Foundations for Understanding and Baselines for Detecting Watermarks in Large Language Models

Leonard Tang [1] **Gavin Uberti** [1] **Tom Shlomi** [1]

Abstract

We consider the emerging problem of detecting the presence of watermarks in widely used, publicly hosted, closed source large language models (LLMs). We introduce a suite of baselines for watermark detection that rely on analyzing distributions of outputs and logits generated by watermarked and unmarked LLMs. Watermarked LLMs tend to produce distributions that diverge qualitatively and detectably from normal models. We investigate the detectability of watermarks with varying strengths, and study tradeoffs of each algorithm with respect to watermarking scenario. Along the way, we formalize the watermarking problem and provide a framework and foundations for theoretically studying watermarks. Ultimately, we set up a monitoring system to determine when publicly hosted LLMs become watermarked.

1. Introduction

Recent progress in large language models (LLMs) has resulted in a rapid increase in the ability of models to produce convincingly human-like text, sparking worries that LLMs could be used to spread disinformation, enable plagiarism, and impersonate people. As such, much work has been done to develop methods to detect AI generated text. These include watermarking algorithms, which subtly modify the outputted text to allow for better detection, given that the detector has sufficient access to watermarking parameters. Differing from previous work, here we propose a series of *black-box* tests for whether a language model is watermarked that only relies on querying the model, and does not require any knowledge of the underlying watermark generation parameters.

[1]Department of Computer Science, Harvard University, Cambridge, MA. Correspondence to: Leonard Tang <leonardtang@college.harvard.edu>.

2^{nd} AdvML Frontiers workshop at 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

2. Related Work

Several papers have proposed the detection of AI generated text by training a detection model to distinguish between AI and human generated text. These include classifiers trained on examples of human and AI generated text, such as a method developed by OpenAI which had detection rates of around 95% for detecting GPT-2 generated text (Solaiman et al., 2019). Improvements have been made to this method including methods like DetectGPT which use queried log probabilities of texts to zero-shot classify LLM generated text (Mitchell et al., 2023).

There is a limit as to the effectiveness of classifier-based AI text detection, as the text becomes closer to human generated text. The total variation distance of two distributions $\mathcal{L}, \mathcal{H}$, denoted $\text{TV}(\mathcal{L}, \mathcal{H})$, is half of the total absolute difference between the probability they assign to all text. Sadasivan et al. showed that the probability of any detector D rating a given positive instance higher than a given negative instance ($\text{AUROC}(D, \mathcal{L}, \mathcal{H})$ can be tightly bounded as seen below (Sadasivan et al., 2023). Thus, as language models get closer to human generated text, inherent limits on detection come into play.

$$AUROC(D, \mathcal{L}, \mathcal{H}) \leq \frac{1}{2} + \text{TV}(\mathcal{L}, \mathcal{H}) - \frac{\text{TV}(\mathcal{L}, \mathcal{H})^2}{2}$$

Instead of relying on detectors being able to distinguish human generated text from the output of language models trained to imitate them, watermarking schemes can be used to make text more detectable. Natural language watermarking dates back to at least the early 2000s, well before the development of large language models (Topkara et al., 2005). These schemes include synonym substitution, syntactic transformation, and semantic transformations. Thought this work predates large language models, they could be applied on top of their outputs. However, even if they preserve strict syntactic meaning, they often fail to take into account stylistic constraints.

Closely related to watermarking is steganography, which has the goal of hiding information in text. These share a goal of modifying text to include information readable to a

detector while keeping the text similar to unmodified text, though steganography usually does not aim to be robust to changes. In the past few years, adversarial training process to have been used to train a a language model to hide a message in a piece of text (Abdelnabi & Fritz, 2021). The field of steganalysis studies the detection of steganographic methods, though to our knowledge no methods have been developed that would be appropriate to use to detect large language model watermarking (Yang et al., 2022).

Recently, Kirchenbauer et al. (2023) propose multiple watermarking schemes which minimize damage to output quality. A hard watermark that utilizes a red-list of words which never appear, and a one which instead merely makes red-listed words less likely. The soft watermark is the primary watermark that we test our detection mechanisms against. For each token, both of these use a hash of the previous token to seed a random number generator, then uses this random number generator to split the list of tokens in half. For the hard watermark, half of those tokens have their probabilities set to 0 when the next token is chosen. For the soft watermark, the logits are computed as usual, and then δ is added to half of the token logits, after which softmax is applied and the token is sampled. Watermarked text is detected by computing a z-statistic and comparing it to a threshold. The hard watermark allows watermarked text to be detected after a certain number of tokens (the exact number depends on the acceptable false positive rate), regardless of the content of the generated text. However, it drastically reduces output quality on low-entropy text, as it can prevent a token with probability near 1 from being sampled. The soft watermark fixes this problem, at the cost of making ease of detection depend on the entropy of the generated text.

This method is robust to small changes to the text; changing n token modifies the green-list status of at most $2n$ tokens. If this is a small fraction of the total tokens, then the detector will still detect the watermark with high probability. While this form of robustness remains regardless of whether an attacker knows the watermark, they describe cases where an attacker can more effectively evade the watermark. An attacker can fairly easily generate a red list dictionary through repeatedly sampling a detection API. This can be mitigated by having the pseudorandomness be computed over the previous h tokens, but this then amplifies the number of changes to the red list count with each changed token. They develop a robust private watermarking algorithm, which iterates through the highest likelihood tokens until it finds a token that is on the green list computed from the hash of itself and a pseudorandomly chosen member of the last h tokens.

Relatedly, in unpublished work, Scott Aaronson developed a watermark which provably leaves the output computationally indistinguishable from unwatermarked output (Aaronson, 2023). To do this, for each token generation, a number $r_i \in [0, 1]$ is pseudorandomly chosen for each token i based on the previous h tokens. Probabilities p_i are computed for each of the tokens by the language model, and then a token i is chosen which maximizes r_i^{1/p_i}.

3. A Framework for Understanding Watermarking Terminology

Before introducing our detection algorithms, we first introduce terminology and framework for studying watermarks in LLMs.

3.1. Language Models

From the perspective of watermarking, the way in which a LLM is trained is largely irrelevant, as is the way it works internally. However, it *is* relevant that models operate over an output set of tokens. The way that LLMs generate outputs varies from model to model – popular tokenization methods include byte-pair encoding and `SentencePiece` (Sennrich et al., 2016; Kudo & Richardson, 2018). In theory, a language model could output raw UTF-8 binary codes, and would just need a vocabulary of $\{0, 1\}$. However, in practice the smallest vocabularies are around $30,000$ tokens (BERT), while the largest (Jurassic-1, BLOOM) are near $256,000$. In all cases, though, the translation to and from raw text to a sequence of tokens can be done losslessly. We thus define a vocabulary as follows:

Definition 3.1 (Vocabulary)**.** A *vocabulary* in the context of LLM watermarking is a set $\mathcal{T}$ along with an encoder and decoder $\mathcal{E}, \mathcal{D}$ that encode and decode a sequence of tokens to raw text. It should always be the case that $x = \mathcal{D}(\mathcal{E}(x))$ for the vocabulary to be valid.

Notably, vocabulary encoders and decoders are sensitive to concatenation. While $\mathcal{D}(\mathcal{E}(x)) = x$, it is not always the case that $\mathcal{D}(\mathcal{E}(x_1)\|\mathcal{E}(x)_2) = x_1\|x_2$.

For an explicit example, consider how numerals are tokenized in Google's `Flan-T5` model. Let x be the token with index `755`, which is encoded in text as $\mathcal{E}(x) =$ `"5"`. However, the decoded version of `"55"` is not token `755` twice - instead, it is a single token with index `3769`. We must be cognizant of this when working with LLM vocabularies.

Definition 3.2 (Sequence Model)**.** A *sequence model* S over a vocabulary $\mathcal{T}$ is a map from a finite sequence of tokens $\mathcal{T}^*$ to a set of logits over all tokens with type $\mathbb{R}^{|\mathcal{T}|}$, along with a sampler of type $\mathbb{R}^{|\mathcal{T}|} \to \Delta\mathcal{T}$ which takes in the logits and randomly outputs some token.

It is important to define sequence models as pairs of logit generators and samplers, rather than simply being functions

$\mathcal{T}^* \to \Delta\mathcal{T}$. This allows us to easily deal with a concept of temperature, manipulation of logits, and different samplers.

However, this definition fails to capture a key characteristic of LLM behavior that is critical for considering watermark detection. In all commercially available trained LLMs, the distribution over logits is highly uneven. As a more nuanced definition, consider:

Definition 3.3 (Language Model)**.** Consider a game played between a model S which was trained on a corpus of data C and an adversary who also has access to C. A portion of text $s \subset \mathcal{T}^*$ is sampled from C, and is then fed to $\texttt{softmax}(S(s))$, after which a token is sampled from the resulting distribution. A sequence model is a *language model* if an adversary is able to guess the sampled token with probability much greater than $1/|\mathcal{T}|$ only knowing s and C.

3.2. Language Model Watermarks

With an understanding of LLM mechanics, we now consider watermarks. As described in Section 2, they can be applied either to the pre- or post-softmax results. Since any post-softmax watermark can be made into a pre-softmax watermark by "folding in" the softmax operator, we define a watermark as follows:

Definition 3.4 (Watermark)**.** A *watermark* W_s over a vocabulary $\mathcal{T}$ with key s is a map $W_s : \mathbb{L}_\mathcal{T} \to \mathbb{L}_\mathcal{T}$, where $\mathbb{L}_\mathcal{T}$ is the set of language models with vocabulary $\mathbb{L}_\mathcal{T}$.

We also define a concept of principledness for watermarks.

Definition 3.5 (Principled Watermark)**.** Let $\mathcal{L}_\infty, \mathcal{L}_\in$ be large language models that are identical up to permutation of the tokens. If, for any such LLMs, and all seeds s, $W_s(\mathcal{L}_\infty)$ is identical to $\mathcal{L}_\in$ up to the same permutation of tokens, then we call W a *principled watermark*.

All language model watermarks that we are aware of satisfy this definition, though there are imaginable watermarks that do not. Early language models, for example, were instead transformations applied to text strings. A watermark which only allowed the language model to output blocks of text at a time could be developed, and in fact this would evade the detection methods developed in this paper.

Most watermarks that obey this definition are not useful. The identity watermark is a valid watermark, but does not aide in text detection. We provide the following definition of watermarks that are *detectable*:

Definition 3.6. A watermark W_s is (p, P)-detectable for a language model $\mathcal{L}$, for some expression P and $p \in (0, 0.5]$, if there exists a detector $D_s : \mathcal{T}^n \times \mathcal{T}^n \to \{0, 1\}$ that runs in $P(n)$ time such that the probability that it correctly classifies a randomly ordered pair of sequences of length n generated by $\mathcal{L}$ and by $W_s(\mathcal{L})$ with probability at least $\frac{1}{2} + p$.

This definition does not capture the entirely of the concept of watermark detectability, as a watermark should also be distinguishable from a sequence generated by another language model or process, but we leave the development of a more general definition of detectability to future work.

A larger problem is that the definition above permits trivial watermarks that destroy the input logit distribution. For example, consider the watermark $W_0(t, \ell) = (1, 0, 0 \cdots 0)$. For nontrivial language models L, this watermark is trivially detectable – all text output by it will appear the same. However, it is also useless.

Ideally, a watermark should not materially change the quality of the generated LLM's text. While quality is subjective, consider the following: if it is impossible to efficiently tell watermarked texts apart from non-watermarked text, then the watermark must not affect any perceivable metric of quality. We use this observation to craft the following definition:

Definition 3.7 (Strong Quality-Preserving Watermark)**.** Let W_s be a watermark where s has length m and Ł an LLM, p, q polynomials, and $n \leq p(m)$. Consider an $p(m)$-time adversary A which takes in two texts and classifies them as watermarked and not watermarked. W_s is quality-preserving if, for all Ł, m, n, A, p, q, A is correct at most $1/2 + 1/q(m)$ of the time when given a texts of length n generated by Ł and W_s(Ł), over the randomness of LLM generation and the choice of s uniformly randomly.

This definition is sufficient for a watermark to preserve the quality of a text. None of the existing watermarks discussed in the paper satisfy this standard of quality preservation, despite being quality-preserving in practice.

For a deterministic language model, strong quality-preservation and detectability are incompatible: the only way to be strong quality-preserving is to almost never modify the output, in which case the watermark is undetectable. Nevertheless, a watermark exists that is quality-preserving for all non-deterministic language models.

Theorem 3.8. *Assuming the existence of one-way functions, there exists a watermark which is strong quality-preserving for all language models that are always non-deterministic.*

Proof. Let f_s be a pseudorandom function (these exist if one-way functions exist). Consider the watermark W_s which generates a pseudorandom number $r \in [0, 1]$ by applying f_s to the previous tokens. The next token is then chosen by using r to select the next token from the LLM logits.

This is strong quality-preserving, as otherwise an adversary that could distinguish a watermarked from unwatermarked

language model could be used to distinguish f_s from a random function. Since W_s is deterministic for any given seed, it can be detected by rerunning the watermarked LLM and observing if it returns the same output. □

This watermark is very detectable, and perfectly preserves quality, though it fails the desiderataum that watermarks should still be detectable after the text is modified slightly. We will not formalize this desiderataum in this paper.

Though other watermarks are less sensitive to changes to the text, all known watermarks are vulnerable to some attacks which preserve output quality while preventing the detector from detecting it. As such, watermarkers might have an incentive to hide their watermarking algorithms or even the fact that they use a watermark.

Definition 3.9 (Measurable watermark over L)**.** Consider the following game played by an efficient (polynomial time in $|s|$) distinguisher A who has blasck box access to a language model that is potentially watermarked. he adversary wins if it is able to guess which text is the watermarked one. The watermark is *measurable* if and only if there exists an adversary A such that the probability of the adversary winning is at least $1/2 + 1/p(|s|)$ where p is some polynomial.

Detectability conflicts with unmeasurability; the easier it is for a detector with access to the seed to detect the watermark, the easier it is for a detector without access to the seed to detect it. This conflict is provable. In fact, for watermarks with a given detectability, there is a single adversary that can detect all of them.

Theorem 3.10. *Let R be a sequence model which always samples uniformly from $\{0,1\}$.*

There exists an adversary A such that, for any watermark W and detector $D : \{0,1\}^n \to \{0,1\}$ such that $\Pr[D(R^n) = 0]/2 + \Pr[D(W(R)^n) = 1]/2 \geq \frac{1}{2} + p$, A can computationally distinguish R and $W(R)$ in $O(n \log(\frac{n}{p\Delta} \log(\frac{1}{\Delta})))$

Proof. Consider the distribution of next-token probabilities for 0 in a text.

If the detector correctly distinguishes between positive and negative distributions is at most $\frac{1}{2} + p$, we can use the bound from (Sadasivan et al., 2023) to bound the total variation distance between R^n and $W(R)^n$:

$$\frac{1}{2} + p \leq \frac{1}{2} + \text{TV}(R^n, W(R)^n) - \frac{\text{TV}(R^n, W(R)^n}{2}$$

$$\Rightarrow \text{TV}(R^n, W(R)^n) \geq 1 - \sqrt{1-p}$$

Imagine that the adversary has the ability to not just sample the generator, but get its probabilities for the next token. Consider the average variation distance from uniform of the next token from the watermarked generator, over a uniformly random in $(0, ..., n-1)$ number of uniformly randomly generated previous tokens. By the subadditivity of the total variation measure, the average variation distance must be at least $\frac{1-\sqrt{1-p}}{n}$. Since it is bounded in $[0, 0.5]$, at least $\frac{2-2\sqrt{1-p}}{n}$ of the sampled probabilities must be at least $\frac{1-\sqrt{1-p}}{n}$. To ensure that with probability $1 - \Delta/2$ the adversary has sampled at least one such probability, it must take at least m samples, with $(1 - \frac{2-2\sqrt{1-p}}{n})^m \leq \Delta/2$, and so

$$m \leq \frac{\log(\Delta/2)}{\log(1 - \frac{2-2\sqrt{1-p}}{n})}$$

Since the adversary cannot sample probabilities, it must repeatedly sample a certain token. Let k be the number of samples it takes from each particular token, and let the adversary classify the sample depending on whether the proportion of '0' generations differs from $1/2$ by at least q. Using the two-sided Chernoff bounds, we can get the probability of any particular sample from the uniform generator being misclassified. We then use union bounds to get the total probability of the uniform generator being misclassified, and use it to get bounds on k and q:

$$2m \exp(-k((0.5+q)\log(1+2q) + (0.5-q)\log(1-2q))) \leq \Delta$$

We use a similar method to get the probability that a token with variation from uniform $v = \frac{1-\sqrt{1-p}}{n}$ avoids detection.

$$\exp(-k((0.5+q)\log(\frac{0.5+q}{0.5+v}) + (0.5-q)\log(\frac{0.5-q}{0.5-v}))) \leq \Delta/2$$

To get the q which requires the fewest samples, we set these bounds to be equal. Doing this, we get a detection algorithm polynomial that takes $O(n \log(\frac{n}{pv\Delta} \log(\frac{1}{\Delta})))$ which correctly classifies both the random and any watermarked generator with probability at leat $1 - \Delta$.

□

This theorem shows that, when the natural distribution of a language model on some prompt is known, it cannot be watermarked undetectably.

4. Visualizing Language Model Output and Probability Distributions

Given that a watermark must always perturb the vocabulary logits at any given decoding step, as explained in Section 3, a watermark can be characterized and thus detected by how it affects the logits distribution of the underlying LLM. As such, our algorithms for watermark detection are centered

heavily on analyzing shifts in language model output and logit/probability distributions. Therefore, it is critical to first gain intuition for how these distributions usually behave.

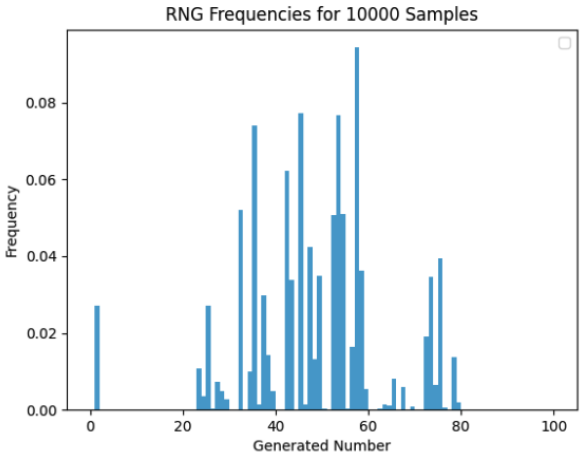
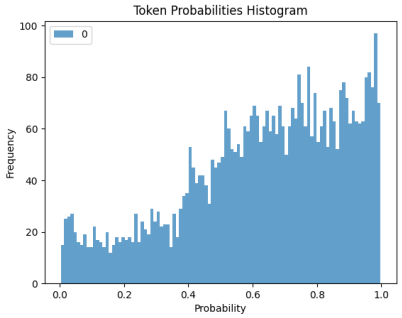


Figure 1. Example of a 10,000-sample RNG distribution generated by Alpaca-LoRA. Clearly, the distribution is far from uniform and exhibits idiosyncratic generations resulting from the training set.

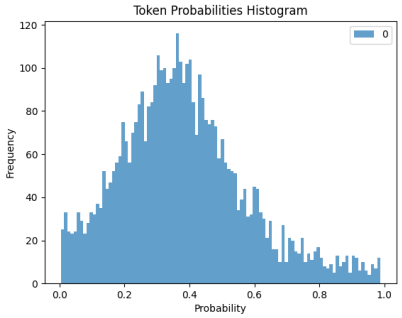
4.1. Random Bit Generation

The simplest case is analyzing language models as random bit generators. In the best case, models trained with sufficient compute would generate bits uniformly at random when prompted, and so the detection mechanism in Theorem 3.10 could be used. We analyzed all the language models available from the OpenAI API that provided logits. For each model, we prompted it with "Choose two digits, and generate a uniformly random string of those digits. Previous digits should have no influence on future digits:" followed by a sequence of 20 '0's and '1's, and generated 100 tokens. For each token, if ' 0' and ' 1' were both in the top 5 logits as provided by the OpenAI API, the probability of a generation of '0' given '0' or '1' being generated was stored. This was repeated 50 times. A graph of the probabilities for the models is in Figure 2 [1]. Unsurprisingly, these distributions failed tests for normality. Surprisingly, the qualitative distribution of the probabilities is entirely different. Ada (2a) results have the likelihood roughly monotonically increasing, babbage (2b) looks roughly like a truncated normal distribution, curie (2c) has the likelihood mass concentrated around 0 and 1, and davinci (2d) is trimodal with peaks around 0, 0.5, and 1.

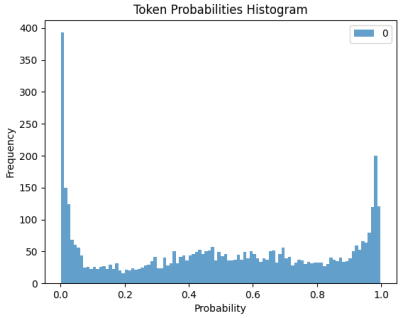
[1]The experiment was also run for instruction fine-tuned models, but they had an abysmal rate of actually generated ' 0's and ' 1's as opposed to other tokens, so we didn't include the data.



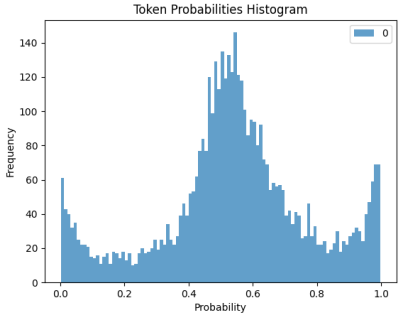
(a) Ada outputs.



(b) Babbage outputs.

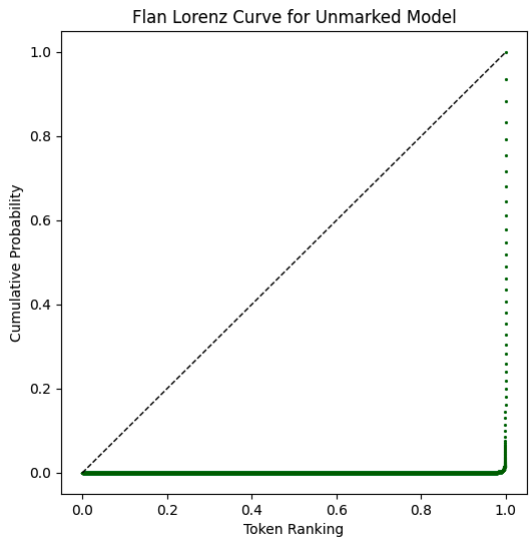


(c) Curie outputs.

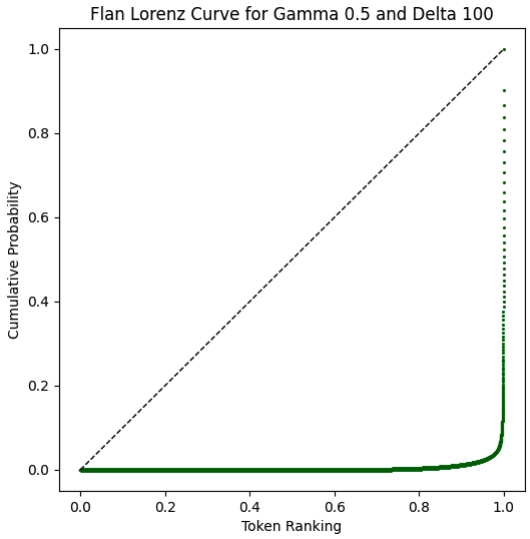


(d) Davinci outputs.

Figure 2. Comparison of OpenAI engine behavior on a simple random bit generation task. The x-axis shows the retrieved probability of generating 0, and the y-axis shows the frequency over multiple generations.



(a) Lorenz curve for an unmarked Flan-T5-XXL language model. Most of the probability mass in concentrated in a few top tokens, as visualized by the sharp spike towards the right of the Lorenz curve.



(b) Lorenz curve for a Flan-T5-XXL model affected by a Kirchenbauer watermark with $\gamma = 0.5$ and $\delta = 100$. Notice that the Lorenz curve is smoother under this setting, due to the δ application on low-probability tokens.

Figure 3. Examples of ranked probability Lorenz curves of the first token generated by Flan-T5-XXL under different Kirchenbauer watermarking strengths. The dashed line represents a perfectly uniform distribution. In both cases, the majority of the probability mass in both settings is concentrated in the top few tokens.

4.2. Ranked Probability Lorenz Curves

Inspired by tools from econometrics, we borrow the Lorenz curve as a means of understanding language model behavior. To do so, we examine the output token probabilities of a model and construct *ranked probability Lorenz curves*. Specifically, the x-axis of a ranked probability Lorenz curve lists the tokens, sorted from lowest to highest probability, and the y-axis of the curve displays the probabilities of each token. Due to the sorted construction of the x-axis, the ranked token Lorenz curve is monotonically increasing. Figure 3 displays an example of a Lorenz curve.

The Lorenz curve is an effective for understanding the effects of a Kirchenbauer watermark. Recall that such a watermark adds a constant term δ to a randomly selected subset of green list token logits. In the ranked token Lorenz curve, this is notably reflected by a smoothing effect, as seen on the right of Figure 3. This indicates that a portion of low-probability tokens have experienced a δ-increase.

To rigorize this notion of smoothness, one can compute the Gini coefficient G of the Lorenz curve:

$$G = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} |x_i - x_j|}{2n^2 \overline{x}}$$

Here x_i, x_j are the probabilities of i-th and j-th tokens on the curve, indexed by the ordered ranking, and $\overline{x}$ is the average probability. Traditionally in economics, G is used to measure the inequality of a distribution. High G suggests more inequality, reflected in unmarked language model distributions, while low G suggests less inequality and thus a smoother distribution, hinting at the existence of a watermark.

Recovering Logits from Sampling In practice, exact logits may not be available for analysis, for example when interacting with ChatGPT. In this case, we approximate token probabilities by sampling a large number of tokens from a language model, and calculating empirical probabilities.

4.3. Random Number Generation

In the case of a publicly hosted API, oftentimes logits data is not directly accessible. As a suitable approximation, we instead consider the distribution of tokens from a small subset of the original vocabulary. This enables us to analyze the shifting behavior of a LLM before and applying a watermark, without requiring access to output logits.

Specifically, we treat LLMs as random number generators, asking them to generate integers from 1 to 100, inclusive. Figure 1 displays an example 10,000-sample distribution from Alpaca-LoRA using the following prompt:

```
"""Below is an instruction that
```

```
describes a task. Write a response that
appropriately completes the request.

### Instruction:
Generate a random number between
1 and 100.

### Response:"""
```

While this is a natural task to constrain the output token set of the model, it is certainly not the only task that would do so. For example, asking a LLM to provide a synonym for a given input word, such as "intelligent", that starts with a specific letter, such as "c", would also severely restrict the output distribution to a subset resembling something like {"clever", "canny", "crafty", "calculating", "cunning"}.

A key benefit of the random number generation task over other alternatives, however, is that the output space for any model is fairly consistent between models, generating integers between 1 and 100, regardless of model capacity. While the distribution of numbers is certainly expected to change across models, the range of outputs is relatively more stable.

4.4. Differing Model Capabilities

There is a large ecosystem of models to choose from to perform our analysis, with each model possessing unique behaviors. For our task, we are most concerned with our models' ability to consistently generate random numbers. Besides parameter size, two critical factors affect a model's ability to generate random numbers: tokenization method and instruction finetuning.

In particular, on the axis of tokenization, digit tokenization versus traditional tokenization methods such as BPE dramatically affect the output distribution of generated numbers. As a stark example, in general text data, the number 12 may not appear as often as the number 99, from texts such as $11.99. Thus, models using BPE tokenization may produce outputs heavily biased towards 99.

Instruction finetuned models also tend to follow instructions more directly, by the nature of the training task. Specifically, they are capable of generating numbers more "randomly" than their non-instruction counterparts. Considering these factors, we perform experiments on Flan-T5-XXL and Alpaca-LoRA, due to their strong instruction-following capabilities but differing tokenization schemes.

5. Baseline Algorithms for Watermark Detection

Here, we introduce three simple watermark detection algorithms based on analyzing exact and approximate probability distributions. Critically, our algorithms do not require any access to information governing the underlying watermark generation procedure, such as a hash function or random number generator. We hope these algorithms can serve as sound baselines for future work in this field.

Our three proposed algorithms vary in their access to exact versus sampled logits, watermark generalizability (can only detect certain types of watermarks or all watermarks), and statistical interpretability (i.e. is there a test versus no test). Depending on the desired result (efficient computation, interpretabtle test statistic, access to logits, robustness to random shifts in the data), a different algorithm will be ideal.

5.1. Measuring Divergence of RNG Distributions

The first algorithm is centered on the simple idea of measuring divergence in "random" number distributions generated by a LLM, as alluded to in Section 3.2. In particular, we make use of the Two-sample Kolmogorov–Smirnov test to determine whether an empirical random number distribution of a *watermarked* LLM shifts from an empirical random number distribution of an *unmarked* LLM.

Given a specific LLM, we first generate a 1000 number empirical distribution $F_{u,n}$ as described in Section 3.2 (link this later) using the *unmarked* model. We then watermark the LLM and produce an empirical distribution $F_{w,m}$ in the same fashion. We then compute the Kolmogorov-Smirnov statistic as follows:

$$D_{n,m} = \sup_x |F_{u,n}(x) - F_{w,m}(x)|$$

Here n and m are the sizes of each sample, and $n = m = 1000$ specifically in our case.

The null hypothesis is that the samples are drawn from the same distribution, i.e.:

H_0: $F_{u,n}$ and $F_{w,m}$ are drawn from the same underlying distribution

We reject H_0 at significance level α if:

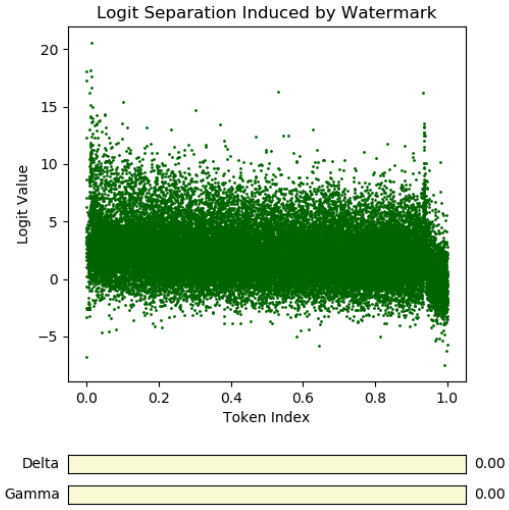
$$D_{n,m} > c(\alpha)\sqrt{\tfrac{n+m}{n \cdot m}}$$

Here $c(\alpha) = \sqrt{-\ln\left(\frac{\alpha}{2}\right) \cdot \frac{1}{2}}$.

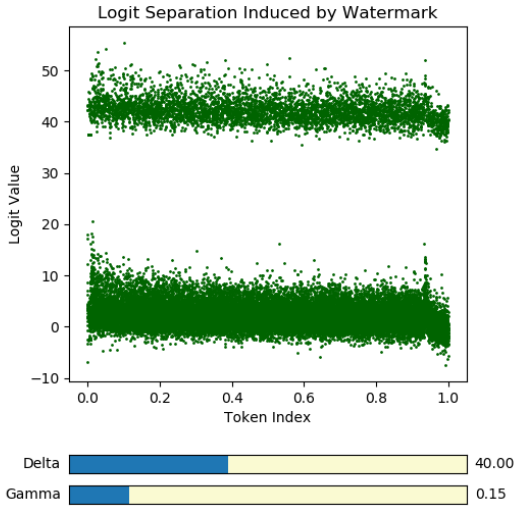
5.2. Mean Adjacent Token Differences

From the ranked token Lorenz curve introduced in Section 4.2, a natural extension is to analyze the average increase in logit value between adjacent tokens. That is, we compute:

$$\mathcal{I} = \frac{\sum_{i=1}^{n-1} \ell_{i+1} - \ell i}{n-1}$$



(a) Logit distribution produced by an unmarked Alpaca-LoRA model. The tokens are indexed by the original tokenizer ordering. While there is certainly a wide variation in logit values, there is no distinct separation.



(b) Logit distribution produced by a Alpaca-LoRA model with a $\gamma = 0.15$ and $\delta = 40$ Kirchenbauer watermark. There is a distinct logit separation of size δ into two bands, one for the original logits, and one for logits perturbed by δ. The width of the top band corresponds to γ.

Figure 4. Logit separation induced by Kirchenbauer watermarks.

Here, ℓ_i is the logit at index i on the Lorenz curve, and n is the total number of tokens in the vocabulary.

Note that for a Kirchenbauer-watermarked LLM with logit perturbation δ and green list G with proportion γ, we have an average logit increase of:

$$\begin{aligned}\mathcal{I}_{\mathcal{W}} &= \frac{\sum_{i=1}^{n-1}(\ell_{i+1} - \ell i)\mathbb{1}[i \in G]}{n-1} \\ &= \frac{\gamma(n-1)\delta + \sum_{i=1}^{n-1}(\ell_{i+1} - \ell i)}{n-1}\end{aligned}$$

Taking the difference with the average logit increase of an unmarked model, $\mathcal{I}_{\mathcal{U}}$, we have:

$$\mathcal{I}_{\mathcal{W}} - \mathcal{I}_{\mathcal{U}} = \frac{\gamma(n-1)\delta}{n-1} = \gamma\delta$$

Taking the above $I_W - I_U$ as inspiration, a simple detection procedure is to periodically compute $\mathcal{I}$ and observe how it varies over time. Notice that $I_W - I_U$ directly varies with γ and δ; that is, the strength of the watermark directly influences its detectability. For a strong watermark, variations in $\mathcal{I}$ will be obvious, while weaker watermarks will manifest subtler differences in $\mathcal{I}$.

5.3. Robustly Detecting Small-δ Watermarks

While Section 5.2 introduces a metric that will successfully detect a Kirchenbauer watermark for small δ, it is sensitive to general logit distribution perturbations introduced by other scenarios, such as routine model updates. A detection method robust to general distribution shifts should rely on shift characteristics specific to a Kirchenbauer watermark.

Notably, a Kirchenbauer watermark will induce perceptible band separations in logit space. Figure 4 shows an example of this phenomenon. Inspired by this observation, we draw an analogue between the separation of logit values into bands and the bimodality of logit frequencies. Under this reframing, testing for bimodality is equivalent to testing for the existence of a band gap.

However, though this approach is robust to other distribution shifts, it does not yet consider small-δ perturbations. To handle such situations, we introduce the δ-Amplification algorithm.

Algorithm 5.1 (δ-Amplification)**.** *Suppose we have a potentially watermarked LLM $\mathcal{L}$. We wish to detect if it is watermarked. We prompt $\mathcal{L}$ with them as follows:*

```
[Random string sampled from training
datasets]. Now write me a story:
```

Take the received logits and average them. If the averaged

Table 1. Tradeoffs between proposed watermarking detection algorithms. An ideal watermarking is not specific to Kirchenbauer and can detect **general watermarks**; does **not require access to logits**, which is common in publicly hosted models; is **sensitive to small** δ values; is **robust against other distribution shifts** not induced by watermarks; and can be **performed in a single snapshot of time** without reference to previous distributions or tests.

DETECTION METHOD	GENERAL WATERMARKS	LOGIT-FREE	δ-SENSITIVE	SHIFT-ROBUST	SINGLE-SHOT
RNG DIVERGENCE	$\surd$	$\surd$	$\surd$	$\times$	$\times$
MEAN ADJACENT	$\times$	$\times$	$\surd$	$\times$	$\surd$
δ-AMPLIFICATION	$\times$	$\times$	$\surd$	$\surd$	$\surd$

logit distribution is bimodal, $W_s(\mathcal{L})$ is watermarked. To recover the watermark parameters, we estimate δ by measuring the distance between the peaks, and γ by measuring their respective masses.

Critically, as watermarks (Kirchenbauer et al., 2023; Aaronson, 2023) only use a fixed-size previous token window (rumored to be 5-tokens for OpenAI) to determine green list indices, the green list across all prompts is the same, as every prompt ends in "Now write me a story:" suffix. Therefore the output logit distributions all experience the same δ mask.

However, the model is still influenced by earlier tokens in the prompt, and thus exhibits differing logit values across prompts. Intuitively then, averaging distributions across different prompts reduces the variation of logits, but maintains the same effect of the δ perturbation. The averaged distribution thus amplifies the effects of a small-δ watermark. Figure 5 demonstrates an example of this effect.

We test for bimodality via the Hartigan dip test (Hartigan & Hartigan, 1985). For a distribution with probability distribution function f, this test computes the largest absolute difference between f and the unimodal distribution which best approximates it.

$$D(f) = \inf_{g \in U} \sup_{x} |f(x) - g(x)|$$

Here U is the set of all unimodal distributions over x. The corresponding p-value is calculated as the probability of achieving a Dip score at least as high as D from the nearest unimodal distribution.

5.4. Tradeoffs Between Detection Algorithms

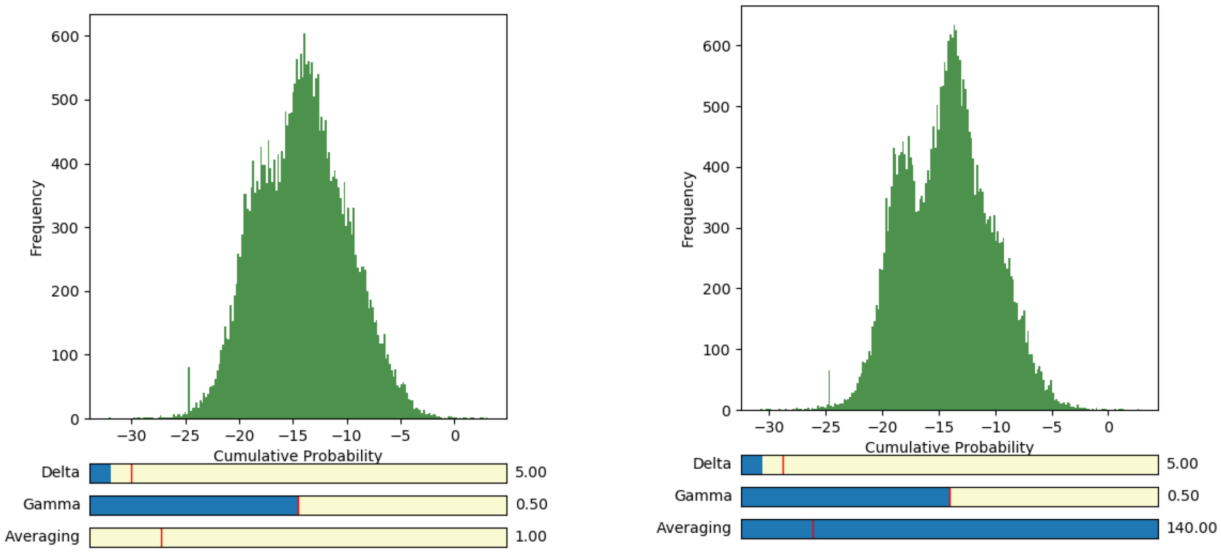
The algorithms proposed above are all effective in different senses. Section 5.1 introduced a RNG divergence approach to watermark detection that is not specific to Kirchenbauer watermarks, does not require access to logits and can thus be used directly on black-box public APIs, and is also sensitive to small δ watermarks.

Section 5.2 introduced an adjacent token metric for analyzing Kirchenbauer watermarks. It is sensitive to small δ watermarks and can also be performed in a single shot. That is, it does not require comparison in behavior between a watermarked model and an unmarked model, as is the case for RNG divergence.

Finally, we extended this approach in Section 5.3 to robustly handle small-δ watermarks, while preserving the single-shot criteria. Moreover, the δ-Amplification approach lent itself

Table 2. Dip and p-values for watermarked Alpaca-LoRA logit distributions at small δ when using prompt prefixes from Pile and OpenWebText.

δ	P-VALUE (OWT & PILE)	DIP (OWT & PILE)	p (PILE)	DIP (PILE)
0	0.886	0.0017	0.908	0.0016
1	1.0	0.00094	0.999	0.00097
2	0.991	0.0013	1.0	0.00092
3	0.900	0.0016	1.0	0.00093
4	0.204	0.0025	1.0	0.00093
5	0.0	0.00497	1.0	0.00093
6	0.0	0.0087	0.947	0.0015
7	0.0	0.012	0.0	0.0048
8	0.0	0.017	0.0	0.013
9	0.0	0.023	0.0	0.025
10	0.0	0.033	0.0	0.037



(a) Distribution of logit values prior to δ-Amplification.

(b) Distribution of logit values after δ-Amplification across 140 prompts.

Figure 5. Distribution of Alpaca-LoRA logit values before and after δ-Amplification for the "Now write me a story:" prompt. The x-axis is the logit value and y-axis is the frequency. Notice that bimodality emerges after amplification, enabling watermark detection.

nicely to statistical testing, specifically of bimodality.

Each method has merit depending on the specific detection setting, but will also sacrifice certain desiderata. Table 1 summarizes these tradeoffs.

5.5. Monitoring

To detect watermarks in publicly hosted models, we set up monitoring scripts with our algorithms that periodically query these models and compute the relevant tests and metrics. Specifically, we are most interested in detecting watermarks in OpenAI models that are *not* API-accessible, as we believe that UI-based versions of these models will be most susceptible to dishonest usage (e.g. students cheating will primarily use ChatGPT and not a Python script accessing API). As such, we also set up an agent to interact and monitor the UI-based version of these models.

6. Results

Table 3 displays the p-values resulting from the Kolmogorov-Test method detailed in Section 5.1. For each model and watermark strength, the method is averaged across 30 instances of 1000-sample distributions. We perform a test between each of the 30 instances and an unmarked model. The p-values are highest when comparing an unmarked distribution against an unmarked distribution, as expected. Notably, the p-values are extremely low for a majority of watermark strengths for both Flan-T5-XXL and Alpaca-LoRA. Under our framework, any p-value less than 0.05 would be considered watermarked.

The results of δ-Amplification method and corresponding bimodality test are in Table 2. Concretely, we sample a diverse range of prompt prefixes from Pile and OpenWebText via HuggingFace datasets and run tests on the logit value distributions from these generations. Notably, diversity in prompt prefix task and content enables orthogonal variance in δ, thus most effectively eliminating logit variance post-averaging. Due to limited compute budgets, we only test on Alpaca-LoRA.

In particular, we observe that at $\delta \geq 5$, our method produces p-values less than 0.05, thus successfully detecting the presence of a watermark. Increasing the number prompts also increases detection potency. Namely, averaging logit distributions only across Pile prompts requires $\delta \geq 7$ for detection, while averaging across both Pile and OpenWebText prompts only requires $\delta \geq 5$ for detection.

As such, both algorithms serve as strong baselines for watermark detection.

7. Conclusion

In this work, we develop a theoretical framework for understanding the watermark detection problem in large language models. We then provide three black-box baseline algorithms – measuring divergence of RNG distributions, mean

Table 3. Kolmogorov-Smirnov test results on 1000-sample "RNG" distributions from models watermarked at varying strengths. A test is performed between 30 distributions generated from the model in each row against a random distribution generated from an unmarked model. The reported p-values are averaged across these 30 samples. Note that the first row of each section is result between generations from a single unmarked model.

MODEL	GAMMA	DELTA	P-VALUE
FLAN-T5-XXL	0	0	0.80
FLAN-T5-XXL	0.1	1	3.54E-7
FLAN-T5-XXL	0.1	10	1.22E-9
FLAN-T5-XXL	0.1	50	6.75E-7
FLAN-T5-XXL	0.1	100	8.11E-8
FLAN-T5-XXL	0.25	1	0.002
FLAN-T5-XXL	0.25	10	6.47E-9
FLAN-T5-XXL	0.25	50	2.59E-7
FLAN-T5-XXL	0.25	100	1.33E-6
FLAN-T5-XXL	0.5	1	0.00024
FLAN-T5-XXL	0.5	10	0.057
FLAN-T5-XXL	0.5	50	0.054
FLAN-T5-XXL	0.5	100	0.054
FLAN-T5-XXL	0.75	1	0.42
FLAN-T5-XXL	0.75	10	0.22
FLAN-T5-XXL	0.75	50	0.34
FLAN-T5-XXL	0.75	100	0.23
ALPACA-LORA	0	0	0.63
ALPACA-LORA	0.1	1	1.40E-10
ALPACA-LORA	0.1	10	4.31E-37
ALPACA-LORA	0.1	50	4.31E-36
ALPACA-LORA	0.1	100	1.48E-35
ALPACA-LORA	0.25	1	3.10E-13
ALPACA-LORA	0.25	10	1.93E-10
ALPACA-LORA	0.25	50	4.52E-14
ALPACA-LORA	0.25	100	2.14E-11
ALPACA-LORA	0.5	1	4.17E-13
ALPACA-LORA	0.5	10	0.0089
ALPACA-LORA	0.5	50	0.066
ALPACA-LORA	0.5	100	0.06
ALPACA-LORA	0.75	1	0.00015
ALPACA-LORA	0.75	10	0.52
ALPACA-LORA	0.75	50	0.40
ALPACA-LORA	0.75	100	0.48

adjacent token differences in logits, and δ-Amplification – for detecting watermarks, which all fundamentally rely on the analysis of the distributions of model outputs and probabilities. Each algorithm trades off in different practical aspects, including detection generalizability, logit-free analysis, sensitivity to watermarks, robustness against general distribution shifts, and single-shot testing. Ultimately, we monitor publicly hosted models in an attempt to detect watermarks. Since we are the first to consider the problem of detecting watermarks in large language models, we hope that our framework and baselines serve as strong foundations for the future work in the community.

References

Aaronson, S. My ai safety projects at openai. Talk given to the Harvard AI Safety Team, March 2023.

Abdelnabi, S. and Fritz, M. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 121–140. IEEE, 2021.

Hartigan, J. A. and Hartigan, P. M. The dip test of unimodality. *The annals of Statistics*, pp. 70–84, 1985.

Kirchenbauer, J., Geiping, J., Wen, Y., Katz, J., Miers, I., and Goldstein, T. A watermark for large language models. *arXiv preprint arXiv:2301.10226*, 2023.

Kudo, T. and Richardson, J. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D., and Finn, C. Detectgpt: Zero-shot machine-generated text detection using probability curvature. *arXiv preprint arXiv:2301.11305*, 2023.

Sadasivan, V. S., Kumar, A., Balasubramanian, S., Wang, W., and Feizi, S. Can ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*, 2023.

Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. In *54th Annual Meeting of the Association for Computational Linguistics*, pp. 1715–1725. Association for Computational Linguistics (ACL), 2016.

Solaiman, I., Brundage, M., Clark, J., Askell, A., Herbert-Voss, A., Wu, J., Radford, A., Krueger, G., Kim, J. W., Kreps, S., et al. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*, 2019.

Topkara, M., Taskiran, C. M., and Delp III, E. J. Natural language watermarking. In *Security, Steganography, and Watermarking of Multimedia Contents VII*, volume 5681, pp. 441–452. SPIE, 2005.

Yang, Y., Zha, L., Zhang, Z., and Wen, J. An overview of text steganalysis. In *The International Conference on Image, Vision and Intelligent Systems (ICIVIS 2021)*, pp. 933–943. Springer, 2022.