Watermarking Large Language Models

Jeremy Goldwasser

1 Motivation: Why Watermark?

The proliferation of powerful Large Language Models (LLMs) raises critical concerns about content authenticity and provenance. Probably the best-known concern is regarding academic integrity. Since the release of ChatGPT in 2022, educators at all levels have had to contend with a tidal wave of cheating. University professors and high school teachers are now inundated with their students' AI-generated content. 2024 surveys found two-thirds of Harvard undergrads relying on AI weekly, and 92% of British students using it in some capacity. This poses a serious challenge for young people's ability to think critically – a core purpose of education.

Yet students aren't entirely to blame for their apparent slothfulness. With a dismal entry-level job market (not to mention exceptionally competitive graduate programs), students are pressured to accomplish as much as possible, and AI enables them to do so efficiently – lest they get left behind by their LLM-using peers.

LLMs pose several societal challenges beyond cheating:

- **Veracity:** Is this dataset, news article, or product review authentic or synthetically generated to manipulate opinion?
- **Security:** Is this an imposter using AI to mimic a person's style for social engineering or fraud?
- **Quality:** Is the internet being flooded with low-quality, AI-generated content that drowns out human creativity?

A common first-line defense is a **Real-or-AI classifier**. However, these models have significant drawbacks:

- They are increasingly unreliable as LLMs become more sophisticated.
- They can be biased, often flagging text from non-native English speakers as AI-generated.
- They are vulnerable to simple *adversarial attacks* (minor edits can fool the classifier).
- They raise privacy concerns, as they require vast amounts of data on how humans interact with and write about AI.

Watermarking offers a proactive solution. Instead of passively trying to detect AI text from the outside, we embed a subtle, statistical signal into the generation process itself.

2 The Watermarking Framework

2.1 Desiderata

A successful watermarking scheme should satisfy two primary goals:

1. **Imperceptibility:** The watermark should not degrade the quality or utility of the LLM's output. The distribution of watermarked text should be statistically indistinguishable from the original LLM's distribution.

2. **Robustness:** The watermark signal should be detectable even after the text has been edited (e.g., paraphrasing, deleting sentences).

2.2 General Setup

At each step t of text generation, an LLM produces a probability distribution P_t over its entire vocabulary V for the next token w_t, conditioned on the previous tokens $w_{1:t-1}$.

A standard LLM simply samples from this distribution: $w_t \sim P_t$. A watermarked LLM alters this process slightly.

1. A **private key** k is known only to the model's owner (e.g. OpenAI).

2. At each step t, a **pseudorandom variable** ζ_t is deterministically generated using a cryptographic hash function H that takes the key k and the recent context $w_{t-c:t-1}$ as input. The hash function generates a seed, which is used for a pseudorandom number generator:

$$s_t = H(k, w_{t-c:t-1}); \quad \zeta_t = \text{PRNG}(s_t)$$

   For example, $H(\cdot, \cdot)$ might use SHA-256, a standard cryptographic hashing algorithm. This deterministically outputs a random-looking 64-digit hexadecimal string, which corresponds to an integer between 0 and $2^{256} - 1$. With the modulo operator, this is converted to a viable seed s_t.
   Note (Aaronson): Using a limited context of c previous tokens (e.g., $c = 5$) instead of the entire history $w_{1:t-1}$ makes the watermark more robust to edits. An attacker would have to change every c-gram to erase the signal.

3. A special **decoder** S uses both the LLM's distribution P_t and the pseudorandom variable ζ_t to select the next token:

$$w_t = S(P_t, \zeta_t)$$

The detection task is then: given a piece of text, the key k, the hash function H, and the decoder algorithm S, determine if the text was generated by this watermarked LLM.

3 Detection as Hypothesis Testing

We can frame the detection problem as a formal hypothesis test.

- **H_0 (Null Hypothesis):** The text is human-generated.
- **H_1 (Alternative Hypothesis):** The text was generated by the watermarked LLM.

To conduct this test, we calculate a **test statistic** from the text. The core insight is:

*For human-generated text, the choice of the next word is statistically independent of the pseudorandom variable ζ_t that **would have been** generated by the watermarking scheme.*

This independence under H_0 allows us to derive the distribution of our test statistic and determine if the observed value is statistically significant. We typically assume, under the null, that the human's next-token distribution is reasonably approximated by the LLM's distribution P_t.

4 Biased Watermarks

A simple, if unsubtle, approach is to intentionally bias the output distribution. While straightforward, the downside is that this method is not truly imperceptible; it explicitly changes the output distribution, which could slightly degrade text quality. The most well-known approach is from "A Watermark for Large Language Models" [1].

Setup: In this context, ζ_t is a pseudorandom partition of the vocabulary V into two sets. For hyperparameter γ (e.g. 0.5), one set of $\gamma|V|$ elements constitutes the "green list" V_{green}, and the remaining $(1-\gamma)|V|$ elements form the "red list" V_{red}.

Decoding Rule: The LLM's logits (pre-softmax probabilities) are modified to slightly increase the probabilities of tokens in V_{green} and decrease the probabilities of tokens in V_{red} before sampling.

A naive approach would simply remove all the elements of the red list; however, this would bias generation too harshly, sometimes prohibiting obvious next tokens from being sampled. e.g. The word "Barack" should be followed by "Obama," so the text would be a lot worse if we weren't able to say his last name.

Instead, what is actually proposed is a soft rule, where the green list's logits are increased by some hyperparameter $\delta > 0$. After the softmax, this increases their probabilities, and decreases those in the red list. Higher δ corresponds to more upweighting; this makes detection easier but generation worse.

$$\hat{p}_k^{(t)} = \begin{cases} \dfrac{\exp\big(\ell_k^{(t)} + \delta\big)}{\sum_{i\in R}\exp(\ell_i^{(t)}) + \sum_{i\in G}\exp(\ell_i^{(t)} + \delta)}, & k \in G, \\ \dfrac{\exp\big(\ell_k^{(t)}\big)}{\sum_{i\in R}\exp(\ell_i^{(t)}) + \sum_{i\in G}\exp(\ell_i^{(t)} + \delta)}, & k \in R. \end{cases}$$

Detection: Detection is straightforward: check if the text has a statistically significant overrepresentation of "green list" tokens compared to what would be expected from a normal text.

Under the null hypothesis, which token is selected has nothing to do with the green versus red list. Therefore the number of green-list tokens, X_G, is binomially distributed under the null, with N being the total number of tokens and γ being the proportion of green-list tokens at each step. Taking a normal approximation to this distribution, we test for significance with the Z-statistic

$$Z = \frac{X_G - \gamma N}{\sqrt{N\gamma(1-\gamma)}}.$$

The test rejects when $Z > Z_{1-\alpha}$, the upper level-α quantile of the standard normal distribution.

The paper also analyzes the distribution of Z under the alternate hypothesis. They lower-bound its expected value, and upper-bound its variance; this gives a sense of the test's ability to discern watermarked text.

5 Unbiased Watermarks

5.1 Gumbel-Max Watermarking

This method, from Scott Aaronson, uses the **Gumbel-Max trick** to sample in an unbiased way. While unpublished, it has been implemented at OpenAI, and described in a Simons Talk and related papers.

Setup: The pseudorandom variable ζ_t is a vector of values $\{U_w\}_{w \in V}$, where each U_w is drawn from a Unif$(0, 1)$ distribution. Of course, this sampling is deterministic given the seed. (We'll drop the t subscripts for convenience.)

Decoding Rule: The next token w_t is chosen deterministically to maximize a score computed from the LLM's probability distribution P_t and the corresponding pseudorandom number U_w:

$$w_t = \arg\max_{w \in V} U_w^{1/P_w} = \arg\min_{w \in V} -\frac{\log(U_w)}{P_w}$$

Imperceptibility (Gumbel-Max Trick): This scheme is **unbiased**. It can be shown that this deterministic rule is equivalent to sampling directly from P_t.

$$P(w = \arg\min_{w \in V} -\frac{\log(U_w)}{P_w}) = P_w.$$

In general, the Gumbel-Max trick provides a way for sampling from a categorical distribution based on sampling from a continuous random variable. It has applications in other areas of statistics and machine learning, for example in variational inference and reinforcement learning settings where one aims to optimize a model based on discrete sampling using a differentiable approximation. The formula presented here is known as the "exponential race view"; an alternative formulation involves sampling from a Gumbel distribution, which appears in extreme value theory.

Proof Sketch. The probability that a word w is chosen is the probability that its score, $S_w := -\frac{\log(U_w)}{P_w}$, is the minimum among all words.

Claim 1. We start off by claiming that $S_w \sim Exp(P_w)$. We can derive this by finding the cumulative distribution function (CDF) of S_w, denoted $F_{S_w}(s) = P(S_w \leq s)$. For any $s > 0$:

$$\begin{aligned} F_{S_w}(s) &= P(S_w \leq t) \\ &= P\left(-\frac{\log(U_w)}{P_w} \leq t\right) \\ &= P(-\log(U_w) \leq tP_w) \\ &= P(\log(U_w) \geq -tP_w) \\ &= P(U_w \geq e^{-tP_w}) \end{aligned}$$

Since for a uniform distribution on $(0, 1)$, $P(U_w \geq x) = 1 - x$, we have:

$$F_{S_w}(s) = 1 - e^{-sP_w}$$

This is the CDF of an exponential distribution with rate parameter $\lambda = P_w$. Therefore, S_w is exponentially distributed with rate P_w.

Claim 2. Next, we show that the minimum of independent exponential random variables is also exponential, with rate equal to the sum of the individual rates.

Let $S_i \sim \text{Exponential}(P_i)$ be independent, and let $Z = \min(S_w)$. The survival function of Z is:

$$\begin{aligned} P(Z > t) &= P(\forall i, S_i > t) \\ &= \prod_i P(S_i > t) \\ &= \prod_i e^{-P_i t} \\ &= e^{-t \sum_i P_i} \end{aligned}$$

This is the survival function for an exponential distribution with rate $\lambda = \sum_i P_i$. Thus, $Z \sim \text{Exponential}(\sum_i P_i)$.

We are ready to complete the proof. Word w is selected with probability

$$P(S_w < S_{w'} \ \forall w' \neq w) = P(S_w < Z)$$

where $S_w \sim \text{Exp}(P_w)$ and $z \sim \text{Exp}(\sum_{w' \neq i} P(w')$.

$$\begin{aligned} P(S_w < Z) &= \int f(Z > s, S = s)\, ds \\ &= \int_0^\infty f_{S_w}(s) P_Z(Z > s) ds \\ &= \int_0^\infty P_w e^{-sP_w} \cdot e^{-s \sum_{w' \neq w} P_{w'}} ds \\ &= P_w \int_0^\infty e^{-s \sum_{w'} P_{w'}} ds \\ &= P_w \left[-\frac{e^{-s \sum_{w'} P_{w'}}}{\sum_{w'} P_{w'}} \right]_0^\infty \\ &= P_w \frac{1}{\sum_{w'} P_{w'}} \\ &= P_w \end{aligned}$$

Claim 2 and the subsequent proof reflects a basic fact of Exponential distributions and Poisson processes: The probability that one exponentially-distributed "ticking clock" is the first to chime in a race against the others is proportional to its weight. □

Detection: Scott Aaronson proposed a test statistic based on this scheme. For a given text of length N, we re-compute the pseudorandom vectors $\{\zeta_t\}_{t=1}^N$ that would have been generated. Let U_{w_t} be the pseudorandom number associated with the *actual token* w_t observed in the text at step t. The test statistic T is:

$$T = \sum_{t=1}^{N} -\log(1 - U_{w_t})$$

Intuition: The LLM's decoding rule favors tokens w with high scores, which often corresponds to high values of U_w. Humans, unaware of the U_w values, will pick tokens independently of them. Therefore, if the text is AI-generated, the chosen tokens will tend to have high U_{w_t} values, leading to a large test statistic T.

Null Distribution: Under H_0, each U_{w_t} is effectively a random draw from Unif$(0, 1)$. Since $-\log(1 - U) \sim$ Exponential(1) for $U \sim$ Unif$(0, 1)$, the test statistic T is a sum of N i.i.d. Exponential(1) random variables.

$$T \sim \text{Gamma}(N, 1)$$

This known distribution allows us to calculate a p-value and decide whether to reject the null hypothesis. The mean is N and the variance is N.

5.2 Inverse Transform Watermarking

This method from Percy Liang and collaborators uses inverse transform sampling, another classic technique for drawing from a distribution [2].

Setup: The pseudorandom variable ζ_t consists of two parts generated from the hash of the key and context:

1. A permutation π_t of the vocabulary V.
2. A single number $u_t \sim$ Unif$(0, 1)$.

Decoding Rule:

1. Order the vocabulary according to the pseudorandom permutation π_t.
2. Calculate the CDF of P_t according to this new ordering.
3. Apply standard inverse transform sampling: find the first word w in the permuted list whose cumulative probability is greater than u_t. This word is the chosen token w_t.

Like the Gumbel-Max method, this is also an unbiased sampling scheme, ensuring imperceptibility. Detection follows a similar hypothesis testing framework, where one checks if the observed tokens fall into suspiciously predictable parts of the CDF given the key.

A recent work published in the Annals of Statistics analyzes the detection rules from the Gumbel-Max and Inverse Transform watermarks [3]. They demonstrate that they may have relatively weak power, and derive optimal alternatives.

6 Practical Limitations

LLM watermarking is a technique that embeds a hidden statistical signature into AI-generated text to trace its origin. While there has been considerable research progress in recent years, the system is fragile and easily circumvented. Users can evade detection by simply using a non-watermarked model like any open-source LLM, or by paraphrasing and editing the watermarked text to disrupt the signal. A significant challenge is that there is no universal detector; each company's watermark requires its own proprietary tool for identification, so no single service can check for watermarks from all the different major AI labs.

Despite being framed as a tool for public safety, watermark detection is not currently enabled for the general public. One potential reason behind this is that providing a public-facing detector could allow malicious actors to test their evasion methods and learn how to perfectly remove the watermark. Instead, the intended users are researchers tracking the spread of AI content, social media platforms needing to identify large-scale spam or influence campaigns, and the AI companies themselves for internal forensic analysis. It is a tool for institutional-level tracking, not individual verification.

References

[1] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR, 23–29 Jul 2023.

[2] Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*, jun 2024. Accepted and published June 2, 2024.

[3] Xiang Li, Feng Ruan, Huiyuan Wang, Qi Long, and Weijie J Su. A statistical framework of watermarks for large language models: Pivot, detection efficiency and optimal rules. *The Annals of Statistics*, 53(1):322–351, 2025.