An Entropy-based Text Watermarking Detection Method

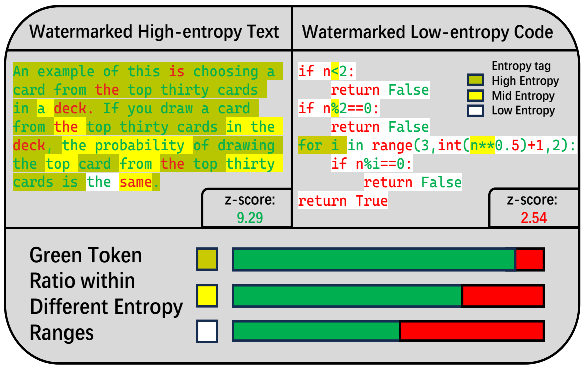
Yijian LU[1], Aiwei Liu[2] , Dianzhi Yu[1] , Jingjing Li[1] , Irwin King[1]*
[1]The Chinese University of Hong Kong
[2]Tsinghua University
luyijian@link.cuhk.edu.hk, liuaw20@mails.tsinghua.edu.cn
dianzhi.yu@link.cuhk.edu.hk, lijj@link.cuhk.edu.hk, king@cse.cuhk.edu.hk

Abstract

Text watermarking algorithms for large language models (LLMs) can effectively identify machine-generated texts by embedding and detecting hidden features in the text. Although the current text watermarking algorithms perform well in most high-entropy scenarios, its performance in low-entropy scenarios still needs to be improved. In this work, we opine that the influence of token entropy should be fully considered in the watermark detection process, *i.e.*, the weight of each token during watermark detection should be customized according to its entropy, rather than setting the weights of all tokens to the same value as in previous methods. Specifically, we propose **E**ntropy-based Text **W**atermarking **D**etection (**EWD**) that gives higher-entropy tokens higher influence weights during watermark detection, so as to better reflect the degree of watermarking. Furthermore, the proposed detection process is training-free and fully automated. From the experiments, we demonstrate that our EWD can achieve better detection performance in low-entropy scenarios, and our method is also general and can be applied to texts with different entropy distributions. Our code and data is available[1]. Additionally, our algorithm could be accessed through MarkLLM (Pan et al., 2024)[2].

1 Introduction

The rapid advancements in large language models (LLMs) have enabled them to generate high-quality outputs indistinguishable from humans, and also achieve better performance on various real-world generation tasks. However, this gives rise to the potential risk of misuse. Specifically, LLM-generated assignments, including essays and codes, pose a threat to academic integrity. Misusing LLMs to generate fake news can also lead to negative societal consequences (Megías et al., 2021). Therefore, algorithms capable of effectively identifying machine-generated contents have become essential.


Watermarked High-entropy Text
Watermarked Low-entropy Code
Entropy tag
High Entropy
Mid Entropy
Low Entropy
z-score:
9.29
z-score:
2.54
Green Token Ratio within Different Entropy Ranges

```
An example of this is choosing a
card from the top thirty cards
in a deck. If you draw a card
from the top thirty cards in the
deck, the probability of drawing
the top card from the top thirty
cards is the same.
```

```
if n<2:
    return False
if n%2==0:
    return False
for i in range(3,int(n**0.5)+1,2):
    if n%i==0:
        return False
return True
```

Figure 1: This figure shows that compared with watermarked texts with mostly high-entropy tokens, watermarked codes with mostly low-entropy tokens see significantly less green tokens, resulting in a small detection z-score. Furthermore, on the bottom of the figure, we demonstrate that the green token ratio in tokens decreases as their entropy decreases.

Text watermarking algorithms can effectively alleviate the problem of LLM misuse by embedding and detecting hidden features in LLM-generated contents. For example, the watermarking algorithm proposed by Kirchenbauer et al. (2023a) (KGW) divides the LLM's model vocabulary into two lists, green and red, and increases the logits value of tokens in the green list by a positive constant (watermark strength). The modified logits will bias the LLM to sample most tokens from the green list. As a result, the watermarked text containing a dominant number of green tokens will be recognized by the watermark detector while human texts will not. However, the successful watermark generation is largely tied to the token entropy during text generation. Under the watermark algorithm, the probability of sampling a token from the green list is proportional to its entropy, meaning that low-

*Corresponding Author.
[1]https://github.com/luyijian3/EWD
[2]https://github.com/THU-BPM/MarkLLM

entropy tokens would produce far less green tokens than high-entropy tokens, as illustrated by Figure 1. In that case, green tokens will occupy only half of a watermarked low-entropy text, resulting in a smaller detection score and being mistakenly labeled as human-written. Furthermore, to ensure the text quality of output contents, the watermark strength in the generator cannot be excessively enlarged to forcefully modify the logits and sample more tokens from the green list (Tu et al., 2023).

Consequently, a crucial problem is how to improve the watermark detector so that low-entropy watermarked texts with a limited number of green tokens can be successfully identified. Lee et al. (2023) propose a selective watermark detector named SWEET to better reflect the watermark level in low-entropy texts. Specifically, they only detect tokens that exceed a certain entropy threshold, thus excluding tokens that are difficult to be affected by the watermark generator. However, this method itself has three weaknesses. First, their method requires a manually-set entropy threshold by analyzing samples of code written by humans. Second, their method also does not consider the distribution of token entropy, that is, their method treats all tokens that exceed the threshold as the same, while in fact the entropy of these tokens may vastly vary. Third, the performance gain using the SWEET detector is limited and there is still significant room for further advancement.

To better address the low-entropy watermarking detection problem than previous methods, we introduce an **E**ntropy-based Text **W**atermark **D**etection (**EWD**) algorithm to fully consider the role of entropy in the detection of a given text. Specifically, we propose that the influence of a token during detection should be proportional to its entropy. Furthermore, in order to fully reflect this positive relation between token entropy and detection influence, we utilize a monotonically-increasing and continuous function to generate influence weights from token entropy. By creating a gap between the weights of high entropy tokens and low entropy tokens, we ensure that if the watermark generation algorithm can hardly affect a low-entropy token, the state of that token will make minimal impact to the detection result. Overall, our method has three major benefits. First, the method saves the trouble of altering the watermark generation process. Second, compared to SWEET, our method is more efficient as the acquisition of token weights is an independent and training-free process, avoiding the trouble of a human dataset or any manual operations. Third, the proposed detection scheme can be applied to machine-generated texts of all entropy levels, not solely low-entropy texts.

From the experiments, we demonstrate that the detection accuracy of EWD in low-entropy scenarios surpasses existing baselines. Furthermore, EWD exhibits versatility in detecting texts of high entropy levels, as well as robustness against the back-translation watermark-removal attack.

In summary, the contributions of our work are summarized as follows:

- We propose an entropy-based text watermarking detection method called **EWD**.
- We provide a theoretical analysis of the detection accuracy in the low-entropy setting.
- We show that EWD not only improves the detection accuracy under the low-entropy scenario, but also provides similar performance as other methods in high-entropy texts.

2 Related Work

In this section, we aim to review existing works on text watermarking algorithms. Overall, text watermarking can be divided into two different categories: watermarking for existing texts and watermarking for LLMs Liu et al. (2023).

Text Watermarking for Existing Texts. Watermark message can be embedded into existing text by performing a series of modifications, such as lexical substitutions (Munyer and Zhong, 2023; Topkara et al., 2006; Yang et al., 2022; Yoo et al., 2023a; Yang et al., 2023). Those methods typically replace words with their alternatives to place watermark. For example, Topkara et al. (2006) used WordNet(Fellbaum, 1998) as his dictionary while Munyer and Zhong (2023) utilized a pretrained Word2Vec model to select the alternative words. To ensure that the repalced words would cause minimal changes to the text semantics, Yang et al. (2022) proposed a BERT-based infill model to generate replacement with regard to the sentence meaning. To improve the robustness of the algorithm against possible corruptions, Yoo et al. (2023a) fine-tuned the infill model to generate consistent word distribution given several corrupted texts of the same origin. However, the level of modification to an existing text is often limited to protect text quality in the watermarked output. Hence, watermarking for existing text is less favored due to its limited capacity and effectiveness.

Text Watermarking for LLMs. Watermark embedding can also be conducted during the LLM generation phase, either by modifying the token sampling or model logits. Christ et al. (2023) introduced a technique that incorporates watermark message by predefining a random number sequence for sampling the tokens. To enhance resistance to text modifications, Kuditipudi et al. (2023) utilized the Levenshtein distance in matching the text and numbers. For the latter, the most representative one that modifies the model logits is proposed by Kirchenbauer et al. (2023a) based on previous tokens during text generation. There are a number of works proposed on top of this algorithm with enhanced performance in payload (Yoo et al., 2023b; Wang et al., 2023), robustness (Zhao et al., 2023; Liu et al., 2024b; Kirchenbauer et al., 2023b; Ren et al., 2023; He et al., 2024) and unforgeability (Liu et al., 2024a; Fairoze et al., 2023). For example, Yoo et al. (2023b) extends the method to a multi-bit setting.Wang et al. (2023) proposed to utilize a proxy LLM for multi-bit message embedding. To increase robustness against editing attacks, Zhao et al. (2023) proposed to set a universal green list during the generation process. Liu et al. (2024b) proposed a different approach called SIR which utilizes the semantic meaning of previous generated text to determine the green lists. Inspired by Liu et al. (2024b), He et al. (2024) firstly proposed the Crosslingual Watermark Removal Attack (CWRA), then offered X-SIR as a defense method against CWRA. Also, to enhance the unforgeability of text watermark, Liu et al. (2024a) introduced a neural network for text generation and detection. In this work, we focus only on the detection of low-entropy texts. Lee et al. (2023) designed a selective watermark generation and detection approach for low-entropy code generation. However, it requires a human code dataset to help manually determine the entropy threshold. To sum up, there lacks an effective method to improve the low-entropy detection problem.

3 Preliminaries

3.1 Text Generation Process of LLMs

Here we introduce the necessary concepts used in this paper. A LLM, M takes a prompt as input and outputs subsequent tokens as the corresponding response. Specifically, we denote the initial prompt as x^{prompt}. Then, at the l-th step, the input to the LLM is the x^{prompt} and the already-generated tokens $T_{:(l-1)}$. The LLM would generate a distribution logits $P_{LLM}(x^{prompt}, T_{:(l-1)})$, consisting of $P_{LLM}(v|x^{prompt}, T_{:(l-1)})$ over each token v of its model vocabulary based on the joint input.

3.2 Text Watermarking

The general watermarking scheme used in this paper is called KGW, which is to modify the logits generation process of an original LLM M to acquire the watermarked LLM, denoted as $\tilde{M}$. The modification on the original logits by adding a small value (the watermark strength δ) to the logits of green-list tokens will result in LLM's preference on those tokens in the output text. Thus such modification could likely produce a green token.

The detection algorithm is to calculate the watermark level using the following equation,

$$z = (|s|_G - \gamma|T|)/\sqrt{\gamma(1-\gamma)|T|}, \qquad (1)$$

where $|s|_G$ is the number of green tokens in text T and γ is the green-list ratio in the model vocabulary. The algorithm would output $Detect(T) = 1$ if the value is greater than a given threshold, and $Detect(T) = 0$ otherwise.

3.3 Token Entropy and Low-entropy Scenario

The entropy of a token is measured by the degree of spread-out of the logits generated by the LLM during the sampling process of that token. In this paper, we follow the spike entropy from KGW,

$$SE(k) = \sum_v \frac{p_v}{1 + \tau p_v}, \qquad (2)$$

where p_v is the logits value of each token v in the vocabulary when sampling the token k, and τ is a scalar. The entropy value reaches the minimal value if the logits concentrates on a single token and the maximal value given a uniform distribution.

The watermark generation process is more effective on a high-entropy token where the logits distribution is nearly uniform, so that the modification could produce a green-list token easily. On the other hand, a low-entropy token is almost deterministic and hard to be affected. Therefore, the green-list ratio for a low-entropy token is much lower, resulting in fewer green tokens in the output. Therefore, low-entropy watermarked texts pose threats on the detection accuracy as they are easily identified as human-written.

4 Proposed Method

In this section, we will first explain the motivation by analysing the watermark generation effect on low-entropy tokens, then provide a detailed explanation of the proposed EWD.

4.1 Motivation

Entropy plays an important role in the watermark generation process, yet its role in the detection process has been underestimated. In the KGW generator, a positive constant is added to the logits value of all green-list tokens. The distribution level of the logits, represented by the entropy, matters significantly during this process. When the logits resembles a uniform distribution, this logits modification would very likely produce a green-list token. However, when the logits concentrates on one single token, this watermark embedding process would hardly affect the token sampling and produce a green token with a much lower probability close to the green list ratio γ. Since entropy determines the watermark generation process, then it should also affect the watermark detection. However, in the KGW detector, all tokens are of the same influence, no matter how their entropy varies.

4.2 Entropy-based Text Watermarking Detection

Our proposed EWD aims to fully consider the role of entropy in the watermark detection process. Specifically, we firstly propose a positive influence-entropy relation, then utilize entropy for detection weight customization.

The influence of a token during detection should be proportional to its entropy. From previous sections, we explain why a high-entropy token is easier to be affected by the watermark generator, as well as the situation for a low-entropy token. Since a low-entropy token is hard to be affected, then whether it is sampled from the green list or not should have a limited influence on the overall watermark level. Similarly, when an easy-to-affect (high-entropy) token is not green, it should contribute more to showing that the text T is not watermarked. Therefore, we introduce a positive influence-entropy relation to determine how much a token t can influence the detection result:

$$I(Detect(T) = \mathbf{1}(t \in G)|t) \propto SE(t), \quad (3)$$

where T is the given text and G is the green list.

Table 1: Features of detection weights by different methods and how they can reflect the proposed positive influence-entropy relation.

Methods	Detection Weight	Reflection of Eq. (3)
KGW	Constant	None
SWEET	Binary	Partial
EWD	Customized	Full

Algorithm 1 Entropy-based Watermark Detection

```
Input: a language model $M$, input text $T$, detection key
    $k$, window size $m$, detection threshold $\tau$, green list
    ratio $\gamma$
Compute token entropy based on model logits,
    $SE = SpikeEntropy[M(T)] = [SE_0, SE_1....]$
Compute token weights based on token entropy,
    $W = ComputeWeight[SE] = [W_0, W_1....]$
for $l = m, m+1....$ do
    Use the detection key $k$ and previous $m$ tokens
      $T_{l-m:l-1}$ to find the green list $G$
    If current token $T_l$ is inside the green list $G$, then
      add its weight $W_l$ to $|s|_G$
Compute the detection score $z'$ by Eq. (5)
if $z' > \tau$ then return 1, $i.e.$, "Input text is watermarked"
else return 0, $i.e.$, "Input text is not watermarked"
```

The detection weight of a token should be customized based on its entropy to fully reflect the positive influence-entropy relation. The influence index in Eq. (3) is represented by the detection weight of each token. However, the detection weight assigning of each token in SWEET fails to reflect the positive influence-entropy relation. This is because that inside the two groups of above-threshold tokens and below-threshold tokens, the weights of tokens with different entropy are identical. Therefore, to fully reflect the influence-entropy relation, we utilize a monotonically-increasing and continuous function f to customize weights from token entropy. The weight $W(t)$ of a token t can be illustrated as follows:

$$W(t) = f(SE(t) - C_0), \quad (4)$$

where C_0 is the minimal value of the spike entropy to normalize the entropy input before computing the weight. Using the entropy-based weight customization, even a sight increase in the token entropy will result in a increased detection weight. Table 1 summarizes the weight features of different methods and their reflection of the proposed influence-entropy relation.

Pseudocode of our proposed detection method is provided in Algorithm 1. We first acquire the text entropy by computing the model logits of each token. The next step is to determine the influence weight of each token, the output of a designed function *ComputeWeight* with their entropy values as inputs. After obtaining the entropy-based weights for each token, we then apply the standard detection procedure by Kirchenbauer et al. (2023a) to determine the group of green tokens, *i.e.*, computing the green lists using the detection key and previously generated tokens. Finally, we sum up the weights of the green tokens $|s|_G$ and calculate the z-score by

$$z' = (|s|_G - \gamma \sum_{i=m}^{|T|-1} W_i) / \sqrt{\gamma(1-\gamma) \sum_{i=m}^{|T|-1} {W_i}^2}, \tag{5}$$

given $|s|_G$ and weight W_t of each detected token t in text T. The detector would give a positive result if the z-score is greater than the given threshold. It is noticably that the proposed algorithm is a fully automated process that eliminates the need for manual operations from the SWEET method.

5 Theoretical Analysis

In this section, we theoretically analyze the Type-I and Type-II error of our proposed detection method. Specifically, we compare EWD with two previous detection methods, KGW and SWEET.

To do so, we adopt the same steps as in KGW (Kirchenbauer et al., 2023a). Each token in both watermarked and non-watermarked texts can either be green or red, so we can model it as a binomial distribution. While Zhao et al. (2023) set the window size in the watermark generator as zero (one universal green list for all tokens) and exhibits major token dependence, *i.e.*, the same tokens in the text are of the same color, in our case, where a non-zero window size is adopted, the token dependence is significantly weakened. For example, the same token with different prefix would exhibit different sampling result.

Therefore, to simplify and facilitate the process of theoretical analysis, we make the reasonable assumption that the sampling of each token is independent. Hence, the sum of green tokens in the text can be approximated with a normal distribution $\mathcal{N}(\mu, \sigma^2)$ with different sets of mean μ and variance σ^2 for different detection schemes. Finally, we calculate and compare the probability of the sum surpassing or falling behind the detection threshold $|\tilde{s}|_G$, which is the $|s|_G$ value in each detection function with a fixed z.

5.1 Type-I Error

Type-I error measures the likelihood a human-written text is incorrectly identified as being generated by a watermark algorithm.

In a human-written text T, each token is independent of the watermark algorithm. Therefore, its probability to be included in the green list is γ. Therefore, KGW's sum of green tokens is approximated by $\mathcal{N}(\gamma|T|, \gamma(1-\gamma)|T|)$. For SWEET that only considers high-entropy tokens, such distribution is modelled as $\mathcal{N}(\gamma|\tilde{T}|, \gamma(1-\gamma)|\tilde{T}|)$, where $\tilde{T}$ is the text excluding low-entropy tokens. Our method EWD assign weights to tokens based on entropy, leading to a weighted binomial distribution. We could also approximate it to a normal distribution with mean $\mu = \gamma \sum_{i=m}^{|T|-1} W_i$ and variance $\sigma^2 = \gamma(1-\gamma) \sum_{i=m}^{|T|-1} W_i^2$.

The theoretical Type-I errors for all detection methods are the same. When the detection threshold is set to 2, the probabilities for $|s|_G$ under each method to surpass its detection threshold $|\tilde{s}|_G$ is identically 2.28%.

5.2 Type-II Error

Type-II error measures the likelihood a machine-generated text is incorrectly identified as human-written. For analysis, we assume that a sequence generated by the watermark algorithm has a length of $|T| = 200$ tokens and the watermark algorithm has hyper-parameters $\gamma = 0.5$ and $\delta = 2$.

Originally, KGW samples over 500 watermark-generated news scripts and reports an average spike entropy of 0.807. In this section, we focus specifically on cases with a much lower average spike entropy. Based on our data, we approximate the distribution of spike entropy in low-entropy watermarked texts with a power law distribution $f(x) = ax^{a-1}$, where the parameters are $a = 0.106$, $loc = 0.566$, $scale = 0.426$. Under this distribution, the average spike entropy is 0.608. The following probability from KGW shows a lower bound of the probability of a token k being sampled from the green list:

$$\mathbb{P}[k \in G] \geq \frac{\gamma\alpha}{1 + (\alpha - 1)\gamma} SE(k), \tag{6}$$

where $\alpha = exp(\delta)$. We would utilize this probability later in the calculation of μ and σ^2.

KGW. Based on Eq. (6), the statistical μ and upper bound of σ^2 can be obtained as $\sum_k \mathbb{P}[k \in G]$ and $\sum_k \mathbb{P}[k \in G] \cdot (1 - \mathbb{P}[k \in G])$. When the

Table 2: This table summarizes our calculated theoretical Type-I and Type-II error using different methods in low-entropy detection scenarios. The data in the table is presented in percentages (%). Lower values indicates better detection accuracy.

Methods	Type-I Error	Type-II Error
KGW	2.28	84.1
SWEET	2.28	41.7
EWD	**2.28**	**33.4**

average entropy SE^* given by the simulated power law distribution is 0.608, we approximate $|s|_G$ by $\mathcal{N}(107.10, 49.70)$. If we set a threshold of z = 2 (corresponding to $|\tilde{s}|_G$ = 114.14) for detection, the false negative rate exceeds 84.1%. This means that KGW is not good at detecting the watermarked texts in cases of low entropy.

SWEET. Following SWEET, we set the entropy threshold at 0.695 and exclude all tokens below this threshold during detection. The average spike entropy SE^* and remaining text length $|\widetilde{T}|$ then change to 0.82 and 24, respectively. Using the equations from KGW, we can get the approximation as $\mathcal{N}(17.48, 4.84)$. If we set a threshold of z = 2 (corresponding to $|\tilde{s}|_G$ = 17.02) for detection, the false negative rate exceeds 41.7%.

EWD. EWD assigns higher influence weights to tokens with high spike entropy. Note that in experiments, we use several weight functions, and here, for analysis purposes, we use a linear weight function for each token k: $W(k) = SE(k) - C_0$, where C_0 is a constant. Based on Eq. (6), the statistical μ and σ^2 can be obtained as $\sum_k \mathbb{P}[k \in G] \cdot W(k)$ and $\sum_k W(k)^2 \mathbb{P}[k \in G] \cdot (1 - \mathbb{P}[k \in G])$. Therefore, we can approximate the $|s|_G$ detected by EWD with $\mathcal{N}(5.79, 0.31)$. If we set a threshold of z' = 2 (corresponding to $|\tilde{s}|_G$ = 5.55) for detection, the false negative rate exceeds 33.4% (details in Appendix A).

Table 2 summarizes our calculated theoretical Type-I and Type-II error using different methods in low-entropy detection scenarios. Overall, our method achieves the lowest Type-II error among all methods while maintaining the same Type-I error.

6 Experiments

In this section, we first introduce the tasks and datasets used for watermarked text generation. Then, we compare the performance of our EWD against several baselines on those datasets. Next, we analyze the role of our proposed entropy-based weight and study the performance of EWD using different weight computing functions. Finally, we evaluate the detection performance without the original prompt, detection speed and robustness of our EWD.

6.1 Experiment Settings

Tasks and Datasets. There are various text generation tasks in the field of Natural Language Processing, and here we aim to generate watermarked text of both low-entropy and high-entropy distributions. Since our focus is on comparing different detection methods, we utilize the same KGW watermark generation algorithms in all the following tasks and datasets.

For low-entropy scenarios, we target at the task of code generation by adopting two datasets following Lee et al. (2023), HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). The two datasets contain test cases of python programming problems and corresponding reference answers, which are utilized as human-written samples. Code generation is conducted by StarCoder (Li et al., 2023). During evaluation, the length of both samples are restricted to be at least 15 tokens.

The first common high-entropy scenario is the news report generation. Our input prompts to the pre-trained OPT-1.3B (Zhang et al., 2022) are from the C4 (Raffel et al., 2020) dataset. Besides, we include another high-entropy data-to-text generation task by utilizing Wu et al. (2021)'s version of the Rotowire dataset (Wiseman et al., 2017). The dataset consists of NBA game records, presented as tables, and human-written descriptions. We fine-tune a T5-small (Raffel et al., 2023) model for text generation and input the table information to the fine-tuned model. We generate 200 samples of length 200 ± 5 tokens for each high-entropy tasks, using binomial sampling and beam search.

Baselines and Evaluation Metrics. Two detection methods are selected as baselines. The first is KGW detector. The second is SWEET's detector, which excludes low-entropy tokens during detection. Note that SWEET proposes both selective watermark generation and detection, while here we only adopt the detection part. The selection of the entropy threshold follows the instructions in SWEET. For evaluation, we set the false positive rate (FPR) under 5%, following Zhao et al. (2023)'s instruction. We report the true positive rate (TPR) and F1 score for each method.

Hyper-parameters. For the KGW generator, γ and δ is set to 0.5 and 2, respectively. Binomial

Table 3: Main results on different datasets using KGW, SWEET and our EWD for detection.

Code Detection (Low-entropy Scenario)										
Methods	HumanEval					MBPP				
	1%FPR		5%FPR		Best	1%FPR		5%FPR		Best
	TPR	F1	TPR	F1	F1	TPR	F1	TPR	F1	F1
KGW	0.331	0.494	0.414	0.567	0.772	0.115	0.205	0.348	0.498	0.744
SWEET	0.455	0.622	0.667	0.778	0.842	0.409	0.577	0.650	0.765	0.865
EWD	**0.466**	**0.633**	**0.692**	**0.797**	**0.859**	**0.567**	**0.720**	**0.744**	**0.830**	**0.878**

Rotowire (High-entropy Scenario)										
Methods	Sampling					Beam-search				
	1%FPR		5%FPR		Best	1%FPR		5%FPR		Best
	TPR	F1	TPR	F1	F1	TPR	F1	TPR	F1	F1
KGW	1.000	0.995	1.000	0.976	1.000	1.000	0.995	1.000	0.976	1.000
SWEET	1.000	0.995	1.000	0.976	1.000	1.000	0.995	1.000	0.976	1.000
EWD	**1.000**	**0.995**	**1.000**	**0.976**	**1.000**	**1.000**	**0.995**	**1.000**	**0.976**	**1.000**

C4 (High-entropy Scenario)										
Methods	Sampling					Beam-search				
	1%FPR		5%FPR		Best	1%FPR		5%FPR		Best
	TPR	F1	TPR	F1	F1	TPR	F1	TPR	F1	F1
KGW	0.985	0.990	1.000	**0.980**	0.992	0.995	0.993	1.000	**0.980**	0.997
SWEET	1.000	0.997	1.000	0.978	1.000	0.995	0.993	1.000	0.976	0.997
EWD	**1.000**	**0.997**	**1.000**	0.978	**1.000**	**0.995**	**0.993**	**1.000**	0.976	**0.997**

sampling sets the sampling temperature to a common value of 0.7. Beam-search sampling sets the number of beams to 8. To avoid excessive repetition of text, "no_repeated_ngrams" is set to 16.

6.2 Main Results

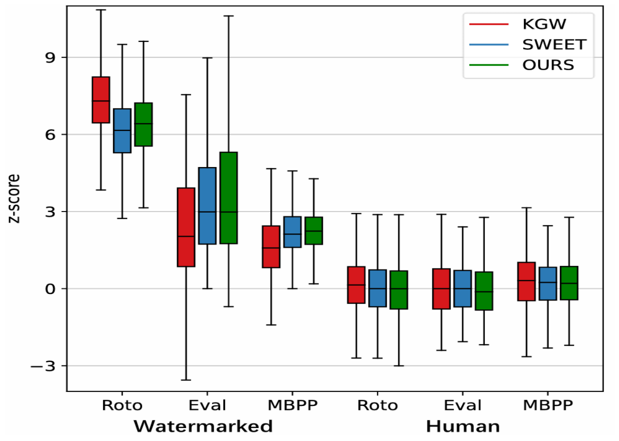
Before detection, we measure the generation quality of all generated texts. Specifically, we are able to achieve 31.0% and 30.4% in pass@1(Chen et al., 2021) for the HumanEval and MBPP dataset, respectively, comparable to Lee et al. (2023)'s. For the Rotowire and C4 dataset, our generated output can achieve 7.24 and 3.25 in text perplexity (PPL) using OPT-6.7B (Zhang et al., 2022).

Table 3 shows that for the low-entropy scenario, watermarked code detection, our detector would achieve the best detection accuracy against other baselines. Specifically, for HumanEval and MBPP, respectively, our method could achieve a 2.5% and 9.4% improvement than SWEET in TPR while maintaining a low-than 5% FPR.

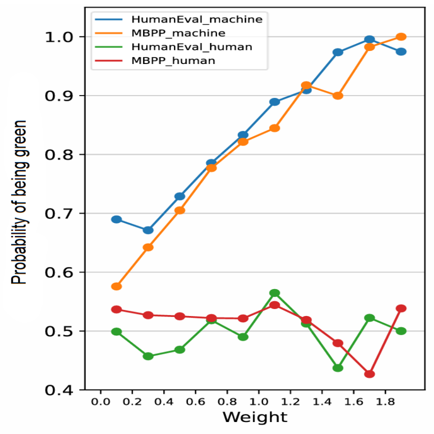
For the high-entropy scenarios using the Rotowire and C4 datasets, the detection performance of our method is overall very similar to other baselines, sometimes even better than some baselines.

6.3 Empirical Analysis

Impact of Entropy-based Weights. In Figure 2, we provide a few visualizations to show the reason why EWD surpasses other baselines in detection performance. Figure 2(a) shows the z-scores of both watermarked and human texts in different datasets, detected by different methods. Our EWD method would result in overall higher z-scores for watermarked texts, and slightly lower z-scores for human texts. The z-score increase is more significant in low-entropy code generation tasks than in the Rotowire dataset. By enlarging the gap between the z-scores of watermarked and human texts, our detector could better distinguish the watermarked texts. In Figure 2(b), we plot the relationship of a token's entropy-based weight and probability of that token being sampled from the green list. In human-written texts, there does not exist a relationship between the probability and token weights, explaining why the z-scores of human text remain low. Meanwhile, in watermarked texts, the probability is proportional to the token weight, which means the tokens assigned with a large weight are very likely to be sampled from the green list and contribute to a larger z-score. This proves that our entropy-based weights can help more accurately reflect the watermark level in low-entropy texts.

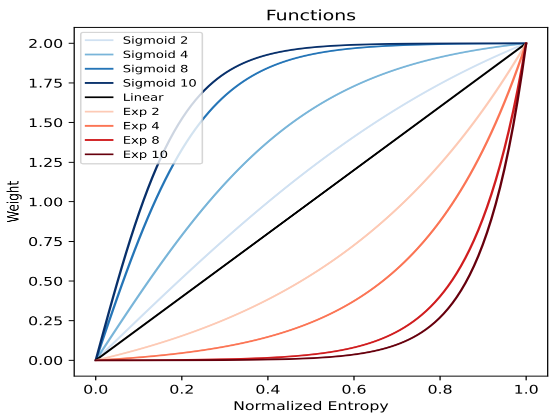


(a)

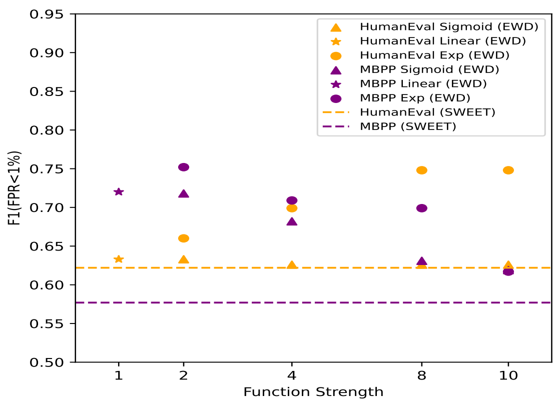


(b)

Figure 2: Subfigure (a) shows the z-scores of watermarked and human texts in the Rotowire, HumanEval and MBPP datasets, respectively, each being detected with 3 different methods. Subfigure (b) shows the relationship between token weights and the probability of being green in both watermarked and human texts.



(a)



(b)

Figure 3: We utilize two additional weight functions other than Linear and measured their performance in the code detection datasets. Subfigure (a) is a visualization of all studied functions, with normalized spike entropy as input. Subfigure (b) shows each function's detection F1 under 1% FPR with comparison to the SWEET baseline. Each data point can correspond to the illustrated function on the left figure by looking at its shape and x-axis value.

Performance with Different Weight Functions. In the experiments, we utilize a linear function to gain the entropy-based weights for each token. We also study two more monotonic increasing functions, the Sigmoid and the Exponential function. As shown in Figure 3(a), we further change the function strength to 4 different values in each function, which represents how far the function deviates from being linear, for example, "Sigmoid 10" deviates from linear more than "Sigmoid 8". The abovementioned functions are applied in Eq. (4) as f. Specifically, each function's gradient is different. For our custom Sigmoid functions, the gradient starts off high and gradually decreases, meaning that tokens with relatively low entropy will have significant differences in weights, while tokens with relatively high entropy will exhibit smaller differences. Similarly, for our custom Exponential functions, the gradient starts off low and gradually increases, meaning that tokens with relatively low entropy will have smaller differences in weight, while tokens with relatively high entropy will exhibit larger differences.

Figure 3(b) demonstrates that the two listed functions with different strength could surpass the SWEET baseline in terms of detection F1. This reflects the versatility of our method, as it is applicable to more functions.

Performance without Original Prompts. The entropy calculation during detection of our EWD

Table 4: Detection performance of watermarked code using the general prompt instead of original prompts.

Detection w/ General Prompt		
Methods	F1 under 5%FPR	Best F1
KGW	0.498	0.741
SWEET	0.580	0.762
EWD	**0.600**	**0.764**

Table 5: This table shows the average time taken to generate 200-token texts using KGW generator, as well as the average time taken for detection using different methods, measured in seconds. The experiment is conducted on a NVIDIA RTX A6000 GPU.

Method	Generation	Detection
KGW		**0.0391**
SWEET	9.489	0.0830
EWD		0.0831

and SWEET is conducted with the original prompts. There exists many real-world scenarios where the original prompts are available, *e.g.*, some programming assignment/exams, in such cases, our detection method would produce the most accurate entropy and detection performance. However, it is possible that the original prompts are not available during detection. Therefore, we design experiments on MBPP dataset where we use a general prompt "`def solution(*args):\n"'Generate a solution\n"'`" to replace the original prompt during entropy calculation. The detection result can be found in Table 4. Even without the original prompt, our EWD could achieve better detection performance against other baselines. However, the detection performance by both SWEET and EWD has dropped compared to using original prompts.

Detection Speed. Compared to the KGW detection, both SWEET and our proposed EWD would demand extra time to compute token entropy. Specifically, our detector would also compute weights given the token entropy before calculating the final z-score. According to Table 5, the detection time per text is 0.0391, 0.0830 and 0.0831 seconds for KGW, SWEET and our EWD, respectively. In spite of being double the value of KGW, our detector remains highly efficient, and the difference with SWEET is virtually insignificant.

Performance against the Back-translation Attack. Watermarked texts usually would be edited before being detected, which would remove part of the watermark and cause detection performance drop. Here we also evaluate the baseline methods and our EWD under this setting by utilizing back-translation as the attack to remove watermark.

Table 6: Detection performance of back-translated watermarked texts using different detection methods.

Detection w/ Back-translation				
Methods	1%FPR		5%FPR	
	TPR	F1	TPR	F1
KGW	0.894	0.940	**0.950**	**0.952**
SWEET	0.890	0.938	0.930	0.940
EWD	**0.900**	**0.943**	0.942	0.947

Specifically, we firstly generate watermarked texts using the C4 dataset, then translate them from English to French, and later back to English again for detection. In Table 6, our EWD would achieve the best detection accuracy under 1% FPR and similar results with KGW under 5% FPR.

7 Conclusion

In this work, we propose a text watermark detection method called EWD that fully considers the influence of token entropy. We first establish the positive influence-entropy relation in the detection process, then utilize entropy for weight customization to fully reflect the proposed relation. We provide theoretical analysis on the detection properties of different methods in low-entropy scenarios. Experiments validate the detection performance of our method in the low-entropy code generation task with a number of weight functions. Furthermore, in high-entropy scenarios, our method would achieve similar performance as previous works in detection accuracy, detection speed and robustness.

Limitations

Our method mainly includes two limitations. The first one is that the low-entropy datasets tested are limited. We utilize two code generation datasets for evaluation. In the future, we aim to include more low-entropy tasks and datasets.

Secondly, our method should be theoretically effective for various types of watermark methods, such as token sampling-based methods. In the future, we aim to implement our method on more types of text watermarking frameworks.

Acknowledgement

We sincerely appreciate the insightful comments from all reviewers. Their help has greatly contributed to the improvement of this work.

The work described in this paper was partially supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (CUHK 14222922, RGC GRF 2151185).

References

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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

Jaiden Fairoze, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Mingyuan Wang. 2023. Publicly detectable watermarking for language models. Cryptology ePrint Archive, Paper 2023/1661. `https://eprint.iacr.org/2023/1661`.

Christiane Fellbaum. 1998. *WordNet: An electronic lexical database*. MIT press.

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

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023a. A watermark for large language models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. 2023b. On the reliability of watermarks for large language models. *arXiv preprint arXiv:2306.04634*.

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

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

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

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 *The Twelfth International Conference on Learning Representations*.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. 2024b. A semantic invariant robust watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Lijie Wen, Irwin King, and Philip S Yu. 2023. A survey of text watermarking in the era of large language models. *arXiv preprint arXiv:2312.07913*.

David Megías, Minoru Kuribayashi, Andrea Rosales, and Wojciech Mazurczyk. 2021. Dissimilar: Towards fake news detection using information hiding, signal processing and machine learning. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*, pages 1–9.

Travis Munyer and Xin Zhong. 2023. Deeptextmark: Deep learning based text watermarking for detection of large language model generated text. *arXiv preprint arXiv:2305.05773*.

Leyi Pan, Aiwei Liu, Zhiwei He, Zitian Gao, Xuandong Zhao, Yijian Lu, Binglin Zhou, Shuliang Liu, Xuming Hu, Lijie Wen, et al. 2024. Markllm: An open-source toolkit for llm watermarking. *arXiv preprint arXiv:2405.10051*.

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. *The Journal of Machine Learning Research*, 21(1):5485–5551.

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.

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

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*, pages 164–174.

Shangqing Tu, Yuliang Sun, Yushi Bai, Jifan Yu, Lei Hou, and Juanzi Li. 2023. Waterbench: Towards holistic evaluation of watermarks for large language models. *arXiv preprint arXiv:2311.07138*.

Lean Wang, Wenkai Yang, Deli Chen, Hao Zhou, Yankai Lin, Fandong Meng, Jie Zhou, and Xu Sun. 2023. Towards codable text watermarking for large language models. *arXiv preprint arXiv:2307.15992*.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. *arXiv preprint arXiv:1707.08052*.

Xueqing Wu, Jiacheng Zhang, and Hang Li. 2021. Text-to-table: A new way of information extraction. *arXiv preprint arXiv:2109.02707*.

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

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

KiYoon Yoo, Wonhyuk Ahn, Jiho Jang, and Nojun Kwak. 2023a. Robust multi-bit natural language watermarking through invariant features. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2092–2115.

KiYoon Yoo, Wonhyuk Ahn, and Nojun Kwak. 2023b. Advancing beyond identification: Multi-bit watermark for language models. *arXiv preprint arXiv:2308.00221*.

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

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

A Proof of Type-II Error for EWD

Here we demonstrate how we obtain the theoretical Type-II error for EWD. First of all, we attempt to calculate the theoretical μ and σ^2 based on Eq. (6). The theoretical μ is the weighted sum of green tokens in the watermarked text, using $|s|_{G_k}$ to represent the weighted number of green tokens from each token k,thus we can obtain:

$$\begin{aligned}\mu &= \sum_k |s|_{G_k}, && (7)\\ &= \sum_k \mathbb{P}[k \in G] \cdot W(k), && (8)\\ &= T\mathbb{E}\frac{\gamma\alpha SE}{1+(\alpha-1)\gamma}(SE - C_0), && (9)\\ &= T\frac{\gamma\alpha}{1+(\alpha-1)\gamma}(\mathbb{E}[SE^2] - C_0\mathbb{E}[SE]). && (10)\end{aligned}$$

Given the simulated power law of spike entropy, we can obtain $E[SE^2]$ and $E[SE]$ as 0.377 and 0.608, respectively, producing μ equal to 5.79. And for σ^2, we first obtain the variance for each token k:

$$\begin{aligned}{\sigma_k}^2 &= \mathbb{E}[(|s|_{G_k} - \mathbb{E}[|s|_{G_k}])^2], && (11)\\ &= \mathbb{E}[|s|_{G_k}^2] - \mathbb{E}[|s|_{G_k}]^2, && (12)\\ &= \mathbb{P}[k \in G] \cdot W(k)^2 - \mathbb{P}[k \in G]^2 \cdot W(k)^2, && (13)\\ &= W(k)^2 \cdot (\mathbb{P}[k \in G])(1 - \mathbb{P}[k \in G]). && (14)\end{aligned}$$

Therefore, if we represent $\frac{\gamma\alpha}{1+(\alpha-1)\gamma}$ by C_1, the variance of the whole text can be formulated as:

$$\begin{aligned}\sigma^2 &= \sum_k W(k)^2 \cdot (\mathbb{P}[k \in G])(1 - \mathbb{P}[k \in G]), && (15)\\ &= T\mathbb{E}[W(k)^2 C_1 SE(1 - C_1 SE)], && (16)\\ &= TC_1\mathbb{E}[(SE - C_0)^2 SE(1 - C_1 SE)], && (17)\\ &= TC_1(-C_1\mathbb{E}[SE^4] + (2C_0C_1 + 1)\mathbb{E}[SE^3] \\ &\quad - (2C_0 + {C_0}^2 C_1)\mathbb{E}[SE^2] + {C_0}^2\mathbb{E}[SE]). && (18)\end{aligned}$$

Given the simulated power law of spike entropy, we can obtain $E[SE^4]$ and $E[SE^3]$ as 0.24 and 0.158, respectively, producing σ equal to 0.56. When the detection threshold z' is set as 2 (corresponding to $|\tilde{s}|_G$ = 5.55), the probability of $|s|_G \sim \mathcal{N}(5.79, 0.31)$ falling below 5.55 is 33.4%, $i.e.$, the Type-II error.