I Know You Did Not Write That! A Sampling Based Watermarking Method for Identifying Machine Generated Text

Kaan Efe Keleş
TOBB ETU
kaanefekeles@etu.edu.tr

Ömer Kaan Gürbüz
Bilkent University
kaan.gurbuz@bilkent.edu.tr

Mucahid Kutlu
Qatar University
mucahidkutlu@qu.edu.qa

Abstract

Potential harms of Large Language Models such as mass misinformation and plagiarism can be partially mitigated if there exists a reliable way to detect machine generated text. In this paper, we propose a new watermarking method to detect machine-generated texts. Our method embeds a unique pattern within the generated text, ensuring that while the content remains coherent and natural to human readers, it carries distinct markers that can be identified algorithmically. Specifically, we intervene with the token sampling process in a way which enables us to trace back our token choices during the detection phase. We show how watermarking affects textual quality and compare our proposed method with a state-of-the-art watermarking method in terms of robustness and detectability. Through extensive experiments, we demonstrate the effectiveness of our watermarking scheme in distinguishing between watermarked and non-watermarked text, achieving high detection rates while maintaining textual quality.

1 Introduction

Transformer based Large Language Models (LLMs) (Vaswani, 2017) such as ChatGPT, Llama2 (Touvron et al., 2023) are able to generate texts that closely resemble human authored texts. For instance, Clark et al. (2021) report that untrained humans are not able to distinguish between texts generated by GPT-3 and texts authored by humans. As we train larger models with more parameters on an ever-expanding corpora, their capabilities in generating human-like text are likely to increase (Hoffmann et al., 2022). With their incredible performance in text generation, they become effective tools for automating text based tasks such as summarization and translation (Radford et al., 2019).

However, these LLMs pose various threats to society because they can be also used for bad causes such as generating credible-sounding misinformation (Pan et al., 2023), creating fake product reviews (Adelani et al., 2019) and academic plagiarism (Dehouche, 2021). Recent studies have discovered that even though LLM-generated responses may sound convincing, they can be frequently incorrect (Lin et al., 2022).

The potential negative consequences associated with LLMs can be reduced significantly if a reliable detection system is in place to differentiate between machine-generated and human-written texts. A number of researchers focused on this important problem and proposed various approaches such as training a classifier (Solaiman et al., 2019; Ippolito et al., 2020), detecting based on linguistic features (Guo et al., 2023) and log probabilities and perturbations (Mitchell et al., 2023). Data driven methods such as training classifiers requires a wide range of data with different styles, sources, and languages. Currently existing perplexity based detectors are biased against non-native English writers (Liang et al., 2023), raising ethical concerns about their usage in real-world applications.

In this paper we propose a novel model-agnostic watermarking method to detect machine generated text. In watermarking, a hidden pattern is inserted to a passage that is imperceptible to humans but can be easily detected an algorithm.

In our proposal, we interfere with the randomness of sampling a new token to be generated in the decoding phase of LLMs. For each token to be generated, we sample multiple candidate tokens based on their probability provided by the LLM and calculate a *secret number* for each of the candidate tokens. Subsequently, we pick the token with the highest secret number value. The way we calculate the secret number enables us to retrieve the same values from generated text. And our maximization effort lets us discriminate against non-watermarked text.

In our experiments, we evaluate the quality of

Proceedings of the 1st Workshop on GenAI Content Detection (GenAIDetect), pages 140–149
January 19, 2025.

the watermarked texts and how accurately we can detect the watermarks using various datasets and LLMs. We also compare our model against watermarking method of Kirchenbauer et al. (2023a). In our experiments, we show that we are able to detect watermarked texts almost in all cases. In addition, we observe that our method based on sampling with replacement does not reduce the text quality in almost all cases while our method based on sampling without replacement yields slight decrease in text quality. In addition, we show that our proposed method is robust to token level paraphrasing attacks.

The main contributions of our work are as follows. i) We introduce a novel watermarking scheme to detect machine-generated text. In our comprehensive evaluation we show that our watermarks are highly detectable while causing a slight decrease in text quality. ii) We share both our code and dataset to ensure reproducibility of our results and help other researchers build upon our findings.[1].

2 Related Work

The remarkable achievements of Large Language Models (LLMs) compelled researchers to shift their attention towards understanding their potential drawbacks and risks. We direct readers to the survey studies conducted by Crothers et al. (2022) and Weidinger et al. (2021) for an in-depth analysis of the risks associated with LLMs. Now, we focus on studies on detecting texts generated by LLMs.

2.1 Non-Watermarking Detection Methods

Gehrmann et al. (2019) propose a tool GLTR which works in a white-box setting and highlights texts based on probability distribution of tokens provided by the LLMs. They show that their visual tool improves the human detection rate of machine generated text from 54% to 72% without any prior training and without tampering with the text generation phase.

Mitchell et al. (2023) also work in a white-box setting and create perturbations of the candidate text and analyze the negative curvature regions of the model's log probability function. Their main hypothesis for detection is as follows. When machine generated text is modified it tends to have lower log probability. However, modifications on the human-written text may have higher or lower log probability than the unmodified text.

Zellers et al. (2019) examine several schemes to detect fake news article using GROVER which is a language model that generates and classifies fake news articles. They conclude that the most effective model for identifying fake news generated by GROVER is the model itself. Adelani et al. (2019) also report that GROVER is highly accurate in detecting fake reviews. Zellers et al. (2019) argue that machine-generated text classification requires a similar inductive bias as the generator model, rather than expressive capability. However, these findings differ from those of Solaiman et al. (2019) as they claim that a fine-tuned RoBERTa model is a more effective detector than a similarly-capable fine-tuned GPT-2 model.

A number of researchers focused on developing machine learning models to identify generated texts. For instance, Fagni et al. (2021) report that transformer based classifiers to be the best discriminators of fake tweets.

Guo et al. (2023) compile a dataset comprising responses from ChatGPT and human experts across various domains, including finance and medicine, and use it to train classifiers that determine whether a given passage is machine-generated. A similar approach is also followed by the creators of ChatGPT with underwhelming results[2]. In our work, we propose a watermarking method to detect generated texts.

2.2 Watermarking Detection Methods

Abdelnabi and Fritz (2020) introduce the Adversarial Watermarking Transformer (AWT) model, which encodes binary messages in text to trace its origin and prevent malicious use, using a jointly trained encoder-decoder and adversarial training, ensuring the watermark is discreet while maintaining the text's original meaning. Ueoka et al. (2021) proposes using a masked language model, which has a high payload capacity and is less susceptible to automatic detection than generation-based methods. Recently, Christ et al. (2024) introduced a cryptographically inspired method that embeds watermarks using pseudo-random functions and entropy thresholds, ensuring the output distribution remains unchanged.

The closest work to our own is Kirchenbauer et al. (2023a)'s watermarking method. They pro-

[1]The code will be made available soon.

[2]https://openai.com/blog/new-ai-classifier-for-indicating-ai-written-text/

pose selecting a randomized subset of approved tokens from the vocabulary and then promoting the sampling of the tokens from chosen approved subset of the vocabulary via increasing the subsets logits. The randomization is seeded on previously generated token(s) in a context window. In our work, we interfere with the sampling process without changing LLMs' probability distribution over vocabulary while Kirchenbauer et al. (2023a) interfere the probability distribution. In our experiments, we extensively compare our proposed method against Kirchenbauer et al. (2023a)'s.

2.3 Paraphrasing Attacks

As there are tools to detect generated texts, people might want to avoid these detection tools by intentionally changing the generated texts. Therefore, prior work also explored how vulnerable detection systems are against paraphrasing attacks.

Sadasivan et al. (2023) demonstrate how effective off-the-shelf sentence-level paraphrasing models can be at evading detection and conclude that detecting generated text is an unsolvable problem. However, this conclusion is contradicted by Chakraborty et al. (2023) as they show that detection should always be possible when there exist enough samples. Krishna et al. (2023) develop a paraphrasing model which successfully evades several detectors including watermarking (Kirchenbauer et al., 2023a) and DetectGPT (Mitchell et al., 2023). In their proposed detection scheme, the API provider maintains a database containing every sequence generated by their LLM. When a detection query is initiated, this database is queried to identify a previously-generated sequence that exhibits the highest semantic similarity to the query. If the level of similarity surpasses a predefined threshold, the query is classified as machine-generated.

3 Problem Definition

Our goal is to develop a model-agnostic watermarking method to identify generated texts. Let LLM be a large language model and LLM^w is its version with watermarking feature. In addition, let $T_{LLM}(P)$/$T^w_{LLM}(P)$ be a text generated by LLM/LLM^w for the given prompt P. An ideal watermarking method should have the following properties:

- The watermarking process should not decrease the quality of the texts, i.e., the quality of $T_{LLM}(P)$ and $T^w_{LLM}(P)$ should be similar for any given P.
- Watermarking text should not necessitate retraining or fine-tuning.
- We should have the capability to compute a statistical confidence interval with interpretable values for the detection and sensitivity analysis of the watermark.
- The watermark should be robust to perturbations. An adversary must make significant modifications to remove the watermark.

4 Proposed Methodology

In this section, we explain our proposed method to generate watermarked text (Section 4.1) and how to detect the watermark within a given text (Section 4.2).

4.1 Generating Watermarked Texts

In our watermarking method, we interfere with the randomness of picking the next token according to its conditional probability provided by a language model in the decoding stage. The details of our method are shown in **Algorithm 1**.

For a given input prompt P, LLM produces a text T in an iterative way [Lines 1-9]. In each iteration, LLM outputs a conditional probability distribution vector over the vocabulary V for the next token to be generated [Line 3]. We multinomially sample y candidate tokens based on the probability distribution vector [Line 4]. Subsequently, we compute a *secret number* for each candidate token t [Lines 5-7]. In order to compute the secret number of a candidate token (S^t), we first concatenate the k previous tokens and the candidate token t and then calculate their SHA256 hash value. Subsequently, we seed a random number generator with the hash value [Line 6] and generate a random number. Next we pick the token with the highest secret number for the next token [Line 8].

The secret number of any token in a candidate passage only depends on itself and the k tokens that precede it. This enables us to retrieve the same secret number for every token in a passage outside of the generation process. Moreover, if a passage is watermarked we expect the average secret number of the tokens that make up the text to be significantly higher than otherwise. This is because while the production of the non-watermarked text is completely ignorant of the secret numbers of tokens,

Algorithm 1 Text Generation with the Sampling Watermarker

Input: P {Prompt given to the model}
Parameter-1: y {The sampling count}
Parameter-2: k {The context window size}
$T_{LLM}(P) = P$ {Keeps the whole text}
for each token to be generated **do**
  $D = LLM(T_{LLM}(P)))$ {Get the probability distribution from the LLM}
  $C_{[1-y]} = sample(D, y)$ {Sample y candidate tokens}
  for $i \in \{1, \ldots, y\}$ **do**
    $S^{C_i} = RNG(seed = hash(T_{LLM}(P)^{[N,N-k]}, C_i))$ {Calculate the secret number}
  end for
  $T_{LLM}(P) = T_{LLM}(P) + C^{argmax(S^{C_1}, \cdots, S^{C_y})}$ {Concatenate the selected token}
end for

our watermarking scheme actively attempts to maximize this value.

During sampling, we have the option to sample candidate tokens with or without replacement. When we sample without replacement, the secret numbers of the candidate tokens are guaranteed to be distinct values. Maximizing the use of distinct values tends to result in larger secret number values, making the watermark more detectable. On the other hand, if the entropy of the probability distribution is low, i.e., there are few plausible tokens to be generated, sampling without replacement would cause the model to pick the unlikely tokens, reducing the quality of the generated text. Therefore, we also explore sampling with replacement and evaluate the impact of both sampling methods in Section 5.

4.2 Detecting the watermark

In order to detect whether a given text X is watermarked or not, i.e., a text generated by our scheme or not, we first tokenize X and calculate the secret number of each token in X. The secret number of the r^{th} token of X can be calculated as follows.

$$S^{X_r} = RNG(seed = hash(X_{(r-k)}, \cdots, X_{(r)}))$$

where RNG is a random number generator which draws values from a continuous uniform distribution spanning the interval from zero to one. The anticipated mean of the secret number for the tokens composing a text aligns with a normal distribution characterized by an expected mean of 0.5 and an expected variance of $\frac{1}{12*N}$ (See Blitzstein and Hwang (2015) for explanation), where N represents the number of tokens within the given text X. As the length of the candidate text increases, the average secret number for non-watermarked text gradually approaches this theoretical distribution with diminishing variance, thus reducing the likelihood of the text's average secret number deviating significantly from 0.5. Conversely, during the watermarking process, tokens are selected from a set of candidates based on their possession of the highest secret number (out of y candidates). This selection dramatically alters the distribution of the average secret number, rendering it exceedingly improbable for the text to have arisen through natural generation. Thus, we classify the text as watermarked if a certain threshold is exceeded. Formally, we define the following null hypothesis.

H_0: *The text sequence is generated without any attempt to maximize the secret number average.*

The formula of the z-score for testing the hypothesis is as follows:

$$z = (\overline{sna} - 0.5)/\sqrt{1/(12 \cdot N)} \tag{1}$$

where $\overline{sna}$ denotes the secret number average of the candidate text and N represents how many tokens make up the candidate text. The null hypothesis is rejected (and the watermark is detected) if $z-score$ is above a chosen threshold u.

5 Experiments

5.1 Experimental Setup

In this section, we explain evaluation metrics (Section 5.1.1) to assess the quality of our watermarking method, describe the models we used for watermarking (Section 5.1.2), baseline methods we compare against our methods (Section 5.1.3), and datasets we utilized in our experiment (Section 5.1.4). Lastly, we provide details about implementation details (Section 5.1.5).

5.1.1 Evaluation Metrics

In order to measure the quality of watermarking methods, we focus on the quality of the generated text and our detection rate. We adopt the measures used by related prior work (Kirchenbauer et al., 2023b; Krishna et al., 2023). In particular, we calculate how the generated texts are similar to the human authored ones using *P-SP* (Wieting et al., 2023). In addition, we use *diversity* which aggregates n-gram repetition rates. A high diversity score represents a more diverse text where fewer n-grams are repeated (Li et al., 2023). Given the fraction of unique n-grams (which is denoted as u_n) diversity up to the N^{th} order is defined as follows.

$$\text{diversity} = -\log\left(1 - \prod_{n=1}^{N}(1 - u_n)\right) \tag{2}$$

Lastly, we use *coherence* to measure the semantic coherence between the prompt and the generated text. We employ the sentence embedding method, SimCSE (Gao et al., 2022) for this calculation. Given the prompt $\boldsymbol{x}$ and the generated text $\hat{\boldsymbol{x}}$, the coherence score is defined as $v_{\boldsymbol{x}}^{\top} v_{\hat{\boldsymbol{x}}} / (\|v_{\boldsymbol{x}}\| \cdot \|v_{\hat{\boldsymbol{x}}}\|)$, where $v_{\boldsymbol{x}} = \text{SimCSE}(\boldsymbol{x})$ and $v_{\hat{\boldsymbol{x}}} = \text{SimCSE}(\hat{\boldsymbol{x}})$.

5.1.2 Models

As our approach can be applied in any model, we utilize three different models that our hardware systems could execute. In particular, we use OPT (Zhang et al., 2022) with 1.3B parameters, BTLM-3B (Dey et al., 2023) with 3B parameters, and Llama2 (Touvron et al., 2023) with 7B parameters. All of the models were loaded using 4-bit quantization (Dettmers et al., 2023) to minimize memory usage.

5.1.3 Baseline Methods

We compare our proposed method against the study by Kirchenbauer et al. (2023a), also known as the "Maryland Watermark" (MWM). For their method's configuration parameters, we follow the default settings specified in their publicly available repository[3], setting the greenlist fraction γ to 0.25 and the logit bias δ to 2. Additionally, we utilized their repository's evaluation pipeline to compute their z-scores, ensuring consistency in the comparison metrics.

5.1.4 Datasets

In our experiment, we use two different datasets: i) the train split of the 'realnewslike' portion of the C4 (stands for "Colossal Clean Crawled Corpus") dataset (Raffel et al., 2020) and ii) the train split for Wikitext (103-v1-raw) dataset (Merity et al., 2016). C4 is an extensive web text collection resembling real news articles while Wikitext consists of 100M tokens extracted from the set of verified *Good* and *Featured* articles on Wikipedia, providing a more structured and manageable source.

We use the first 100 tokens of the passages as prompts. In order to have a fair comparison, we use 200 tokens for all cases. Therefore, we allow models to generate maximum 200 new tokens. For a given prompt, if any of the generated text is less than 200 tokens, we discard it, and try another prompt drawn from the corresponding dataset. We continue this process until we reach 500 samples for each dataset. Eventually, for each dataset and model we use, we create five text subdatasets: i) texts generated by Maryland watermarking (T_{MWM}), ii) texts generated by our approach with sampling with replacement (T_{SWR}), iii) texts generated by our approach with sampling without replacement (T_{SWOR}), iv) texts generated without watermark (T_{NoWM}), and v) texts authored by humans (T_{Humans}).

5.1.5 Implementation

We implemented the sampling watermarker using the PyTorch (Paszke et al., 2019) backend of the Hugging Face library (Wolf et al., 2019). We utilized the `generate` API provided by Hugging Face for generating text. This API allows for passing a custom `LogitsProcessor` which can be used to modify the prediction scores of a language model head for generation. We use Top-k sampling (Fan et al., 2018) with $top-k = 40$ before doing any sampling on all methods. For our proposed method we set the context window size k to 1 and sampling count y to 5 unless otherwise is mentioned.

5.2 Experimental Results

This section comprises of four subsections, each serving distinct research objectives. The first (Section 5.2.1) assesses watermark detectability, the second (Section 5.2.2) examines textual quality under watermarking, the third (Section 5.2.3) evaluates watermark robustness against attacks, and the final subsection (Section 5.2.4) investigates the impact of various generation parameters on watermarking performance.

5.2.1 Detectibility Experiments

In this experiment, we assess how accurate watermark detection mechanisms work. Specifically, we run our watermarking methods and MWM for all datasets we create and calculate average z-scores

[3]https://github.com/jwkirchenbauer/lm-watermarking

over the generations. In addition, we set the z-score threshold (u) to 4 for both watermarking schemes as in Kirchenbauer et al. (2023a) and calculate the percentage of the texts detected as watermarked. The results are shown in **Table 1**.

The average z-scores exceed 10 in most of the watermarked texts, and is near 0 for non-watermarked text, showing the effectiveness of watermarking schemes. SWOR achieves achieves the highest z-score and detection rates in watermarked texts.

Our watermarking methods consistently avoid false positives when applied to human authored text, whereas MWM occasionally misidentifies such content as watermarked. Moreover, both MWM and our approach have higher false positive when dealing with non-watermarked machine-generated text compared to human authored text. This is because non-watermarked machine-generated text inherently resembles watermarked machine-generated text.

5.2.2 Textual Quality Experiments

In this experiment, we assess how watermarking affects the textual quality. We report P-SP, diversity, and coherence scores in in **Table 2** for texts watermarked with our approaches, Maryland Watermarking, and without any watermark.

Regarding similarity with respect to human authored text (P-SP), we observe that MWM achieves higher scores than our methods for OPT-1.3B and BTLM-3B. However, SWR outperforms others when Llama2-7B is used for generation. Interestingly, SWR even yields higher P-SP score than non-watermarked text with Llama2-7B in Wikitext. We observe a similar pattern in other metrics such that MWM yields higher score with OPT-1.3B and BTLM-3B models than our models in most of the cases. On the other hand, SWR outperforms others with the largest model we use. Regarding SWOR vs. MWM with Llama2-7B is mix such that SWOR outperform MWM in Wikitext but not in C4.

5.2.3 Robustness Experiments

In order to assess how vulnerable the watermarking methods are against token level paraphrasing attacks, we conduct an experiment similar to the one in Kirchenbauer et al. (2023a). In particular, we randomly pick $\%t$ of tokens in the watermarked and mask them. Next, we use DistilRoBERTa-Base model (Sanh et al., 2020) to replace masked tokens, ensuring that the model did not predict the same token that was initially masked. **Figure 1** shows how different attack percentages effect the detection of the watermarked text. Sampling without replacement achieves high detection rates even in attacks with %40, outperforming all other methods. Sampling with replacement and Maryland Watermarker achieve similar detection rates.

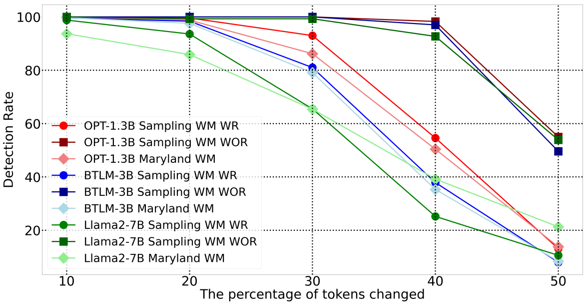

Figure 1: Impact of paraphrasing attacks on the detection rate of watermarked texts.

5.2.4 The Impact of Sampling Count

We explore the impact of the sampling count used for secret number generation, y on the quality of the generated texts and the detection rate. In particular, we vary y from 2 to 11 and generate text using our approach with and without replacement using C4 dataset and Llama-2-7B model. **Table 3** shows the text quality metrics along with average z-score and detection rate. We observe that increasing the sampling count y results in decreasing quality scores in all cases, but yields higher z-scores. Detection rate for SWOR remains at %100 even at a low sampling count of $y = 2$ and SWR achieves 99% rate when $y = 5$.

5.2.5 Entropy in Probability Distribution

The effectiveness of our proposed method and the Maryland watermarking depends on the language model's output distribution. For instance, if the model outputs a low entropy distribution for the next token, our sampling with replacement based method is likely to sample the same y tokens as candidates. However, in sampling without replacement case, the watermarker is guaranteed to sample y unique tokens and pick the one that has the highest secret number.

In this experiment, we manually manipulate the output distribution entropy of our models by adjusting the sampling temperatures to assess its impact. **Table 4** shows the average z-score for varying temperature values for Llama2-7B model on C4 dataset. As expected we observe that both SWR and MWM

Text	Detector	C4						Wikitext					
		OPT-1.3B		BTLM-3B		Llama2-7B		OPT-1.3B		BTLM-3B		Llama2-7B	
		z-score	%WM	z-score	%WM	z-score	%WM	z-score	%WM	z-score	%WM	z-score	%WM
T_{SWR}	SWR	11.31	99.8%	10.11	99.8%	9.44	99%	12.09	99.8%	10.33	100%	10.36	99.8%
T_{SWOR}	SWOR	**16.85**	100%	**16.29**	100%	**16.66**	100%	**16.92**	100%	**16.26**	100%	**17.23**	100%
T_{MWM}	MWM	10.77	100%	9.82	100%	9.71	99.4%	11.79	100%	10.43	100%	10.65	97%
T_{Humans}	SWR	0.27	0%	-0.07	0%	0.22	0%	0.03	0%	-0.05	0%	0.28	0%
	MWM	-0.23	0%	-0.46	0.2%	0.21	0.2%	0.35	0.6%	0.21	0.2%	-0.01	0.2%
T_{NoWM}.	SWR	0.22	0%	-0.25	0%	0.44	1.4%	0.69	0.6%	-0.22	0%	0.17	3.6%
	MWM	-0.25	0%	-0.42	0.2%	0.32	1%	0.01	0.4%	-0.17	0.2%	0.39	3.4%

Table 1: The average z-scores over the generations when attempted to detect the watermark and the ratio of samples detected as “watermarked" by the corresponding detector. The text in **bold** represent the highest z-score for watermarked text and lowest for baseline completion text.

Metric	Method	C4			Wikitext		
		OPT-1.3B	BTLM-3B	Llama2-7B	OPT-1.3B	BTLM-3B	Llama2-7B
P-SP	SWR	0.44	0.48	**0.48**	0.45	0.48	**0.52**
	SWOR	0.40	0.42	0.38	0.42	0.43	0.44
	MWM	**0.46**	**0.49**	0.45	**0.47**	**0.49**	0.41
	NWM	0.47	0.50	0.48	0.49	0.49	0.46
Diversity	SWR	6.92	7.50	**8.16**	6.26	7.06	**6.96**
	SWOR	6.84	7.49	7.48	6.42	7.23	6.66
	MWM	**7.40**	**7.90**	5.88	**6.77**	**7.46**	5.38
	NWM	7.87	7.87	6.17	7.16	7.55	6.1
Coherence	SWR	0.63	**0.64**	0.64	0.67	**0.66**	**0.65**
	SWOR	0.58	0.59	0.53	0.63	0.60	0.54
	MWM	**0.64**	**0.64**	**0.65**	**0.68**	**0.66**	0.58
	NWM	0.66	0.66	0.67	0.70	0.66	0.62

Table 2: The impact of watermarking on the the quality of the generated text. The highest score among watermarked texts for each case is shown in **bold**. MWM: Maryland Watermarking, SWR: Sampling with replacement, SWOR: Sampling without replacement, NWM: No Watermarking.

y	P-SP		Diversity		Coherence		z-score		Detection Rate	
	SWR	SWOR	SWR	SWOR	SWR	SWOR	SWR	SWOR	SWR	SWOR
2	0.49	0.45	8.33	8.65	0.66	0.61	4.79	8.33	%76	%100
5	0.48	0.38	8.16	7.48	0.64	0.53	9.44	16.66	%99	%100
8	0.46	0.34	7.66	6.4	0.62	0.50	11.72	19.51	%100	%100
11	0.45	0.30	7.65	5.83	0.62	0.46	12.91	20.94	%100	%100

Table 3: The effect of sampling count y on textual quality metrics. Model: Llama-2-7B, Dataset:c4, k:1.

exhibit stronger watermarks when the output distribution entropy is higher. SWOR shows slight variations in the average z-score but these are just statistical noises as SWOR is designed to be unaffected by the underlying distribution entropy.

Temperature	0.8	0.9	1	1.1	1.2
SWR	8.14	8.91	9.44	10.38	10.82
SWOR	16.89	16.75	16.66	16.68	16.61
MWM	8.02	8.85	9.71	10.65	11.24

Table 4: The effect of sampling temperature on the average z-score. Lower temperatures yield output distributions with lower entropy vice versa. Model: Llama2-7B, Dataset:C4, k:1,y:5

6 Limitations

While our work makes a significant contribution to the research on LLMs, there are certain limitations that warrant further exploration in the future. Firstly, the prompts used in our experiments are derived from two datasets. However, watermarking performance is highly dependent on the nature of the prompt. For example, when asking a factual question (e.g., "What is the full text of the U.S. Constitution?"), watermarking the generated output becomes challenging due to the limited flexibility in the model's response. To address this, a broader range of datasets covering diverse topics is necessary. Furthermore, our experiments were conducted using only three models, primarily due to hardware constraints. Since the performance of watermarking methods is influenced by the specific models used for text generation, evaluating a wider variety of LLMs is essential for more robust assessments. Additionally, we did not account for human paraphrasing in our evaluation, which limits the scope of robustness testing and highlights another avenue for future research.

Furthermore, in our study, we focus on only the task of completing a text for a given prompt. We acknowledge that further evaluation of the proposed watermark across different down stream tasks such as question answering and summarization would be beneficial. We leave this exploration as future work.

Lastly, we explore only token level paraphrasing attacks to measure the robustness of the models. There exist different methods for manipulating text to evade watermarking detection such as deletion, unicode attacks and human paraphrasing. Thus, other types of attacks should be explored to further analyze the robustness of watermarking methods.

7 Conclusion and Future Work

In this work, we propose a watermarking scheme which embeds a unique pattern into the generated text while preserving its coherence and natural readability for human readers. Specifically, We modify the token sampling process of LLMs. In particular, we first sample multiple tokens based on probability distribution over vocabulary and then calculate a unique secret number for each sampled one. We always pick the token with the highest secret number, allowing us to trace the hints of generation process.

In our experiments with multiple datasets and LLMs, we show that our method we show that our watermarking is detectable and reduce slight decrease in text quality. Furthermore, our method outperforms Kirchenbauer et al. (2023a)'s method in terms of detectability and robustness. Regarding text quality, we achieve slightly superior results compared to Kirchenbauer et al. (2023a) when applied to larger models, albeit with less favorable outcomes when dealing with smaller models.

There are multiple research directions we plan to extend in the future. Firstly, we plan to conduct our experiments on a larger scale in terms of data and model size and types. Secondly, a more sophisticated watermark could be implemented by adaptively choosing the sampling count y based on the entropy of the output distribution. Specifically, when the output distribution exhibits low entropy, we can select a smaller value for y and conversely, when the entropy is high, we can opt for a larger value. This method would ensure less perplexity on low entropy text while allowing for a stronger watermark to be embedded on higher entropy text. We leave this extension as a future work.

Lastly, there are no inherent obstacles to abstaining from the concurrent application of both our and Kirchenbauer et al. (2023a)'s watermarks during text generation. This would enable texts that are detectable by both watermarking methods. Employing two relatively less intrusive watermarks might potentially better maintain the textual quality while preserving high detectability.

References

Sahar Abdelnabi and Mario Fritz. 2020. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. *arXiv preprint*.

David Ifeoluwa Adelani, Haotian Mai, Fuming Fang, Huy H. Nguyen, Junichi Yamagishi, and Isao Echizen. 2019. Generating sentiment-preserving fake online reviews using neural language models and their human- and machine-based detection. *Preprint*, arXiv:1907.09177.

Joseph K. Blitzstein and Jessica Hwang. 2015. *Introduction to Probability*. CRC Press.

Souradip Chakraborty, Amrit Singh Bedi, Sicheng Zhu, Bang An, Dinesh Manocha, and Furong Huang. 2023. On the possibilities of ai-generated text detection. *Preprint*, arXiv:2304.04736.

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

Elizabeth Clark, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah A. Smith. 2021. All that's 'human' is not gold: Evaluating human evaluation of generated text. *Preprint*, arXiv:2107.00061.

Evan Crothers, Nathalie Japkowicz, and Herna Viktor. 2022. Machine generated text: A comprehensive survey of threat models and detection methods. *arXiv preprint*.

Nassim Dehouche. 2021. Plagiarism in the age of massive generative pre-trained transformers (gpt-3): "the best time to act was yesterday. the next best time is now.". *Ethics in Science and Environmental Politics*, 21.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Preprint*, arXiv:2305.14314.

Nolan Dey, Daria Soboleva, Faisal Al-Khateeb, Bowen Yang, Ribhu Pathria, Hemant Khachane, Shaheer Muhammad, Zhiming, Chen, Robert Myers, Jacob Robert Steeves, Natalia Vassilieva, Marvin Tom, and Joel Hestness. 2023. Btlm-3b-8k: 7b parameter performance in a 3b parameter model. *Preprint*, arXiv:2309.11568.

Tiziano Fagni, Fabrizio Falchi, Margherita Gambini, Antonio Martella, and Maurizio Tesconi. 2021. TweepFake: About detecting deepfake tweets. *PLOS ONE*, 16(5):e0251415.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. *Preprint*, arXiv:1805.04833.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2022. Simcse: Simple contrastive learning of sentence embeddings. *Preprint*, arXiv:2104.08821.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M. Rush. 2019. GLTR: statistical detection and visualization of generated text. *CoRR*, abs/1906.04043.

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

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models. *Preprint*, arXiv:2203.15556.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2020. Automatic detection of generated text is easiest when humans are fooled. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1808–1822, Online. Association for Computational Linguistics.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023a. A watermark for large language models. *Preprint*, arXiv:2301.10226.

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. *Preprint*, arXiv:2306.04634.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Preprint*, arXiv:2303.13408.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023. Contrastive decoding: Open-ended text generation as optimization. *Preprint*, arXiv:2210.15097.

Weixin Liang, Mert Yuksekgonul, Yining Mao, Eric Wu, and James Zou. 2023. Gpt detectors are biased against non-native english writers. *Preprint*, arXiv:2304.02819.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *Preprint*, arXiv:1609.07843.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. *arXiv preprint*.

Yikang Pan, Liangming Pan, Wenhu Chen, Preslav Nakov, Min-Yen Kan, and William Yang Wang. 2023. On the risk of misinformation pollution with large language models. *Preprint*, arXiv:2305.13661.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. *Preprint*, arXiv:1912.01703.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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. *Preprint*, arXiv:1910.10683.

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

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *Preprint*, arXiv:1910.01108.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, Miles McCain, Alex Newhouse, Jason Blazakis, Kris McGuffie, and Jasmine Wang. 2019. Release strategies and the social impacts of language models. *Preprint*, arXiv:1908.09203.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Honai Ueoka, Yugo Murawaki, and Sadao Kurohashi. 2021. Frustratingly easy edit-based linguistic steganography with a masked language model. *arXiv preprint*.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, Zac Kenton, Sasha Brown, Will Hawkins, Tom Stepleton, Courtney Biles, Abeba Birhane, Julia Haas, Laura Rimell, Lisa Anne Hendricks, William S. Isaac, Sean Legassick, Geoffrey Irving, and Iason Gabriel. 2021. Ethical and social risks of harm from language models. *CoRR*, abs/2112.04359.

John Wieting, Kevin Gimpel, Graham Neubig, and Taylor Berg-Kirkpatrick. 2023. Paraphrastic representations at scale. *Preprint*, arXiv:2104.15114.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *CoRR*, abs/1905.12616.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *Preprint*, arXiv:2205.01068.