniform strength level to enable fair performance evaluation. Wu et al. (2024a) systematically evaluates the performance of code watermarking. Al Ghanim et al. (2025) conducts watermark evaluations across multiple languages. Liang et al. (2025) reveals the impact of various design choices on attack robustness. Zhang et al. (2025) provides a systematic evaluation of unbiased watermarking methods. Piet et al. (2025) assesses watermarks across multiple dimensions, including quality, effectiveness, and tamper resistance.

3 EntroBench

3.1 Threat Model

We propose a threat model for LLM watermarking that captures a common real-world behavior: users typically retain only a critical subset of the LLM-generated content or restructure it through structured distillation.

To address this, we propose *LLM-as-a-User*, which leverages an LLM to simulate the diverse operations that real users commonly apply during post-processing. Specifically, we model the following three representative behaviors: (1) *Summarize*: Users extract the core ideas from the generated content. (2) *Bullet Points*: Users extract key points from the context and present them as a list of discrete, concise items. (3) *Code Only*: In hybrid tasks (e.g., code generation with explanatory text), users retain only the code snippet and discard the natural language explanation.

These operations are not malicious attacks but natural behaviors in LLM usage. However, they inherently involve compression and restructuring of the original output, which can lead to the loss of watermark signals.

Attacker knowledge : We assume the attacker (*i.e.*, the user) may be unaware of the watermarking technique. They act to make LLM output more practical, more informative and easier to read. Therefore, watermark robustness must hold under scenarios of unintentional removal. We assume the attacker has access to LLM chat interfaces and APIs, and is capable of performing operations within the *LLM-as-a-User* framework.

Real-world Scenarios : In software development, developers use LLMs to generate code ac-

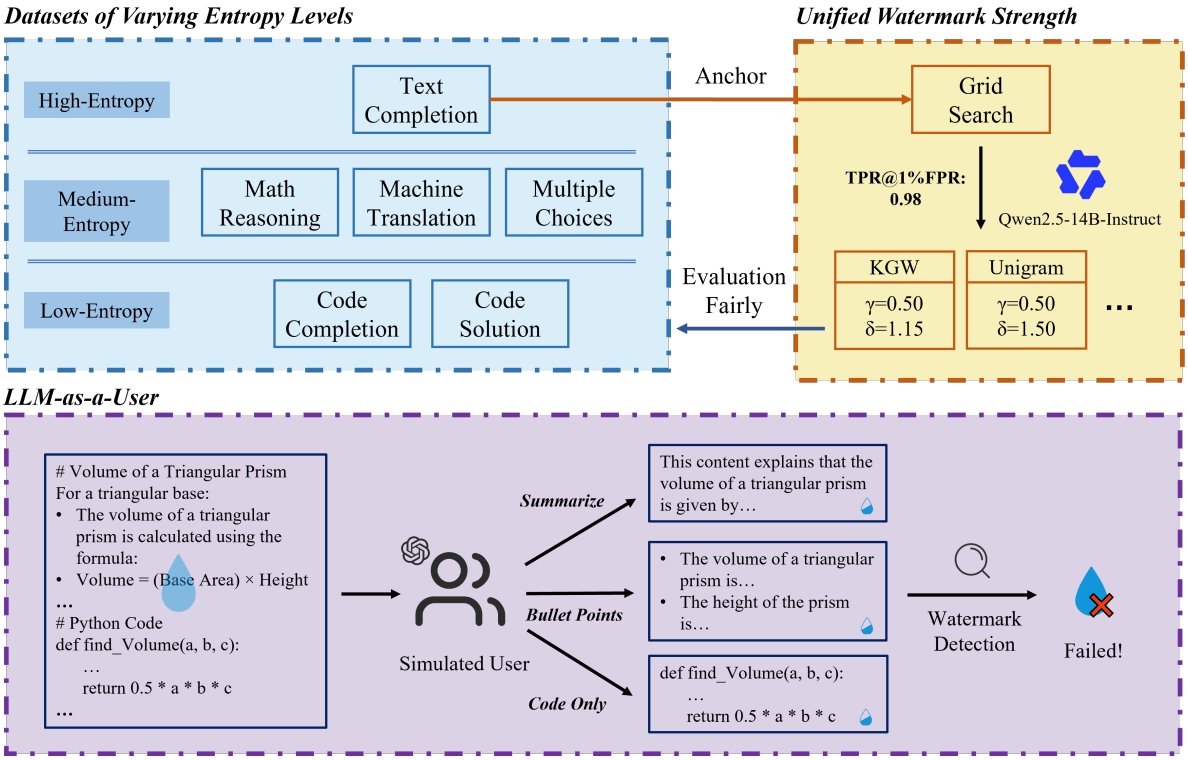


Figure 2: An illustration of the evaluation pipeline on EntroBench. Given a LLM, we first select the high-entropy dataset as the anchor and align the watermark strength across methods to ensure a fair comparison. We then evaluate detection performance and downstream quality across datasets spanning different entropy levels. To assess real-world robustness, we further process watermarked texts through three *LLM-as-a-User* pipelines and measure their detectability.

companied by natural-language explanations, but ultimately integrate only the plain code into their projects. In writing tasks, users prompt LLMs to produce background analyses or literature reviews, then extract key points for inclusion in formal documents. In office work, employees often convert LLM-generated descriptive text into bullet points for weekly reports.

These common scenarios are rarely considered in current watermark threat models. A robust watermark must not only withstand adversarial attacks but also maintain reliability in real-world LLM usage settings.

3.2 Dataset Entropy

The entropy of a token is measured by the degree of spread-out of the logits generated by the LLMs during that token's sampling process(Kirchenbauer et al., 2023; Lu et al., 2024). In this paper, we follow the spike entropy definition introduced by Kirchenbauer et al. (2023).

$$S(p,z) = \sum_k \frac{p_k}{1+zp_k} \quad (1)$$

where p_k represents the probability vector for each token k in the vocabulary during the sampling, and z is a scalar. The spike entropy reaches its minimum value of $\frac{1}{1+z}$ when the logits are concentrated on a single token, and its maximum value of $\frac{N}{N+z}$ in the case of a uniform distribution across all tokens.

To systematically investigate the performance of watermarking on datasets with different entropy levels, we divide the datasets into three categories: low, medium, and high, based on their overall entropy. To quantify the entropy of a dataset $D = \left\{x^{(1)}, \ldots, x^{(M)}\right\}$, where each sample $x^{(i)} = (x_1^{(i)}, \ldots, x_{L_i}^{(i)})$ is a sequence of tokens, we compute the spike entropy at the token level using a variety of models with different sizes and architectures, as shown in Figure 4.

For a given sample $x^{(i)}$, let $p_k^{(i,t)}$ denote the model's predicted probability of token k at position t. The spike entropy at position t is defined as:

$$S^{(i,t)} = \sum_k \frac{p_k^{(i,t)}}{1+zp_k^{(i,t)}} \quad (2)$$

We then average over all valid token positions in the sample to obtain the sample-level entropy:

$$\overline{S}^{(i)} = \frac{1}{L_i - 1} \sum_{t=1}^{L_i - 1} S^{(i,t)} \tag{3}$$

where we exclude the first position (as it has no preceding context for prediction in autoregressive models).

Finally, the dataset-level entropy is computed by averaging over all samples:

$$\overline{S}(D) = \frac{1}{M} \sum_{i=1}^{M} \overline{S}^{(i)} \tag{4}$$

3.3 Multi-Entropy Task

Due to the entropy complexity of real-world generation tasks, existing LLM watermarking methods are typically evaluated under single-entropy settings, leading to an incomplete understanding of watermark robustness across diverse scenarios. To address this gap, we introduce a multi-entropy, multi-task benchmark for comprehensive watermark evaluation under practical conditions. More details are provided in Appendix B.

High-Entropy Tasks. Due to the lack of strict semantic or syntactic constraints, the model has high freedom in token selection, making it well suited for evaluating watermarks under ideal conditions. Following prior works(Kirchenbauer et al., 2023; Liang et al., 2025), We select 500 samples from C4 dataset (Raffel et al., 2020) as **General-purpose text generation** task. Specifically, we split the *text* field of the C4 dataset and use the first approximately 30 tokens of each sample as a prompt to feed into a LLM for generating the subsequent 200 tokens. The corresponding continuation from the original text serves as the non-watermarked text.

Medium-Entropy Tasks. Medium-entropy tasks involve scenarios where the generation is partially constrained by logical reasoning or specific content requirements, yet retains flexibility in expression. For this, we select 3 tasks: (1) **Multiple choice questions.** Multiple choice questions (e.g., MMLU(Hendrycks et al., 2020)) typically yield single-token answers. To adapt this for watermark evaluation, we prompt a LLM to rewrite the correct option from MMLU (100 samples of $high\_school\_computer\_science$ subset) into a complete, coherent sentence (see Appendix B). This maintains the factual constraint while providing sufficient token length for watermark embedding. (2) **MATH problems.** We randomly select 200 samples from each of two mathematical reasoning benchmarks: GSM8K(Cobbe et al., 2021), which consists of grade-school-level word problems, and MATH-500(Hendrycks et al., 2021), a more challenging dataset covering advanced topics in algebra, geometry, and beyond. (3) **Machine translation.** We select 100 English–Chinese sentence pairs from the WMT23 test set (Kocmi et al., 2023), keeping only those where the English source is longer than the average length to allow sufficient space for watermark embedding. For each Chinese sentence, we prompt a watermarked LLM to generate English translation. The original English reference serves as the non-watermarked text.

Low-Entropy Tasks. Low-entropy tasks are characterized by a "peaked" output distribution, where valid next tokens are tightly constrained. This property makes watermarking challenging since perturbations to high-probability tokens are more likely to violate task constraints and introduce errors.**Code generation** is a canonical example of such low-entropy tasks due to its strict syntactic and semantic constraints. We randomly sample 200 programming problems from MBPP(Austin et al., 2021), a widely used benchmark consisting of Python programming problems.

While pure code generation is a useful test setting, real-world use of LLMs goes beyond producing code alone. In practice, users often prompt models to generate code together with natural-language explanations or comments. Motivated by this observation, and unlike prior work (Lee et al., 2024; Lu et al., 2024; Kim et al., 2025) that considers only pure code generation, we build a mixed-format programming dataset that combines code with text, better reflecting how real users interact with LLMs when solving programming tasks. Specifically, we prompt GPT-4o(Achiam et al., 2023) with the original problem statements from MBPP and ask it to generate both a step-by-step analysis of the problem and the corresponding solution code, addressing the limitation that the original MBPP contains only bare code without any explanation. The prompt we use can be found in Appendix I.

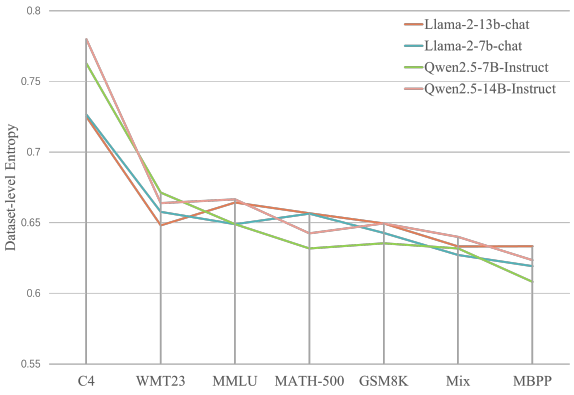


Figure 3: The dataset-level entropy across datasets evaluated by multiple models. Despite differences of architecture and model size, the relative ranking of datasets remains consistent, validating our categorization strategy.The scalar z is fixed to $z = 0.761$ following (Lu et al., 2024). We also conduct an ablation study on z in Appendix C.

3.4 Hyper-Parameter Search

Following (Tu et al., 2024), to fairly evaluate different watermarking algorithms, we adopt an anchor-dataset-based hyperparameter calibration strategy. This ensures that all methods achieve the same detectability on a common baseline, thereby focusing the evaluation on their adaptability to datasets with varying entropy distributions. We select the widely used C4 for hyper-parameter calibration. Since the hyper-parameters of different algorithms are not directly comparable, we perform a grid search(Alibrahim and Ludwig, 2021) for each algorithm on the C4 dataset.

3.5 Evaluation Metrics

To provide a comprehensive assessment of LLM watermarking, we employ an evaluation that covers both detectability and generation quality.

Detectability. Following previous studies(Tu et al., 2024; Liang et al., 2025), we evaluate detection performance using the **True Positive Rate (TPR)** at a low **False Positive Rate (FPR)** threshold (typically 1%), denoted as **TPR@1%FPR**. This metric reflects the reliability of the watermark detector in minimizing false accusations while effectively identifying watermarked content.

Generation Quality. A practical watermark must not degrade the text quality or the downstream ability. We assess this using task-specific metrics: (1) For open-ended generation tasks, we measure text quality using Perplexity (PPL) calculated by Llama-2-13B-chat(Touvron et al., 2023) and semantic consistency using BERTScore(Zhang et al., 2020) and P-SP(Wieting et al., 2022). (2) For code generation and mathematical reasoning tasks, we report Pass@1, which measures the percentage of generated solutions that pass unit tests or yield the correct final answer. (3) For machine translation, we introduce BLEU(Papineni et al., 2002). (4) For MMLU, we evaluate the Accuracy of the generated answers against the ground truth labels. More details can be found in Appendix E.

4 Experiments

4.1 Experimental Settings

Following prior work(Liang et al., 2025), we use Qwen2.5-14B-Instruct to generate watermarked text (a broader range of LLMs in Appendix D). We evaluate watermarking across various methods: Kirchenbauer et al. (2023); Zhao et al. (2023); Dathathri et al. (2024); Wu et al. (2024b); Wang et al. (2025b). We also introduce entropy-aware methods including Lee et al. (2024); Lu et al. (2024); Gu et al. (2025). All experiments are conducted using 4 Nvidia A800 GPUs (80GB) based on the MarkLLM repository*. We set the target TPR@1%FPR to 0.98. And through grid search, we selected the hyper-parameter combination that yields the smallest deviation from this target. The combination of watermarking parameters is summarized in Appendix F. More implementation details can be found in Appendix G. And the mechanisms of these watermarking algorithms are detailed in the Appendix H.

4.2 Main Results

Table 2 presents the performance of current LLM watermarking methods on EntroBench, from which several conclusions can be drawn:

No Single Method Universally Dominates Across All Tasks. Despite EWD achieving the highest average TPR@1%FPR on EntroBench, it does not consistently outperform other methods in all tasks across all entropy levels. For instance, methods like SynthID-Text and IE exhibit superior performance in specific sub-tasks. This underscores the diversity of watermarking challenges across varying entropy levels and the complexity of EntroBench, suggesting that different approaches excel in distinct aspects and are thus well-suited to a wide range of task requirements.

*https://github.com/THU-BPM/MarkLLM

Task (→)	High-Entropy	Medium-Entropy				Low-Entropy		Avg
Method ↓	C4	WMT23	GSM8K	MATH-500	MMLU	MBPP	Mix	
KGW	0.980	0.170	0.330	0.135	0.610	0.115	0.045	0.341
Unigram	0.980	**0.480**	0.300	0.195	0.800	0.280	0.065	0.443
SynthID-Text	0.980	0.140	**0.585**	0.270	0.840	0.015	**0.810**	0.520
DiPMark	0.980	0.090	0.485	0.440	0.740	0.250	0.365	0.479
MorphMark	0.980	0.150	0.210	0.060	0.440	0.155	0.010	0.286
SWEET	0.980	0.100	0.415	**0.460**	0.780	0.235	0.485	0.493
EWD	0.980	0.240	0.550	0.410	0.810	0.360	0.675	**0.575**
IE	0.980	0.310	0.250	0.235	**0.900**	**0.530**	0.735	0.562

Table 2: TPR@1%FPR of various LLM watermarking methods on EntroBench. Best score in bold.

Demonstration of Benchmark Diversity and Challenge. Models that perform well on mainstream LLM watermarking benchmarks do not consistently maintain high performance across all tasks of EntroBench. For example, KGW and Unigram achieve high scores on C4 but show poor performance on low-entropy tasks. This performance variation reflects the challenge posed by EntroBench and suggests a need for LLM watermarking methods that can adapt to different entropy levels.

Low Entropy Does Not Necessarily Imply Difficulty. While low-entropy tasks are often assumed to be more challenging due to their constrained output spaces, our results demonstrate that this is not universally true. For instance, the Mix benchmark which classified as low-entropy yields high scores for SynthID-Text and several entropy-aware methods. This suggests that task difficulty is determined not only by entropy level, but also by the data composition and how well the watermarking strategy adapts to these characteristics.

Effectiveness of Entropy-Aware Watermarking Strategies. Methods that incorporate adaptive or entropy-aware mechanisms, such as EWD and IE, show consistently solid performance across tasks. EWD achieves strong results on several medium-entropy and low-entropy benchmarks, suggesting it handles constrained outputs effectively and illustrating the potential benefits of entropy-aware watermarking strategies.

Figure 4 presents box plots of all methods. EWD achieves the highest performance, demonstrating strong adaptability across varying entropy levels. Entropy-aware methods, in general, exhibit superior multi-entropy adaptability compared to other approaches. Methods that perform well on the conventional C4 dataset show notably poor adaptability under diverse entropy conditions. This highlights the challenge posed by EntroBench and its value as a benchmark for LLM watermarking research.

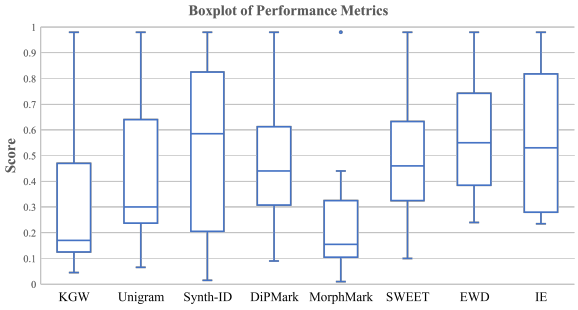
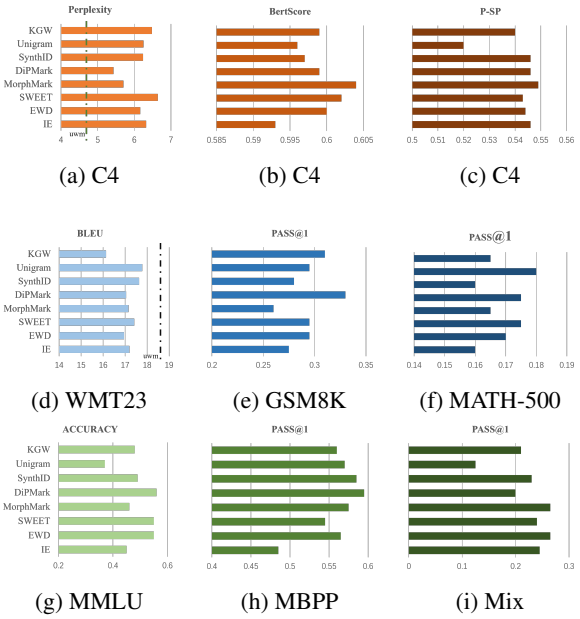


Figure 4: Box plots of performance across all methods on EntroBench.

4.3 Analysis of Downstream Quality

In practical scenarios, generation quality is also a key factor in evaluating watermarking methods, as it largely influences LLM users' willingness to adopt them. In this section, we assess the generation quality of watermarked outputs across various tasks to provide a comprehensive evaluation. The results are presented in Figure 5.

We have made a few interesting observations. (1) No watermarking method is able to consistently preserve downstream quality across datasets with varying entropy distributions. (2) DiPMark demonstrates the best overall performance, maintaining high levels of effectiveness across a wide range of downstream tasks. This can be attributed to DiPMark preserving the original token distribution during the watermarking process. (3) There is a



(a) C4 (b) C4 (c) C4

(d) WMT23 (e) GSM8K (f) MATH-500

(g) MMLU (h) MBPP (i) Mix

Figure 5: Downstream Quality of different methods across all tasks.The *uwm* means unwatermarked text.

Method ↓	Summarize	Bullet Points	Code Only
GPT-4o			
KGW	0.266	0.320	0.020
Unigram	**0.494**	0.540	0.045
SynthID-Text	0.078	0.210	0.125
DipMark	0.026	0.028	0.090
MorphMark	0.176	0.238	0.010
MorphMark	0.176	0.238	0.010
SWEET	0.234	0.534	0.080
EWD	0.244	**0.598**	0.180
IE	0.144	0.328	**0.200**
Llama2-13B-chat			
KGW	0.058	0.558	0.015
Unigram	**0.356**	**0.720**	0.055
SynthID-Text	0.054	0.238	0.070
DipMark	0.040	0.154	0.065
MorphMark	0.178	0.390	0.010
SWEET	0.168	0.474	0.020
EWD	0.214	0.458	0.015
IE	0.150	0.470	**0.080**

Table 3: Effects of different user operations. Best score in bold.

trade-off between detectability and downstream quality. Methods like EWD, despite their excellent multi-entropy detectability, suffer from degraded downstream quality. This discrepancy highlights a common trade-off in LLM watermarking systems.

4.4 Watermarking Resilience to User Operations

This section investigates the impact of *LLM-as-a-User* on watermarking performance, as measured by the TPR@1%FPR, and the detection thresholds are recalibrated on edited non-watermarked text generated by the same operation. We evaluate the *Summarize* and *Bullet points* on C4, and the *Code only* on Mix. We employ both open-source and closed-source LLMs as surrogate user models, including GPT-4o and Llama2-13B-chat. For entropy-aware methods, entropy is re-estimated on the edited text using the original prompt, and the same LLM as used during generation (IE with its entropy predictor). The results are presented in Table 3. The prompts used to simulate user behavior are provided in the Appendix I.

Regarding watermarking resilience, We can draw the following observations: (1) All methods are vulnerable to our *LLM-as-a-User* pipeline. The performance of many methods on the C4 dataset drops from 0.98 to levels as low as 0.1 or 0.2. Among them, Unigram demonstrates the most robust performance in both the *summarize* and *bullet points* operations, possibly due to its global red-green vocabulary strategy, which makes it highly sensitive to the key information retained after processing. (2) Entropy-aware methods suffer greater performance degradation after user operations, likely because the removal of surrounding context disrupts entropy estimation, leading to unreliable watermark signals.

5 Conclusion

In this paper, we introduce EntroBench, a benchmark for LLM watermarking that covers 3 entropy levels and 7 tasks. We evaluate 8 watermarking methods fairly on EntroBench and find that current approaches struggle to perform well across different entropy settings, showing the challenge of our benchmark. Our analysis further uncovers a trade-off between detectability and downstream quality across multiple tasks, underscoring the need for watermarking techniques that can adapt to varying entropy levels while preserving output quality. Moreover, we assess watermark robustness under user operations, exposing their vulnerability when confronted with practical, usage-driven perturbations. Through EntroBench, we aim to catalyze progress in the field of LLM watermarking by en-

couraging the development of more adaptive, robust, and broadly applicable methods that better serve the needs of the research community and real-world deployment.

Limitations

Despite the comprehensive evaluation framework provided by EntroBench, there are still some limitations to be addressed:

(1) **Limited Diversity**: Although we evaluated watermarks across tasks with multiple entropy levels, we believe it is necessary to include an even wider range of entropy levels and more diverse task types.

(2) **Simplified Modeling**: We simulated only three basic user behaviors, but real user actions are far more complex and varied. Thus, we believe finer-grained modeling of user behavior is needed.

(3) **Restricted Model Scope**: The experiments were conducted on only a limited set of LLMs; future work should explore a broader range of model architectures and sizes.

Ethical Considerations

In this section we will discuss the ethical considerations for our work.

Licenses. We ensure that the use of each dataset comply with its respective license. The C4 RealNewsLike(Raffel et al., 2020) is provided under the ODC-BY license. The original data were released by the WMT23(Kocmi et al., 2023) shared task for research use. The MMLU(Hendrycks et al., 2020) is provided under the MIT license. The GSM8K(Cobbe et al., 2021) is provided under the MIT license. The MATH-500(Hendrycks et al., 2021) is provided under the MIT license, and the MBPP(Austin et al., 2021) is provided under the CC-BY-4.0 license. Additionally, the Licenses for the LLMs are also available. Llama2-7B-chat and Llama2-13B-chat(Touvron et al., 2023) are released under the Meta License which needs to apply on their web sites. Qwen-2.5-7B-Instruct and Qwen-2.5-14B-Instruct(Yang et al., 2024) are shared under the Apache-2.0 license.

Considerations for AI Systems. Our work reveals that current LLM watermarking techniques can be inadvertently bypassed under real-world user behaviors, potentially enabling malicious actors to evade detection and compromise digital authenticity and cybersecurity. While we acknowledge this finding could be misused, our goal is not to provide attack methods, but to expose the limitations of existing LLM watermarks in practical settings, thereby encouraging the development of more robust, user-resilient solutions. We advocate for responsible disclosure and call on the community to jointly strengthen the security and reliability of watermarking technologies.

References

Scott Aaronson. 2022. My AI safety lecture for UT effective altruism. https://scottaaronson.blog/?p=6823.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Mansour Al Ghanim, Jiaqi Xue, Rochana Prih Hastuti, Mengxin Zheng, Yan Solihin, and Qian Lou. 2025. Evaluating the robustness and accuracy of text watermarking under real-world cross-lingual manipulations. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 7396–7416, Suzhou, China. Association for Computational Linguistics.

Hussain Alibrahim and Simone A Ludwig. 2021. Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization. In *2021 IEEE congress on evolutionary computation (CEC)*, pages 1551–1559. IEEE.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Miranda Christ, Sam Gunn, and Or Zamir. 2024. Undetectable watermarks for language models. In *Proceedings of Thirty Seventh Conference on Learning Theory*, volume 247 of *Proceedings of Machine Learning Research*, pages 1125–1139. PMLR.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, and 1 others. 2024. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823.

Tianle Gu, Zongqi Wang, Kexin Huang, Yuanqi Yao, Xiangliang Zhang, Yujiu Yang, and Xiuying Chen. 2025. Invisible entropy: Towards safe and efficient low-entropy LLM watermarking. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6716–6733, Suzhou, China. Association for Computational Linguistics.

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. 2023. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arXiv:2301.07597*.

Zhiwei He, Binglin Zhou, Hongkun Hao, Aiwei Liu, Xing Wang, Zhaopeng Tu, Zhuosheng Zhang, and Rui Wang. 2024. Can watermarks survive translation? on the cross-lingual consistency of text watermark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4115–4129, Bangkok, Thailand. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

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

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

Beining Huang, Du Su, Fei Sun, Qi Cao, Huawei Shen, and Xueqi Cheng. 2025. Low-entropy watermark detection via Bayes' rule derived detector. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 14330–14344, Vienna, Austria. Association for Computational Linguistics.

Jungin Kim, Shinwoo Park, and Yo-Sub Han. 2025. Marking code without breaking it: Code watermarking for detecting llm-generated code. *arXiv preprint arXiv:2502.18851*.

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

Tom Kocmi, Eleftherios Avramidis, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Markus Freitag, Thamme Gowda, Roman Grundkiewicz, Barry Haddow, Philipp Koehn, Benjamin Marie, Christof Monz, Makoto Morishita, Kenton Murray, Masaaki Nagata, Toshiaki Nakazawa, Martin Popel, and 3 others. 2023. Findings of the 2023 conference on machine translation (WMT23): LLMs are here but not quite there yet. In *Proceedings of the Eighth Conference on Machine Translation*, pages 1–42, Singapore. Association for Computational Linguistics.

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. 2024. Who wrote this code? watermarking for code generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4890–4911, Bangkok, Thailand. Association for Computational Linguistics.

Yafu Li, Qintong Li, Leyang Cui, Wei Bi, Zhilin Wang, Longyue Wang, Linyi Yang, Shuming Shi, and Yue Zhang. 2024. Mage: Machine-generated text detection in the wild. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 36–53.

Jiacheng Liang, Zian Wang, Spencer Hong, Shouling Ji, and Ting Wang. 2025. Watermark under fire: A robustness evaluation of LLM watermarking. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 21050–21074, Suzhou, China. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Aiwei Liu, Leyi Pan, Xuming Hu, Shuang Li, Lijie Wen, Irwin King, and Philip S Yu. 2024a. An unforgeable publicly verifiable watermark for large language models. In *ICLR*.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. 2023. A semantic invariant robust watermark for large language models. *arXiv preprint arXiv:2310.06356*.

Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip Yu. 2024b. A survey of text watermarking

in the era of large language models. *ACM Comput. Surv.*, 57(2).

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. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11724–11735, Bangkok, Thailand. Association for Computational Linguistics.

James B McQueen. 1967. Some methods of classification and analysis of multivariate observations. In *Proc. of 5th Berkeley Symposium on Math. Stat. and Prob.*, pages 281–297.

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

OpenAI. 2023. New AI Classifier for Indicating AI-Written Text.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Julien Piet, Chawin Sitawarin, Vivian Fang, Norman Mu, and David Wagner. 2025. Markmywords: Analyzing and evaluating language model watermarks. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 68–91. IEEE.

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. 2024a. A robust semantics-based watermark for large language model against paraphrasing. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 613–625, Mexico City, Mexico. Association for Computational Linguistics.

Yubing Ren, Ping Guo, Yanan Cao, and Wei Ma. 2024b. Subtle signatures, strong shields: Advancing robust and imperceptible watermarking in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5508–5519, Bangkok, Thailand. Association for Computational Linguistics.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Shangqing Tu, Yuliang Sun, Yushi Bai, Jifan Yu, Lei Hou, and Juanzi Li. 2024. WaterBench: Towards holistic evaluation of watermarks for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1517–1542, Bangkok, Thailand. Association for Computational Linguistics.

Yidan Wang, Yubing Ren, Yanan Cao, and Binxing Fang. 2025a. From trade-off to synergy: A versatile symbiotic watermarking framework for large language models. *arXiv preprint arXiv:2505.09924*.

Zongqi Wang, Tianle Gu, Baoyuan Wu, and Yujiu Yang. 2025b. MorphMark: Flexible adaptive watermarking for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4842–4860, Vienna, Austria. Association for Computational Linguistics.

John Wieting, Kevin Gimpel, Graham Neubig, and Taylor Berg-kirkpatrick. 2022. Paraphrastic representations at scale. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 379–388, Abu Dhabi, UAE. Association for Computational Linguistics.

BenLong Wu, Kejiang Chen, Yanru He, Guoqiang Chen, Weiming Zhang, and Nenghai Yu. 2024a. Codewmbench: An automated benchmark for code watermarking evaluation. In *Proceedings of the ACM Turing Award Celebration Conference-China 2024*, pages 120–125.

Yihan Wu, Zhengmian Hu, Junfeng Guo, Hongyang Zhang, and Heng Huang. 2024b. A resilient and accessible distribution-preserving watermark for large language models. In *International Conference on Machine Learning*, pages 53443–53470. PMLR.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian

Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Jingyuan Yi, Zeqiu Xu, Tianyi Huang, and Peiyang Yu. 2025. Challenges and innovations in llm-powered fake news detection: A synthesis of approaches and future directions. In *Proceedings of the 2025 2nd International Conference on Generative Artificial Intelligence and Information Security*, pages 87–93.

Da Zhang, Chenggang Rong, Bingyu Li, Feiyu Wang, Zhiyuan Zhao, Junyu Gao, and Xuelong Li. 2025. Uwbench: A comprehensive vision-language benchmark for underwater understanding. *arXiv preprint arXiv:2510.18262*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *Proceedings of International Conference on Learning Representations*.

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

A LLM use

We used LLMs to assist with language editing and improve the clarity and fluency of the text.

B Dataset Descriptions

C4 RealNewslike(Raffel et al., 2020). A subset of the Colossal Clean Crawled Corpus, filtered to retain only documents resembling news articles. Due to its journalistic style, diverse topics, and fluent narrative structure, C4 generates high-entropy outputs when used as prompts for open-ended continuation. It is widely adopted in language modeling and watermarking studies.

GSM8K(Cobbe et al., 2021). A dataset of 8.5K grade-school math word problems, each requiring multi-step arithmetic reasoning. Correct answers are numerical, but models often generate chain-of-thought rationales before the final answer. While the solution path can vary, the underlying logic constrains token choices, placing GSM8K in the medium-entropy regime.

MATH-500(Hendrycks et al., 2021). A subset of the MATH dataset(Lightman et al., 2023), containing 500 advanced mathematics problems spanning algebra, geometry, and calculus. Solutions require formal symbolic reasoning and are typically expressed in LaTeX. The need for precise notation and limited valid solution forms reduces linguistic freedom compared to open text, yet allows more variation than code, justifying its medium-entropy classification.

MMLU(Hendrycks et al., 2020). The Massive Multitask Language Understanding contains exam-style questions from 57 subject areas, including mathematics, physics, law, and medicine, spanning difficulty levels from high school to professional. It is designed to evaluate a model's real-world comprehension and its ability to solve interdisciplinary problems. We leverage a LLM to expand a single-token answer into a coherent and complete response, ensuring that the output remains grounded in factual knowledge while providing ample embedding space for watermarking. The prompt used for rewriting MMLU is provided in Table 4.

MBPP(Austin et al., 2021). It contains 974 short Python programming tasks described in natural language. Generated outputs must be syntactically valid and functionally correct code, which imposes strong structural priors. This results in highly predictable token sequences, characteristic of low entropy.

Mix. A curated collection of 200 samples combining natural language explanations with embedded code snippets. It serves as a low-entropy testbed for watermarking robustness in hybrid text-code environments. This dataset is constructed using GPT-4o solely as a data generation tool to produce paired natural language descriptions and corresponding code snippets. GPT-4o is not used in any stage of watermark generation, entropy estimation, or detection. To ensure data quality, we remove samples with syntax errors, incomplete code, or abnormal lengths, and near-duplicate generations are removed. We further conduct manual spot checks to verify basic correctness and diversity.

C Ablation Study on the Effect of z

To perform a sensitivity analysis of z, we conduct ablation studies using Llama-2-7b-chat(Touvron et al., 2023). Specifically, we vary δ over the values $[1.7, 1.8, 1.9, 2.0, 2.1]$, which correspond to $z = [0.691, 0.716, 0.739, 0.761, 0.781]$, respectively. Using these z values, we compute dataset-level entropy scores. The results are shown in Figure 6.

Rewriting Template

You are given a multiple-choice question from the MMLU benchmark and its correct answer. Your task is to generate a single, fluent, declarative sentence that:

- Starts directly with the correct answer content (the letter label or number).
- Clearly and concisely explains or affirms the answer in the context of the question.
- Does not mention the question format, or meta-information.
- Output only the rewritten sentence.

Question: {question}
Choices: {choices}
Correct Answer: {answer}

Output:

Table 4: Instruction used to rewrite MMLU.

D More Results

We additionally performed detectability experiments on Qwen2.5-7B-Instruct(Yang et al., 2024) and Phi-3.5-mini-instruct. Results are provided in Table 5 and Table 6. Although LLMs differ in scale, their performance remains remarkably similar.

E Metric Details

E.1 Perplexity (PPL)

Perplexity (PPL) measures the uncertainty of a language model in predicting a given text sequence, with lower values indicating higher confidence and fluency. For a sequence $S = (s_1, \ldots, s_N)$ under a language model parameterized by θ, perplexity is defined as:

$$\mathrm{PPL}_\theta(S) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P_\theta(s_i \mid s_{<i})\right) \tag{5}$$

where $P_\theta(s_i \mid s_{<i})$ denotes the conditional probability of token s_i given the preceding context. We compute the PPL of watermarked text using the large language model to assess their confidence and fluency.

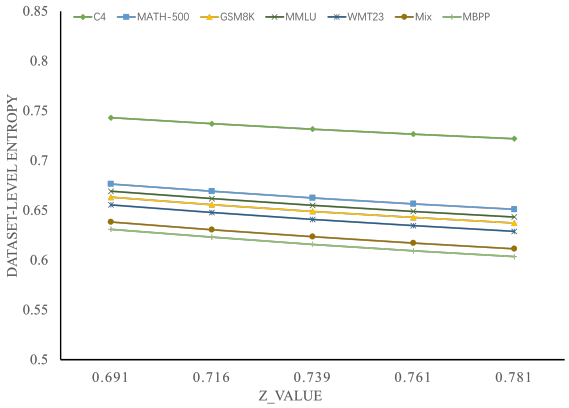


Figure 6: Dataset-level entropy under varying z

E.2 BERTScore

BERTScore(Zhang et al., 2020) evaluates the similarity between a candidate and a reference text by comparing their contextual embeddings derived from a pretrained transformer model. BERTScore captures both lexical and semantic alignment, enabling it to better assess meaning preservation even when word overlap is minimal. In our experiments, we compute BERTScore between watermarked text and unwatermarked text using deberta-xlarge-mnli to measure downstream performance in terms of content fidelity, with higher scores indicating stronger alignment.

E.3 P-SP

The Paraphrase Semantic Proximity (P-SP) metric(Wieting et al., 2022) is a learned semantic similarity measure specifically trained on large-scale paraphrase datasets to distinguish genuine paraphrases from semantically unrelated sentence pairs. Given an original sentence x and its rewritten version y, P-SP first encodes both sentences into fixed-dimensional embeddings using a paraphrase-aware encoder. The similarity between x and y is then quantified as the cosine similarity between their respective embeddings, computed as follows:

$$\mathrm{P{-}SP}(x, y) = \cos(g(x), g(y)) \tag{6}$$

where $g(\cdot)$ is the P-SP encoder. Higher P-SP scores indicate greater semantic equivalence, making it a suitable automatic metric for evaluating meaning preservation in text generation tasks. We use paraphrase-at-scale-english as the P-SP encoder.

E.4 BLEU

BLEU(Papineni et al., 2002) is a standard automatic metric for evaluating machine translation

Task (→)	High-Entropy	Medium-Entropy				Low-Entropy		Avg
Method ↓	C4	WMT23	GSM8K	MATH-500	MMLU	MBPP	Mix	
KGW	0.980	0.220	0.440	0.145	0.640	0.155	0.107	0.384
Unigram	0.980	**0.490**	0.275	0.105	**0.810**	0.255	0.024	0.420
SynthID-Text	0.980	0.130	0.325	0.305	0.630	0.070	**0.670**	0.444
DiPMark	0.980	0.150	0.365	0.310	0.540	0.135	0.265	0.392
MorphMark	0.980	0.190	0.250	0.145	0.560	0.015	0.025	0.309
SWEET	0.980	0.150	0.585	**0.490**	0.670	0.235	0.320	0.490
EWD	0.980	0.230	**0.640**	0.455	**0.810**	**0.390**	0.665	**0.596**
IE	0.980	0.260	0.155	0.175	0.730	0.375	0.445	0.445

Table 5: TPR@1%FPR of various LLM watermarking methods on EntroBench using Qwen2.5-7B-Instruct. Best score in bold.

Task (→)	High-Entropy	Medium-Entropy				Low-Entropy		Avg
Method ↓	C4	WMT23	GSM8K	MATH-500	MMLU	MBPP	Mix	
KGW	0.980	0.370	0.045	0.085	0.760	0.070	0.175	0.355
Unigram	0.980	0.270	**0.435**	0.150	0.450	0.420	0.245	0.421
SynthID	0.980	0.060	0.215	0.080	**0.960**	0.005	0.515	0.402
DIP	0.980	0.300	0.390	0.380	0.900	0.070	0.435	0.493
MorphMark	0.980	**0.390**	0.030	0.030	0.770	0.040	0.100	0.334
SWEET	0.980	0.240	0.235	0.205	0.690	0.220	0.435	0.429
EWD	0.980	0.250	0.390	**0.405**	0.850	0.460	**0.525**	**0.551**
IE	0.980	0.290	0.250	0.245	0.830	**0.560**	0.505	0.522

Table 6: TPR@1%FPR of various LLM watermarking methods on EntroBench using Phi-3.5-mini-instruct. Best score in bold.

quality by comparing a system-generated translation (candidate) against one or more human reference translations. It quantifies lexical similarity through modified n-gram precision and incorporates a brevity penalty to discourage overly short translations that may achieve artificially high precision. In our experiments, BLEU serves as a primary indicator of translation adequacy and fluency—higher scores reflect closer alignment with human references in terms of word choice and phrase structure. Formally, the BLEU score is computed as:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \quad (7)$$

where p_n is the modified precision for n-grams of order n, w_n is the corresponding weight, and BP denotes the brevity penalty. The brevity penalty BP equals 1 when the candidate length ℓ_c is at least the reference length ℓ_r; otherwise, it is set to $\exp(1 - \ell_r/\ell_c)$ to penalize excessively short outputs.

F Parameter Settings

All parameter settings can be found in Table 7.

G Implementation Details

The detectability and generation quality evaluation was carried out using 4 Nvidia A800 GPUs (80GB) and CUDA 12.4. The software versions employed in this study include PyTorch 2.6.0 and Transformers 4.57.3. Generation is performed with sampling (temperature = 0.7, top-k = 20, top-p = 0.8) to promote fluent and diverse outputs. The open-source models or APIs used in EntroBench are listed in Table 8. And the URLs of dataset resources used

Method	Parameter	Setting
KGW	γ	0.5
	δ	1.15
Unigram	γ	0.5
	δ	1.5
SynthID-Text	ngram_len	5
	mode	non-distortionary
DipMark	γ	0.5
	α	0.45
	z_threshold	1.517
MorphMark	γ	0.5
	type	linear
	k_linear	1.00
SWEET	γ	0.5
	δ	1.2
EWD	γ	0.5
	δ	1.2
IE	γ	0.5
	δ	3.0

Table 7: Parameter settings of watermarking methods on Qwen2.5-14B-Instruct via hyper-parameter search.

in EntroBench are listed in Table 9.

H LLM Watermarking Methods

KGW(Kirchenbauer et al., 2023). The KGW method embeds a watermark into generated text by adjusting the logits of the next token. The vocabulary $\mathcal{V}$ is partitioned into green and red lists according to a split ratio γ. A secret key S_k, combined with a hash of the preceding $k-1$ token IDs, seeds a pseudorandom number generator to determine this partitioning for the next token position k. During generation, the model either restricts token selection exclusively to the green list (hard watermarking) or biases selection toward it (soft watermarking) by adding a small constant δ to the logits of green-list tokens. KGW use the following equation to detect the watermark.

$$z = \frac{|s|_G - \gamma|s|}{\sqrt{|s|\gamma(1-\gamma)}} \tag{8}$$

where $|s|_G$ is the number of green tokens in the generated text, and $\gamma = \frac{|v_G|}{|v|}$.

Unigram(Zhao et al., 2023). Like KGW, Unigram partitions the vocabulary $\mathcal{V}$ into a green list and a red list. However, hese lists in Unigram are globally fixed and remain consistent throughout the generation process, as it does not use hashing based on previous tokens. Instead, it employs SHA-256 hashing with a secret key to partition the vocabulary and subsequently adjusts the logit values. Consequently, this approach is highly robust against the *LLM-as-a-user* proposed in this paper. Unigram uses the same detection method as KGW.

SWEET(Lee et al., 2024). Rather than applying watermarking uniformly across all generation steps, SWEET selectively embeds watermarks only at time steps where the model's predictive distribution exhibits high entropy, that is, where the generation process is inherently uncertain. At low-entropy steps, where token choices are strongly constrained by syntactic or semantic correctness, watermarking is deliberately omitted to preserve output fidelity. Building upon logit-based watermarking frameworks, SWEET employs a fixed entropy threshold τ during both generation and detection: only tokens whose entropy exceeds τ are considered for perturbation and subsequent statistical testing. This selective strategy enables effective watermark detection while minimizing adverse impacts on code correctness and quality. SWEET use the following equation to detect.

$$z = \frac{N_G^h - \gamma N^h}{\sqrt{N^h \gamma(1-\gamma)}} \tag{9}$$

where N^h denote the number of tokens that have an entropy value higher than the threshold τ, N_G^h denote the number of green tokens among in N^h.

EWD(Lu et al., 2024). Unlike prior methods that treat all tokens uniformly or apply binary inclusion based on an entropy threshold, EWD leverages a continuous, monotonically increasing function to map each token's spike entropy to a customized weight. Tokens that have higher entropy are considered more reliable indicators of watermark presence. EWD sums up the weights of the green tokens $|s|_G$ and calculate the z-score.

$$z = (|s|_G - \gamma \sum_{i=m}^{|T|-1} W_i) / \sqrt{\gamma(1-\gamma) \sum_{i=m}^{|T|-1} W_i^2} \tag{10}$$

where W_t is the weight of each token t in the text T.

IE(Gu et al., 2025). In contrast to existing entropy-aware approaches that depend on the original language model to compute token-level entropy, IE eliminates this requirement by employing

Model	Link
Qwen2.5-7B-Instruct	https://huggingface.co/Qwen/Qwen2.5-7B-Instruct
Qwen2.5-14B-Instruct	https://huggingface.co/Qwen/Qwen2.5-14B-Instruct
Llama-2-7b-chat-hf	https://huggingface.co/meta-llama/Llama-2-7b-chat-hf
Llama-2-13b-chat-hf	https://huggingface.co/meta-llama/Llama-2-13b-chat-hf
Phi-3.5-mini-instruct	https://huggingface.co/microsoft/Phi-3.5-mini-instruct
GPT-4o	https://openai.com/
sup-simcse-roberta-base	https://huggingface.co/princeton-nlp/sup-simcse-roberta-base
deberta-xlarge-mnli	https://huggingface.co/microsoft/deberta-xlarge-mnli
paraphrase-at-scale-english	http://www.cs.cmu.edu/ jwieting/paraphrase-at-scale-english.zip

Table 8: Publicly available model links used in EntroBench.

Dataset	Link
C4 RealNewsLike	https://huggingface.co/datasets/allenai/c4
WMT23	https://huggingface.co/datasets/haoranxu/WMT23-Test
GSM8K	https://huggingface.co/datasets/openai/gsm8k
MATH-500	https://huggingface.co/datasets/HuggingFaceH4/MATH-500
MMLU	https://huggingface.co/datasets/cais/mmlu
MBPP	https://huggingface.co/datasets/google-research-datasets/mbpp

Table 9: Datasets and links used in EntroBench.

a lightweight unified feature extractor together with a binary entropy tagger trained to predict whether the entropy of the next token exceeds a predefined threshold. To further enhance adaptability, IE incorporates a threshold navigator that dynamically selects per-sample entropy thresholds by optimizing the balance between watermark detectability and text naturalness, halting when the watermark ratio declines while the count of green tokens rises. IE uses the same detection method as KGW.

SynthID-Text(Dathathri et al., 2024). SynthID-Text introduces Tournament Sampling, a watermarking mechanism that embeds a detectable signal into LLM-generated text by selectively sampling tokens through a multi-round tournament. At each generation step, multiple candidates are drawn from the model's distribution and progressively filtered using a pseudorandom scoring function $g_1(\cdot, r_t), \ldots, g_m(\cdot, r_t)$ derived from the context and a secret key. The final token is chosen to align with both the original model distribution and the watermark, enabling reliable detection without altering the marginal output distribution under the non-distortionary setting. To detect whether a piece of text $x = x_1, \ldots, x_T$ is watermarked, SynthID-Text measures how highly x scores with respect to these functions. Specifically, they compute the mean g-values of the text:

$$\text{Score}(x) = \frac{1}{mT} \sum_{t=1}^{T} \sum_{\ell=1}^{m} g_\ell(x_t, r_t) \qquad (11)$$

DipMark(Wu et al., 2024b). DiPMark embeds a watermark into text generated by LLMs by dynamically partitioning the vocabulary into red and green subsets at each generation step using a secret key and context. It then applies a distribution-preserving reweighting function that increases the sampling probability of green tokens while preserving the original output distribution in expectation. This ensures that watermarked text remains statistically indistinguishable from natural model output, yet enables reliable detection given only the secret key and the generated text. DipMark introduces a watermark detection approach that differs from KGW primarily by replacing the z-test with a more accurate binomial test.

MorphMark(Wang et al., 2025b). When the cumulative probability of green-list tokens is low, applying strong watermarks yields little gain in detectability while degrading text fluency and semantic fidelity. To overcome this, MorphMark estimates the green-list probability at each token position during generation and uses this estimate to adaptively adjust the local watermark strength. By attenuating the watermark in contexts where green

tokens are unlikely and reinforcing it where they are probable, MorphMark maintains or improves detection power while minimizing unnecessary perturbations to the model's natural output distribution. They use the same detection method as KGW.

I Prompt Template

The prompts used for simulation are provided in Table 10, Table 11 and Table 12 respectively. The prompt employed to rewrite MBPP is given in Table 13.

Summarize Operation Template

Imagine you just received the following response from an AI assistant and want to quickly share its key point with others. Write a short, natural-sounding summary that captures the essential information a user would care about.
The summary must:

- Preserve only information explicitly stated in the response.
- Do not add external knowledge.
- Avoid any interpretation or opinion.

AI Response: {text}

Summary:

Table 10: Instruction used to simulate *Summarize* operation.

Bullet Points Operation Template

Imagine you just received the following response from an AI assistant and want to quickly extract the key facts to share or reference later. List all individual pieces of information as a set of concise, standalone bullet points.
The bullet points must:

- Include only facts explicitly stated in the response.
- Avoid vague pronouns like "it" or "they" without clear reference.
- Each point should capture one distinct piece of information.

AI Response: {text}

Bullet Points:

Table 11: Instruction used to simulate *Bullet Points* operation.

Code Only Operation Template

Imagine you just received the following response from an AI assistant that includes both explanatory text and a code snippet. As a user, you only care about the executable code which will be used in your project and ignore everything else.
Extract only the code portion according to the following rules:

- Return exactly the code block as it appears, without any surrounding natural language.
- If multiple code blocks exist, include all that are directly relevant to the task.
- If no valid code is present, output: No Code Found.

AI Response: {text}

Code:

Table 12: Instruction used to simulate *Code Only* operation.

Rewriting Template

You are an expert Python programming assistant. When given a natural language description of a coding problem (like those in the MBPP dataset), respond with:

- Step-by-step reasoning: Break down the problem into clear, logical steps. Explain what needs to be done, how to approach it, and any key considerations. Keep this concise but instructive.
- Provide a single, clean, correct and well-formatted Python function that solves the problem exactly as specified.
- The function should be ready to run without modification

Now solve the following problem:
{problem}
Output:

Table 13: Instruction used to rewrite MBPP.