Low-Entropy Watermark Detection via Bayesian Inference

Anonymous ACL submission

Abstract

Text watermarking, which modify tokens to embed watermark, has proven effective in detecting machine-generated texts. Yet its application to low-entropy texts like code and mathematics presents significant challenges. A fair number of tokens in these texts are hardly modifiable without changing the intended meaning, causing statistical measures to falsely indicate the absence of a watermark. Existing research addresses this issue by rely mainly on a limited number of high-entropy tokens, which are considered flexible for modification, and accurately reflecting watermarks. However, their detection accuracy remains suboptimal, as they neglect strong watermark evidences embedded in low entropy tokens modified through watermarking. To overcome this limitation, we introduce **B**ayesian **I**nference-based **W**atermark **D**etector (BIWD), which thoroughly exploit watermark information from every token, by leveraging the posterior probability of watermark's presence. We theoretically prove the optimality of our method in terms of detection accuracy, and demonstrate its superiority across various datasets, models, and watermark injection strategies. Notably, our method achieves up to 50% and 70% relative improvements in detection accuracy over the best baselines in code generation and math problem-solving tasks, respectively.

1 Introduction

Text watermarking is an effective technique for differentiating machine-generated text from human-written content that subtly injects an invisible marker, i.e. watermark, into text. It serves as a safeguard against unauthorized or malicious use of large language models, such as creating fake news (Augenstein et al., 2023) and election manipulating (Alvarez et al., 2023).

Generative watermark, which integrates watermark during LLM's generation process, generally exhibits superior detectability and robustness. A generative watermarking method injects a watermark through perturbing tokens' output distribution during text generation, and subsequently computes a score indicating its presence for detection. For instance, the KGW method (Kirchenbauer et al., 2023) partitions the model's vocabulary into green and red tokens at each generation step, and increases the output probability of green tokens, resulting in higher proportion of green tokens in a watermarked text. Subsequent detection is performed by computing the z-score of green token occurrences in the text.

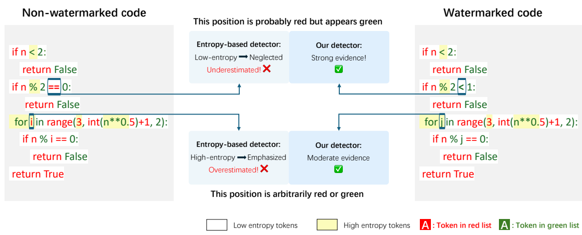


Figure 1: Comparison of entropy-based detector and ours. It demonstrates the misalignment between entropy and informativeness for watermark detection, where low-entropy positions with critical information are overlooked, and high-entropy positions are overestimated.

However, text watermarking exhibits suboptimal performance in low-entropy scenarios, such as code generation or mathematical problem solving, where a fair number of tokens are unmodifiable without compromising output quality. Statistics, such as the z-score computed for these tokens, suggest the absence of a watermark, providing contrary evidence of its presence and thus diminishing the effectiveness of the watermark detector. This limitation poses a substantial barrier to detecting malicious or unauthorized activities in software development[1], academic exams (Susnjak, 2022), and

[1] `https://www.recordedfuture.com/research/i-chatbot`

job interviews (Canagasuriam and Lukacik, 2024), raising concerns about social equality and ethical use of technology. Subsequent works (Lee et al., 2024; Lu et al., 2024) attempt to enhance detection accuracy by prioritizing high-entropy tokens, which are considered more adjustable to watermark perturbations, thus more indicative of watermark's presence. In contrast, low-entropy tokens receive little weight during detection. Nevertheless, their detection accuracy remains suboptimal, as they fail to leverage the substantial information in highly indicative low-entropy tokens, which are actually modified through watermarking.

We identified a ***misalignment*** between the entropy-based mechanism and the goal of detection: Entropy inadequately captures the modifiability of a token, nor does it account for the actual modification introduced by watermarking. To illustrate the misalignment, consider a low-entropy position where the probability is concentrated on a red token but presents a green one. It serves as a strong evidence of the watermark's presence. However, entropy-based detectors would assign a small weight to this position and neglect this evidence.

To address this misalignment, we propose **B**ayesian **I**nference-based **W**atermark **D**ector (BIWD), which quantifies the impact of watermark injection on every token. We calculate the posterior likelihood of every token altered by watermark injection with Bayesian inference, then aggregate it into a total score. A token that deviates more from its original distribution and consequently shows stronger evidence of watermark injection has a greater impact on the score. In this way, we extract more information from every token, especially highly indicative and low-entropy tokens, instead of neglect them.

We prove that BIWD *achieves optimal* true positive rate (TPR), given any false positive rate (FPR) limit. Our method is compatible with various watermark injection techniques. Extensive experiments across multiple language models, generative tasks and watermark injection schemes demonstrate the superior performance of our approach. In particular, under a 1% FPR constraint in mathematical contexts, BIWD boosts the TPR from below 60% to over 90%. Furthermore, BIWD demonstrates adaptability to general high-entropy texts and exhibits robustness against removal attacks and scenarios involving unknown prompts.

In summary, our main contributions are:

- We propose a watermark detection approach called BIWD, which significantly improves watermark detection accuracy in low-entropy scenarios.

- We provide theoretical analysis on the optimality of BIWD under any constraint on false positive rate.

- We conduct experiments with various watermarking methods in low-entropy scenarios and empirically verify BIWD's superiority.

2 Related Works

Text watermarking emerges as a promising solution to identifying machine-generated contents. They can be categorized into two types (Liu et al., 2024b): generative watermarking methods and watermarking applied to existing text.

Generative Watermarking Methods. Generative watermarking methods inject watermarks during the LLM generation phase through modifying model output logits or meticulously designed sampling process.

KGW (Kirchenbauer et al., 2023) is a seminal approach within logits-modifying type, and its proposed green-red list paradigm has been widely adopted by subsequent studies. Observing its impact on the quality of generated text, several studies (Huo et al., 2024; Chen et al., 2024) have proposed improvements. For instance, TS-watermark (Huo et al., 2024) dynamically adjusts the watermark strength for each token based on the preceding token. Some other works enhance the robustness of KGW through using fixed green list (Zhao et al., 2023) or determining the green list by semantics (Liu et al., 2024a; Liu and Bu, 2024). Additionally, some studies (Fernandez et al., 2023; Wang et al., 2024; Yoo et al., 2024) focused on injecting watermark with more information (multi-bit watermark). Research also explored the adaptability of watermarking methods to low-entropy scenarios (Lee et al., 2024; Lu et al., 2024), such as programming tasks. Logits-modifying watermarking methods typically adopt z-score based detector. To improve detectability in low-entropy scenarios, Lu et al. (2024) introduced EWD detector based on token entropy.

Sampling-based watermarking methods typically introduce pseudo-randomness accessible during detection to influence the sampling process while maintaining the token distribution on average.

For example, Kuditipudi et al. (2023) employed a long random number sequence to modify sampling and used edit distance for detection to enhance robustness. Dathathri et al. (2024) introduced tournament sampling, achieving a balance between text quality, detectability, and efficiency.

Watermarking for Existing Texts. This type of watermarking methods inject hidden features into existing texts and detect them afterwards. One way to achieve this is using end-to-end models (Abdelnabi and Fritz, 2021; Zhang et al., 2024). For instance, AWT proposed by (Abdelnabi and Fritz, 2021) employs a transformer network to inject watermarks and another transformer network to detect them. Some other methods attempt to watermark existing texts through synonym substitutions. The techniques they adopt for synonym selection include consulting an electronic dictionary (Topkara et al., 2006) and utilizing pre-trained models (Yang et al., 2023; Yoo et al., 2023). The quality of watermarked text generated by such methods is constrained by the synonym database or model used.

3 Preliminaries

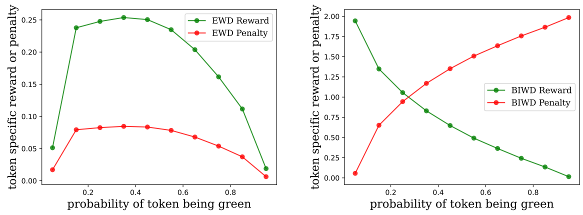
As we focus on generative watermarking due to its superior detectability and robustness, the term "watermarking" will henceforth refer to generative watermarking. In this section, we present the preliminaries of watermark injection and detection. Notations used throughout the paper are provided in Appendix A.

3.1 Watermark Injection

Watermark injection is typically achieved by perturbing the output distribution of large language model. Specifically, when generating the t-th token, the language model M computes a logits vector $\boldsymbol{l}_t$ from preceding context, which is then normalized via $\mathrm{softmax}$ to produce a distribution over candidate tokens. Watermarking methods introduce perturbations to this distribution by modifying the logits or sampling process.

In this study, we select two representative watermarking approaches as research objects: the classical KGW (Kirchenbauer et al., 2023) method and SWEET (Lee et al., 2024), the state-of-the-art method for watermarking in low-entropy scenarios.

At generation step t, KGW adds a bias δ to the logits of a subset of tokens in vocabulary $\mathcal{V}$, namely green list, denoted by $\mathcal{V}_g$. It is determined by the hash of preceding tokens and the proportion of $\mathcal{V}_g$



(a) EWD's token scoring mechanism (b) BIWD's token scoring mechanism

Figure 2: Reward and penalty with respect to total probablity of tokens being green. Statistics from 500 texts in MBPP dataset.

in $\mathcal{V}$ is γ. The remaining tokens in $\mathcal{V}$ are called "red-list", denoted by $\mathcal{V}_r$. This logits perturbation increases the probability of green tokens, resulting in a generated text with more green tokens.

The SWEET method follows the same procedure but restricts perturbations to high-entropy positions only. In SWEET method, the entropy at position t is $H_t = -\sum_{v\in\mathcal{V}} p_t^v \log p_t^v$ where p_t^v is the probability of candidate token v given by M at position t. A threshold is manually set to determine high-entropy positions.

3.2 Watermark Detection

We present a watermark detection framework encompassing detectors used by KGW and SWEET. Let $\boldsymbol{x} = \{x_0, x_1, \ldots, x_{T-1}\}$ denote the text to be detected. Within this framework, the detector assigns a score to every token x_t in $\boldsymbol{x}$. These scores are then summed and (optionally) normalized by $\mathrm{norm}(\cdot,\cdot)$ to produce a total score. A higher score indicates a greater likelihood that the text has been watermarked. Specifically, the score given to $\boldsymbol{x}$ is:

$$S(\boldsymbol{x}) = \mathrm{norm}(\sum_{t=0}^{T-1} s(\boldsymbol{x}, t), \boldsymbol{x})$$

where s is the score given to token at each position. In practice, a threshold is defined and any text scored higher than it is identified as watermarked.

4 Method

In this section, we first illustrate our motivation by examining the limitations of current watermark detection methods. Subsequently, we present our detection approach.

4.1 Motivation

Although EWD and SWEET detectors achieve better accuracy in low-entropy scenarios, their improvement stems from attaching more importance to those high entropy tokens, which are more adjustable to watermark perturbation. However, the goal of watermark detection is to measure the influence of watermark perturbation. We explain this misalignment from a token scoring perspective.

The EWD detector assigns a positive score as a reward, if $x_t \in \mathcal{V}_g$, and a negative score, whose absolute value is a penalty if $x_t \in \mathcal{V}_r$. As for token-level reward or penalty, we have the following ***intuition***: if a position was likely to output a red token according to its original distribution without watermark but a green token appears here. It indicates the influence of watermark and should receive a significant reward. Inversely, a position with high probability of green tokens but appears a red one strongly suggests that watermark is absent, thus should receive a significant penalty.

However, EWD does not fully exploit these evidences that are critical for determining the presence of the watermark. Figure 2a illustrates the relationship between the reward and penalty assigned by the EWD detector ($\gamma = 0.25$) and the probability of token being green. It can be observed that both reward and penalty exhibit a bell-shaped curve trend. The reward is lower when the probability of a token being green is relatively low, and the penalty is lower when this probability is relatively high, which contradicts our previous intuition. Other entropy-based methods like SWEET have the same misalignment problem.

To further analyze the impact of this misalignment on token scores, we define a watermark information score (WIS). For the t-th token x_t in a text $\boldsymbol{x}$, its WIS is formulated as follows:

$$\mathrm{WIS}(\boldsymbol{x}, t) = \mathbf{1}_{\mathcal{V}_g}(x_t) \sum_{v \in \mathcal{V}_r} p_t^v + \mathbf{1}_{\mathcal{V}_r}(x_t) \sum_{v \in \mathcal{V}_g} p_t^v$$

This score quantifies the amount of information a token carries about the watermark. Specifically, for a green token, its WIS equals the total probability of red tokens. A higher value indicates a greater likelihood of the watermark's presence, as the position would otherwise likely contain a red token. For a red token, its WIS equals the total probability of green tokens. A higher value suggests a lower likelihood of the watermark's presence.

We plotted a scatter diagram of the entropy and WIS of tokens in watermarked codes. In Figure 3, the green tokens within the black rectangle exhibit both high WIS and low entropy. This indicates that entropy-based detectors like EWD, assign these tokens relatively low importance, thereby impairing

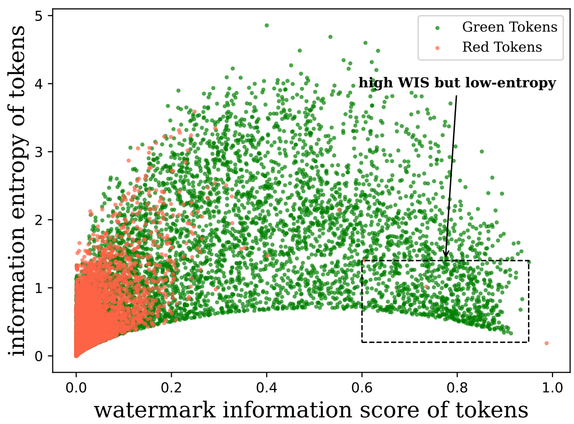


Figure 3: Watermark information score (WIS) and entropy of tokens in watermarked codes. Statistics from 500 watermarked codes generated using MBPP dataset and StarCoder2-7B model.

detection performance. This observation further supports our hypothesis regarding the misalignment in entropy-based detectors.

To address this issue, we devise a detection method based on Bayesian inference that aims at measuring watermark influence. We treat the target text $\boldsymbol{x}$, as the value of a random variable, and infer the likelihood of the watermark's presence from it.

4.2 Bayesian Inference Detection

Let w denote the presence of a watermark (0/1 binary) and P_w represent the distribution of w. Suppose the distribution of all natural language (watermarked and non-watermarked text) is P and the distribution given by model M is P_M. Given a text $\boldsymbol{x}$ to be detected and its corresponding prompt $\boldsymbol{a}$, we derive the following using Bayes' theorem:

$$P_w(w = 1|\boldsymbol{a}; \boldsymbol{x}) = \frac{P(\boldsymbol{x}|\boldsymbol{a}; w = 1)P_w(w = 1)}{\sum_{i=0,1} P(\boldsymbol{x}|\boldsymbol{a}; w)P_w(w = i)} \quad (1)$$

Dividing both the numerator and the denominator of the above equation by $P(\boldsymbol{x}|\boldsymbol{a}; w = 0)$, we obtain (1) equals:

$$\frac{A(\boldsymbol{x}, \boldsymbol{a})P_w(w = 1)}{P_w(w = 0) + A(\boldsymbol{x}, \boldsymbol{a})P_w(w = 1)} \quad (2)$$

where

$$A(\boldsymbol{x}, \boldsymbol{a}) = \frac{P(\boldsymbol{x}|\boldsymbol{a}; w = 1)}{P(\boldsymbol{x}|\boldsymbol{a}; w = 0)} \quad (3)$$

$P_w(w = 0)$ and $P_w(w = 1)$ are constant across different $\boldsymbol{x}$ as they are the probability of text being watermarked in the whole corpus. Therefore, the posterior likelihood (2) increases monotonically with $A(\boldsymbol{x}, \boldsymbol{a})$, So we can use $A(\boldsymbol{x}, \boldsymbol{a})$ to represent the likelihood of the watermark's presence. The numerator in (3) is the output probablity of model M with watermark injection, which is identical to $P_M(\boldsymbol{x}|\boldsymbol{a}; w = 1)$. The denominator is the conditional probability of human written texts. As large language models are pretrained on large amount of human written text, this probability can be approximated by $P_M(\boldsymbol{x}|\boldsymbol{a}; w = 0)$. Consequently, our focus reduces to computing:

$$\hat{A}(\boldsymbol{x}, \boldsymbol{a}) = \frac{P_M(\boldsymbol{x}|\boldsymbol{a}; w = 1)}{P_M(\boldsymbol{x}|\boldsymbol{a}; w = 0)} \tag{4}$$

Applying the chain-rule $P_M(\boldsymbol{x}|\boldsymbol{a}; w) = \prod P_M(x_t|\boldsymbol{a}, x_{:t}; w)$ ($x_{:0}$ denotes empty sequence) and taking the log-transform, we have:

$$\hat{A}(\boldsymbol{x}, \boldsymbol{a}) = \sum_{t=0}^{T-1} \big(\log P_M(x_t|\boldsymbol{a}, x_{:t}; w = 1) - \log P_M(x_t|\boldsymbol{a}, x_{:t}; w = 0) \big) \tag{5}$$

For the KGW and SWEET (assume t is a high-entropy position) injection method, given the original logit of token v at position t, l_t^v, the perturbed logit is $\tilde{l}_t^v = l_t^v + \delta \cdot \mathbf{1}_{\mathcal{V}_g}(v)$. For the model output distribution, we have $P_M(\cdot|\boldsymbol{a}, x_{:t}; w = 1) = \mathrm{softmax}(\tilde{\boldsymbol{l}}_t)$ and $P_M(\cdot|\boldsymbol{a}, x_{:t}; w = 0) = \mathrm{softmax}(\boldsymbol{l}_t)$. By substituting these equations of P_M and $\tilde{\boldsymbol{l}}_t$ into (5), we obtain:

$$\hat{A}(\boldsymbol{x}, \boldsymbol{a}) = \sum_{t=0}^{T-1} \big(\delta \cdot \mathbf{1}_{\mathcal{V}_g}(x_t) - \log \frac{e^{\delta} G_t + R_t}{G_t + R_t}\big) \tag{6}$$

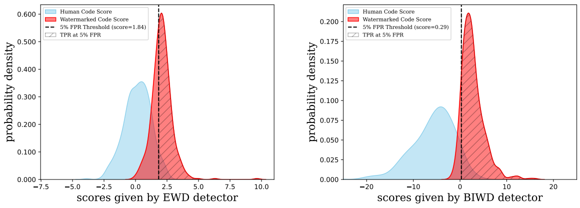
where $G_t = \sum_{v \in \mathcal{V}_g} e^{l_t^v}$ and $R_t = \sum_{v \in \mathcal{V}_r} e^{l_t^v}$ are the unnormalized total probability of green, red tokens, respectively.

Similar to the discussion on EWD, we can interpret the score of green token as a reward, and the absolute value of score given to a red token as a penalty. Figure 2b demonstrates that the reward given by BIWD decreases monotonically, while the penalty increases monotonically with the probability of token being green, consistent with the intuitive analysis in 4.1.

We also present the text-level score distributions of EWD and BIWD. Figures 4a and 4b show that BIWD exhibits stronger discriminative ability.

4.3 Theoretical Analysis

We theoretically prove that BIWD has optimal true positive rate within any false positive rate constraint, given a prompt $\boldsymbol{a}$ and model M. In order to formalize this conclusion, we need the following notations and definitions.



(a) EWD detector (b) BIWD detector

Figure 4: Distribution of scores given by EWD and BIWD. We use MBPP dataset, StarCoder2 model and KGW watermark injection method for this experiment. The size of the dashed area represents the TPR at 5% FPR. It can be observed that BIWD significantly outperforms EWD on this metric.

- Let Ω denote the set of all possible output token sequence $\boldsymbol{x}$. Conditional probabilities $P_M(\cdot|\boldsymbol{a}; w = 0), P_M(\cdot|\boldsymbol{a}; w = 1)$ given by M and its watermarked counterpart both endow Ω with a probability measure.
- For any watermark detection method ϕ which functions as a binary classifier on Ω, its effect is equivalent to a split of Ω:

  $$\Omega = \mathcal{K}_\phi \cup \mathcal{K}_\phi^c$$

  where $\mathcal{K}_\phi$ means the subset of sequences identified as watermarked by ϕ. Therefore, $\mathcal{K}_\phi$ fully expresses the effect of detection ϕ on Ω. For our method, ϕ=BIWD, and the detection effect is expressed by $\mathcal{K}_{\text{BIWD}}$.

Definition 4.1. For a detection method ϕ, its false positive rate α_ϕ on $\Omega(P_M(\cdot|\boldsymbol{a}))$ is defined as:

$$\int_\Omega \mathbf{1}_{\mathcal{K}_\phi}(\boldsymbol{x}) P_M(\boldsymbol{x}|\boldsymbol{a}; w = 0)\ \mathrm{d}\boldsymbol{x}$$

Definition 4.2. For a detection method ϕ, its true positive rate β_ϕ on $\Omega(P_M(\cdot|\boldsymbol{a}; w = 1))$ is defined as:

$$\int_\Omega \mathbf{1}_{\mathcal{K}_\phi}(\boldsymbol{x}) P_M(\boldsymbol{x}|\boldsymbol{a}; w = 1)\ \mathrm{d}\boldsymbol{x}$$

Definition 4.3. For a threshold $\eta > 0$, the BIWD detector with this threshold, denoted as $\text{BIWD}(\eta)$, classify any sequence $\boldsymbol{x}$ with score $\hat{A}(\boldsymbol{x}, \boldsymbol{a}) > \eta$ as watermarked. Therefore, its positive sample subset is:

$$\mathcal{K}_{\text{BIWD}(\eta)} = \{\boldsymbol{x} \in \Omega | \hat{A}(\boldsymbol{x}, \boldsymbol{a}) > \eta\}$$

Theorem 4.1. *For any limit on false positive rate $\sigma > 0$, if there exists a threshold $\eta > 0$ such that $\alpha_{\text{BIWD}(\eta)} = \sigma$, then for any detection method ϕ s.t. $\alpha_\phi \leq \sigma$, its true positive rate is no greater than that of* $\text{BIWD}(\eta)$

$$\beta_\phi \leq \beta_{\text{BIWD}(\eta)}$$

A proof is available in appendix B.

5 Experiments

We conduct experiments across different low-entropy datasets, language models and watermark injection methods.

Tasks and Datasets. We select two tasks: code generation and math problem solving. For code generation, we use HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) datasets which contain python programming problems and reference answers following the setups in Lee et al. (2024) and Lu et al. (2024). For math problem solving, we select 500 samples from GSM8K (Cobbe et al., 2021), which contains mathematical problems in English. In all experiments, we use texts longer than 15 tokens for detection.

Models. For code generation, we employ StarCoder2-7B (Lozhkov et al., 2024), a model specializes in programming, following prior research (Lee et al., 2024; Lu et al., 2024). For mathematical problem solving, we use DeepSeekMath-7B-Instruct (Shao et al., 2024), which is optimized for mathematical reasoning. To assess the generalizability of our method, we also utilize the versatile Llama2-7B model (Touvron et al., 2023), applying it to both code generation and math problem-solving tasks.

Watermark injection Methods. We utilize two injection methods, KGW and SWEET. KGW acts as a classic baseline in this field, while SWEET is specifically optimized for low-entropy scenarios. Detailed configurations are provided in Appendix C. To differentiate watermark injection schemes from their detectors, we denote KGW's injection and detection methods as $\mathrm{KGW_I}$ and $\mathrm{KGW_D}$, and SWEET's as $\mathrm{SWEET_I}$ and $\mathrm{SWEET_D}$.

Baselines and Metrics. To detect watermarks injected by $\mathrm{KGW_I}$ and $\mathrm{SWEET_I}$, we employ their corresponding detectors as baselines. We also apply EWD as a detector for both, as it is optimized for low-entropy scenarios. Additionally, we employ the detection method proposed by DIPMark (Wu et al., 2024) as another baseline, as it does not rely on the classical z-score framework and is adaptable to other injection methods. This detector is denoted as $\mathrm{DIP_D}$. Following Lee et al. (2024) and Lu et al. (2024), we use true positive rates at 1% and 5% false positive rates(TPR@1%FPR, TPR@5%FPR), along with the corresponding F1-score, as our evaluation metrics. True positive rate (TPR) and false positive rate (FPR) represent the proportion of watermarked text successfully detected and the proportion of human-written text mistakenly classified as AI-generated, respectively. Best F1 scores are also reported to show the overall performance of detectors. All metrics are measured in a single run.

6 Results

Table 1 and 2 (all metrics are shown in percentage) demonstrate that our method exhibits substantial advantages in detection accuracy for code and mathematical text compared to baselines, and it is applicable to different models and watermark injection methods.

For code generation task with StarCoder2 model and $\mathrm{KGW_I}$ injection method, the average improvements of BIWD over the best baseline, on two datasets are 14% in TPR@1%FPR and 24% in TPR@%5FPR. For the same task and model with $\mathrm{SWEET_I}$ injection method, the avergae improvements of the two metrics are 23% and 27%, respectively. For mathematical problem solving task with DeepSeek model, the improvements of the two metrics are 38%, 19% while using $\mathrm{KGW_I}$ injection method and 35%, 24% while using $\mathrm{SWEET_I}$ injection method. Notably, for this model, our method improves the TPR@1%FPR from below 60% to over 90%.

For Llama2 model, our detection method also outperforms all baselines. Specifically, it achieves an improvement of 34% in TPR@1%FPR on MBPP dataset while using $\mathrm{SWEET_I}$ injection method. Although our method shows relatively small advantages in certain settings, the baselines have already achieved high accuracy in these settings, leaving limited room for improvement.

7 Analysis

7.1 Performance without Original Prompts

In applications like detecting cheating in job interviews, prompts are usually accessible, specifically the question text. However, for code copyright protection, the original prompt is often unavailable. Therefore, we also conduct experiments with a general prompt following previous studies (Lee et al., 2024; Lu et al., 2024). The details about this setting are in Appendix C.2. Table 3 and Table 4 show that our method still outperforms all baselines. For the code generation task and $\mathrm{KGW_I}$ watermark injection method, BIWD achieves average improvements of 8.49% and 16.28% in TPR@1%FPR and

Model	Injection	Detection	HumanEval					MBPP				
			1%FPR		5%FPR		Best	1%FPR		5%FPR		Best
			TPR	F1	TPR	F1	F1	TPR	F1	TPR	F1	F1
StarCoder2	KGW_I	KGW_D	10.92	19.55	29.41	44.03	73.87	8.02	14.74	24.30	37.58	74.05
		DIP_D	17.65	29.79	29.41	44.03	73.22	19.09	31.83	35.57	50.62	73.81
		EWD	44.54	61.27	60.50	73.47	84.13	35.14	51.67	66.38	77.47	88.28
		BIWD	**63.87**	**77.55**	**83.19**	**88.79**	**93.98**	**43.60**	**60.36**	**91.32**	**93.04**	**94.25**
	SWEET_I	SWEET_D	32.23	48.45	49.59	64.52	80.74	37.09	53.77	54.66	68.57	84.84
		DIP_D	9.09	16.54	32.23	47.27	82.96	35.36	51.91	67.68	78.39	86.71
		EWD	38.84	55.62	54.55	68.39	83.97	32.75	49.03	62.69	74.77	88.21
		BIWD	**58.68**	**73.58**	**83.47**	**88.60**	**92.37**	**59.00**	**73.81**	**88.29**	**91.36**	**93.97**
Llama2	KGW_I	KGW_D	58.78	73.73	74.32	83.02	84.93	28.43	43.99	61.09	73.63	84.17
		DIP_D	21.62	35.36	62.16	74.49	80.23	34.68	51.19	61.29	73.79	83.01
		EWD	96.62	97.95	98.65	97.01	97.95	50.81	67.02	79.64	86.34	91.07
		BIWD	**97.97**	**98.64**	**100**	**97.69**	**99.00**	**66.33**	**79.37**	**98.59**	**96.93**	**97.11**
	SWEET_I	SWEET_D	84.67	91.37	90.67	92.83	94.77	49.09	65.50	77.17	84.79	88.62
		DIP_D	77.37	86.89	86.67	90.59	93.46	60.81	75.25	88.08	91.31	90.25
		EWD	85.33	91.76	90.67	92.83	94.53	50.10	66.40	84.04	88.98	90.84
		BIWD	**92.00**	**95.50**	**96.00**	**95.68**	**96.75**	**83.64**	**90.69**	**97.98**	**96.61**	**97.19**

Table 1: Detection accuracy on HumanEval, MBPP datasets using StarCoder2-7B, Llama2-7B models and KGW_I, SWEET_I injection methods.

Model	Injection	Detection	GSM8K				
			1%FPR		5%FPR		Best
			TPR	F1	TPR	F1	F1
DeepSeek	KGW_I	KGW_D	8.00	14.68	24.20	37.46	67.73
		DIP_D	1.80	3.47	7.40	13.17	67.71
		EWD	52.60	68.49	78.20	85.37	87.86
		BIWD	**91.00**	**94.79**	**97.60**	**96.35**	**96.83**
	SWEET_I	SWEET_D	57.60	72.64	75.40	83.59	88.21
		DIP_D	31.20	47.20	69.40	79.86	87.23
		EWD	40.40	57.14	70.40	80.27	85.47
		BIWD	**92.80**	**95.77**	**99.80**	**97.46**	**97.65**
Llama2	KGW_I	KGW_D	73.35	84.23	81.16	87.28	88.35
		DIP_D	59.32	74.09	71.34	81.00	82.80
		EWD	90.38	94.55	94.79	94.98	95.93
		BIWD	**90.98**	**94.88**	**95.99**	**95.61**	**95.96**
	SWEET_I	SWEET_D	84.77	91.36	93.39	94.24	94.27
		DIP_D	88.38	93.43	94.79	95.36	94.83
		EWD	89.58	94.11	95.39	95.30	95.94
		BIWD	**95.99**	**97.56**	**97.60**	**96.44**	**97.57**

Table 2: Detection accuracy on GSM8K dataset using DeepSeek-math-7B-instruct, Llama2-7B models and KGW_I, SWEET_I watermark injection methods.

TPR@5%FPR over the best baseline, respectively. For the same task and SWEET_I watermark injection method, BIWD demonstrates average improvements of 6.61% and 16.45% in TPR@1%FPR and TPR@5%FPR, respectively. For the task of solving mathematical problems and KGW_I watermark injection method, BIWD achieves improvements of 15.68% and 11.81% for the two metrics, respectively. For this task with SWEET_I watermark injection method, BIWD results in improvements of 4.48% and 6.31% for the same metrics. We also find that as text length increases, the absence of the original prompts has a diminishing impact on detection accuracy.

7.2 Adaptability to High-Entropy Texts

To evaluate the applicability of our detection method in general high-entropy scenario, we conduct experiments using 500 samples from C4 (Raffel et al., 2023) English news dataset and the Llama2-7B model. Detailed settings are in Appendix D. Table 5 shows that our detection method outperforms the baselines.

7.3 Robustness against Removal Attack

Since malicious users tend to modify the content generated by LLMs to escape detection, we need to assess the robustness of our detector against removal attacks. Following Lee et al. (2024), we use variable name substitution as our attack method. We replace 50% of the variables in each function with random strings of 2 to 5 characters. The watermark injection method used is $\text{KGW}_\mathbf{I}$ with $\delta = 2$ and $\gamma = 0.5$. All methods show a significant decrease in accuracy after the attack, but our method remains the best overall. Exploring how to enhance the robustness of low-entropy text watermarking could be a direction for further research.

Injection	Detection	HumanEval					MBPP				
		1%FPR		5%FPR		Best	1%FPR		5%FPR		Best
		TPR	F1	TPR	F1	F1	TPR	F1	TPR	F1	F1
KGW_I	KGW_D	9.65	17.46	28.95	43.42	73.57	8.03	14.74	24.30	37.58	74.05
	DIP_D	12.28	21.71	28.07	42.38	73.57	19.09	31.83	35.57	50.62	73.81
	EWD	24.56	39.16	42.11	57.49	80.65	17.14	29.04	36.88	51.99	77.77
	BIWD	**29.82**	**45.64**	**44.74**	**60.00**	**87.10**	**28.85**	**44.48**	**66.81**	**77.78**	**86.76**
SWEET_I	SWEET_D	19.83	32.86	37.93	53.33	77.65	9.33	16.93	32.54	47.32	77.32
	DIP_D	6.03	11.29	33.62	48.75	77.78	20.17	33.33	35.57	51.01	76.25
	EWD	23.28	37.50	40.52	55.95	79.70	14.75	25.52	36.88	51.99	78.25
	BIWD	**25.86**	**40.82**	**50.00**	**64.80**	**84.43**	**25.38**	**40.21**	**60.30**	**72.97**	**86.33**

Table 3: Detection accuracy on HumanEval, MBPP datasets using StarCoder2-7B model and KGW_I, SWEET_I injection methods without original prompts.

Injection	Detection	GSM8K				
		1%FPR		5%FPR		Best
		TPR	F1	TPR	F1	F1
KGW_I	KGW_D	7.13	13.21	23.83	37.03	67.65
	DIP_D	1.8	3.47	7.40	13.17	67.71
	EWD	11.61	20.65	40.53	55.74	78.32
	BIWD	**27.29**	**42.61**	**52.34**	**66.58**	**78.80**
SWEET_I	SWEET_D	8.35	15.30	28.92	43.23	73.74
	DIP_D	7.20	13.31	22.40	35.16	73.63
	EWD	7.13	13.21	30.96	45.58	**75.05**
	BIWD	**12.83**	**22.58**	**37.27**	**52.44**	74.14

Table 4: Detection accuracy on GSM8K dataset using DeepSeek-math-7B-instruct model and KGW_I, SWEET_I injection methods without original prompts.

8 Conclusion

In this study, we identify the misalignment issue in entropy-based watermark detection methods and propose a detection approach that addresses this problem. Our detector utilizes Bayesian inference to fully leverage the distribution given by language model. We provide theoretical analysis on its optimality and empirically verify its superiority in low-entropy scenario. Notably, compared to the best baseline, our method achieves up to 1.5 times detection accuracy on the code datasets and up to 1.7 times detection accuracy on the mathematics dataset. Additionally, experiments demonstrate that our watermark detection method outperforms the baselines in terms of accuracy in high-entropy scenarios, detection accuracy without prompts, and robustness against removal attacks.

Injection	Detection	1%FPR		5%FPR		Best
		TPR	F1	TPR	F1	F1
KGW_I	KGW_D	99.59	99.39	99.59	97.42	99.39
	DIP_D	98.99	99.09	99.59	97.42	98.59
	EWD	99.59	99.39	**100**	**97.62**	99.70
	BIWD	**100**	**99.60**	**100**	**97.62**	**100**
SWEET_I	SWEET_D	99.80	99.50	99.80	97.53	99.70
	DIP_D	**100**	**99.60**	**100**	**97.63**	99.90
	EWD	99.80	99.50	**100**	**97.63**	99.70
	BIWD	**100**	**99.60**	**100**	**97.63**	**100**

Table 5: Detection performance on C4 dataset using Llama2-7B model and KGW_I, SWEET_I watermark injection methods.

Injection	1%FPR		5%FPR		Best
	TPR	F1	TPR	F1	F1
KGW_D	5.51	10.35	13.62	22.98	70.60
DIP_D	10.43	18.75	15.94	26.38	68.84
EWD	**15.94**	**27.30**	29.86	44.30	76.38
BIWD	14.78	25.56	**46.96**	**61.83**	**79.83**

Table 6: Detection performance on MBPP dataset after variable substitution attack. The model and watermark injection method used are StarCoder2-7B and KGW_I

9 Limitations

Our detection method has two main limitations. First, the generation tasks and datasets tested are limited. We employ two code datasets and one mathematics dataset for evaluation. we plan to test our method on a broader range of low-entropy tasks with more diverse datasets. Second, we only test two watermark injection methods in this study. Our approach is applicable to a variety of generative watermark injection methods based on distribution perturbation. We plan to implement and evaluate our method with other watermark injection techniques in future work.

References

Sahar Abdelnabi and Mario Fritz. 2021. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. *Preprint*, arXiv:2009.03015.

R. Michael Alvarez, Frederick Eberhardt, and Mitchell Linegar. 2023. Generative ai and the future of elections.

Isabelle Augenstein, Timothy Baldwin, Meeyoung Cha, Tanmoy Chakraborty, Giovanni Luca Ciampaglia, David Corney, Renee DiResta, Emilio Ferrara, Scott Hale, Alon Halevy, Eduard Hovy, Heng Ji, Filippo Menczer, Ruben Miguez, Preslav Nakov, Dietram Scheufele, Shivam Sharma, and Giovanni Zagni. 2023. Factuality challenges in the era of large language models. *Preprint*, arXiv:2310.05189.

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

Damian Canagasuriam and Eden-Raye Lukacik. 2024. Chatgpt, can you take my job interview? examining artificial intelligence cheating in the asynchronous video interview. *International Journal of Selection and Assessment*, 33(1):e12491.

Liang Chen, Yatao Bian, Yang Deng, Deng Cai, Shuaiyi Li, Peilin Zhao, and Kam-Fai Wong. 2024. WatME: Towards lossless watermarking through lexical redundancy. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9166–9180, Bangkok, Thailand. Association for Computational Linguistics.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *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, Jamie Hayes, Nidhi Vyas, Majd Al Merey, Jonah Brown-Cohen, Rudy Bunel, Borja Balle, Taylan Cemgil, Zahra Ahmed, Kitty Stacpoole, Ilia Shumailov, Ciprian Baetu, Sven Gowal, Demis Hassabis, and Pushmeet Kohli. 2024. Scalable watermarking for identifying large language model outputs. *Nature*, 634:818–823.

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

Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Wen tau Yih, Luke Zettlemoyer, and Mike Lewis. 2023. Incoder: A generative model for code infilling and synthesis. *Preprint*, arXiv:2204.05999.

Mingjia Huo, Sai Ashish Somayajula, Youwei Liang, Ruisi Zhang, Farinaz Koushanfar, and Pengtao Xie. 2024. Token-specific watermarking with enhanced detectability and semantic coherence for large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 20746–20767. PMLR.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. *The Fortieth International Conference on Machine Learning*.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2023. Robust distortion-free watermarks for language models. *Trans. Mach. Learn. Res.*, 2024.

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.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. 2024a. A semantic invariant robust watermark for large language models. *Preprint*, arXiv:2310.06356.

Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip S. Yu. 2024b. A survey of text watermarking in the era of large language models. *Preprint*, arXiv:2312.07913.

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

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, Tianyang Liu, Max Tian, Denis Kocetkov, Arthur Zucker, Younes Belkada, Zijian Wang, Qian Liu, Dmitry Abulkhanov, Indraneil Paul, Zhuang Li, Wen-Ding Li, Megan Risdal, Jia Li, Jian Zhu, Terry Yue Zhuo, Evgenii Zheltonozhskii, Nii Osae Osae Dade, Wenhao Yu, Lucas Krauß, Naman Jain, Yixuan Su, Xuanli He, Manan Dey, Edoardo Abati, Yekun Chai, Niklas Muennighoff, Xiangru Tang, Muhtasham Oblokulov, Christopher Akiki, Marc Marone, Chenghao Mou, Mayank Mishra, Alex Gu, Binyuan Hui, Tri Dao, Armel Zebaze, Olivier Dehaene, Nicolas Patry, Canwen Xu, Julian McAuley, Han Hu, Torsten Scholak, Sebastien Paquet, Jennifer Robinson, Carolyn Jane Anderson, Nicolas Chapados, Mostofa Patwary, Nima Tajbakhsh, Yacine Jernite, Carlos Muñoz Ferrandis, Lingming Zhang, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2024. Starcoder 2 and the stack v2: The next generation. *Preprint*, arXiv:2402.19173.

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.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Teo Susnjak. 2022. Chatgpt: The end of online exam integrity? *Preprint*, arXiv:2212.09292.

Umut Topkara, Mercan Topkara, and Mikhail J. Atallah. 2006. The hiding virtues of ambiguity: quantifiably resilient watermarking of natural language text through synonym substitutions. In *Proceedings of the 8th Workshop on Multimedia and Security*, MM and Sec '06, page 164–174, New York, NY, USA. Association for Computing Machinery.

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

Lean Wang, Wenkai Yang, Deli Chen, Hao Zhou, Yankai Lin, Fandong Meng, Jie Zhou, and Xu Sun. 2024. Towards codable watermarking for injecting multi-bits information to llms. *Preprint*, arXiv:2307.15992.

Yihan Wu, Zhengmian Hu, Junfeng Guo, Hongyang Zhang, and Heng Huang. 2024. A resilient and accessible distribution-preserving watermark for large language models. *Preprint*, arXiv:2310.07710.

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

KiYoon Yoo, Wonhyuk Ahn, Jiho Jang, and Nojun Kwak. 2023. Robust multi-bit natural language watermarking through invariant features. *Preprint*, arXiv:2305.01904.

KiYoon Yoo, Wonhyuk Ahn, and Nojun Kwak. 2024. Advancing beyond identification: Multi-bit watermark for large language models. 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 4031–4055, Mexico City, Mexico. Association for Computational Linguistics.

Ruisi Zhang, Shehzeen Samarah Hussain, Paarth Neekhara, and Farinaz Koushanfar. 2024. Remarkllm: A robust and efficient watermarking framework for generative large language models. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 1813–1830.

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

A Notations

Symbol	Description
M	Language model used for generation.
$\mathcal{V}$	Vocabulary of M.
$\mathcal{V}_g$	Green list.
$\mathcal{V}_r$	Red list.
v	Token in V.
w	Whether watermark exists. (0/1)
P_M	Distribution given by M.
P	Distribution of all natural language.
P_w	Distribution of w.
$\boldsymbol{a}$	Prompt used for generation.
$\boldsymbol{x}$	Text generated by M.
x_t	The t-th token in $\boldsymbol{x}$.
t	Generation step.
$\boldsymbol{l}_t$	Model logits at the t-th step.
l_t^v	Model logit of token v at the t-th step.
$\boldsymbol{p}_t$	Distribution given by M at step t.
p_t^v	Probability of v given by M at step t.
H_t	Information entropy at step t.
$S(\boldsymbol{x})$	Watermark detection score of $\boldsymbol{x}$.
$s(\boldsymbol{x}, t)$	Watermark detection score given to x_t.
norm	Normalizer for watermark detection.
T	Length of $\boldsymbol{x}$.
γ	Green-list ratio in KGW and SWEET.
δ	δ in KGW and SWEET.

Table 7: Notations used throughout the paper.

B Proof of Theorem

From definition 4.3, we have the following inequality (simply consider $\boldsymbol{x} \in \mathcal{K}_{\mathrm{BIWD}(\eta)}$ and $\boldsymbol{x} \notin \mathcal{K}_{\mathrm{BIWD}(\eta)}$ respectively):

$$[\mathbf{1}_{\mathcal{K}_{\mathrm{BIWD}(\eta)}}(\boldsymbol{x}) - \mathbf{1}_{\mathcal{K}_\phi}(\boldsymbol{x})] \cdot [P_M(\boldsymbol{x}|\boldsymbol{a}, w=1) - \eta P_M(\boldsymbol{x}|\boldsymbol{a}, w=0)] \geq 0$$

Then we integrate the above inequality over Ω and simplify the integral with definition 4.1 and 4.2. The result is as follows:

$$\beta_{\mathrm{BIWD}(\eta)} - \beta_\phi \geq \eta(\alpha_{\mathrm{BIWD}(\eta)} - \alpha_\phi)$$

According to our assumptions in 4.1, $\alpha_{\mathrm{BIWD}(\eta)} \geq \alpha_\phi$. Therefore, we have $\beta_{\mathrm{BIWD}(\eta)} \geq \beta_\phi$. $\square$

C Detailed Experiment Configurations

C.1 Generation Settings

The sampling strategy we use in main experiments is top-p sampling with top-p=0.95. We set temperature=0.2 during generation. For the HumanEval dataset, we feed the original prompts in it to language models. For the MBPP data set, we use three-shot prompts following Fried et al. (2023). For the GSM8K dataset, we add a chain-of-thought instruction to each question to construct our prompt. The instruction we use is "Please reason step by step, and put your final answer within \boxed{}.". Regarding the watermark injecting methods, we set $\gamma = 0.5$ and $\delta = 2$ for the KGW and SWEET methods. As for the entropy-threshold in SWEET method, we set it to 0.65 for experiments with Llama model and 0.6 in all other cases.

C.2 General Prompts

For Python function completion tasks in HumanEval dataset, we used the general prompt "def solution(*args):\n '''Generate a solution'''\n". For programming tasks in MBPP dataset, we used the general prompt "Write a python function to implement a specific requirement.\n". For math problem solving tasks in GSM8K dataset, the general prompt we used is "Solve a math problem, please think step by step.\n".

D High-Entropy Dataset

The watermark injection schemes we used are KGW and SWEET. For both injection schemes, we set $\gamma = 0.5$ and $\delta = 2$. For SWEET injection, we set the entropy threshold at 0.65. The temperature we used for generation is 0.7, and the sampling strategy adopted is top-p sampling with top-p=0.95. The generated texts are truncated to 200 tokens, and the minimum length for detection is 15 tokens.