A Watermark for Large Language Models

John Kirchenbauer * 1 Jonas Geiping * 1 Yuxin Wen 1 Jonathan Katz 1 Ian Miers 1 Tom Goldstein 1

Abstract

Potential harms of large language models can be mitigated by watermarking model output, i.e., embedding signals into generated text that are invisible to humans but algorithmically detectable from a short span of tokens. We propose a watermarking framework for proprietary language models. The watermark can be embedded with negligible impact on text quality, and can be detected using an efficient open-source algorithm without access to the language model API or parameters. The watermark works by selecting a randomized set of “green” tokens before a word is generated, and then softly promoting use of green tokens during sampling. We propose a statistical test for detecting the watermark with interpretable p-values, and derive an information-theoretic framework for analyzing the sensitivity of the watermark. We test the watermark using a multi-billion parameter model from the Open Pretrained Transformer (OPT) family, and discuss robustness and security.

1. Introduction

Large language models (LLMs), such as the recently developed ChatGPT, can write documents, create executable code, and answer questions, often with human-like capabilities (Schulman et al., 2022). As these systems become more pervasive, there is increasing risk that they may be used for malicious purposes (Bergman et al., 2022; Mirsky et al., 2023). These include social engineering and election manipulation campaigns that exploit automated bots on social media platforms, creation of fake news and web content, and use of AI systems for cheating on academic writing and coding assignments. Furthermore, the proliferation of synthetic data on the web complicates future dataset creation efforts, as synthetic data is often inferior to human content and must be detected and excluded before model training.

In this work, we study watermarking of language model output. A watermark is a hidden pattern in text that is imperceptible to humans, while making the text algorithmically identifiable as synthetic. We propose an efficient watermark that makes synthetic text detectable from short spans of tokens (as few as 25 tokens), while false-positives (where human text is marked as machine-generated) are statistically improbable. The watermark detection algorithm can be

Figure 1. Outputs of a language model, both with and without the application of a watermark. The watermarked text, if written by a human, is expected to contain 9 “green” tokens, yet it contains 28. The probability of this happening by random chance is $\approx 6 \times 10^{-14}$, leaving us extremely certain that this text is machine generated. Words are marked with their respective colors. The model is OPT-6.7B using multinomial sampling. Watermark parameters are $\gamma, \delta = (0.25, 2)$. The prompt is the whole blue paragraph marked below.

(Radford et al., 2022). For many reasons, the ability to detect and audit the usage of machine-generated text becomes a key principle of harm reduction for large language models (Bender et al., 2021; Crothers et al., 2022; Grinbaum & Adomaitis, 2022).

Code is available at https://www.github.com/jwkirchenbauer/lm-watermarking. * Equal contribution. 1 University of Maryland. Correspondence to: John Kirchenbauer <jkirchen@umd.edu>.

made public, enabling third parties (e.g., social media platforms) to run it themselves, or it can be kept private and run behind an API. We seek a watermark with the following properties:

- The watermark can be algorithmically detected without any knowledge of the model parameters or access to the language model API. This property allows the detection algorithm to be open sourced even when the model is not. This also makes detection cheap and fast because the LLM does not need to be loaded or run.
- Watermarked text can be generated using a standard language model without re-training.
- The watermark is detectable from only a contiguous portion of the generated text. This way, the watermark remains detectable when only a slice of the generation is used to create a larger document.
- The watermark cannot be removed without modifying a significant fraction of the generated tokens.
- We can compute a rigorous statistical measure of confidence that the watermark has been detected.

1.1. Notation & Language model basics

Language models have a “vocabulary” \( V \) containing words or word fragments known as “tokens.” Typical vocabularies contain \(|V| = 50,000 \) tokens or more (Radford et al., 2019; Liu et al., 2019). Consider a sequence of \( T \) tokens \( \{s_t\} \in V^T \). Entries with negative indices, \( s_{-N_p}, \ldots, s_{-1} \), represent a “prompt” of length \( N_p \) and \( s_0, \ldots, s_T \) are tokens generated by an AI system in response to the prompt.

A language model (LM) for next word prediction, is a function \( f \), often parameterized by a neural network, that accepts as input a sequence of known tokens \( s_{-N_p}, \ldots, s_{t-1} \), which contains a prompt and the first \( t-1 \) tokens already produced by the language model, and then outputs a vector of \(|V|\) logits, one for each word in the vocabulary. These logits are then passed through a softmax operator to convert them into a discrete probability distribution over the vocabulary. The next token at position \( t \) is then sampled from this distribution using either standard multinomial sampling, or greedy sampling (greedy decoding) of the single most likely next token. Additionally, a procedure such as beam search can be employed to consider multiple possible sequences before selecting the one with the overall highest score.

1.2. A caveat: The difficulty of watermarking low-entropy sequences

Consider the following two sequences of tokens, with prompts in red:

- The quick brown fox jumps over the lazy dog

The method works by generating a pseudo-random red list of tokens that are barred from appearing as \( s_t \). The red list generator is seeded with the prior token \( s_{t-1} \), enabling the red list to be reproduced later without access to the entire generated sequence.

Detecting the watermark. While producing watermarked text requires access to the language model, detecting the watermark does not. A third party with knowledge of the hash function and random number generator can re-produce the red list for each token and count how many times the red list rule is violated. We can detect the watermark by testing

\[
\text{for}(i=0; i<n; i++) \text{sum}+=\text{array}[i]
\]

Were they produced by a human or by a language model? Determining this is fundamentally hard because these sequences have low entropy; the first few tokens strongly determine the following tokens.

Low entropy text creates two problems for watermarking. First, both humans and machines provide similar if not identical completions for low entropy prompts, making it impossible to discern between them. Second, it is difficult to watermark low entropy text, as any changes to the choice of tokens may result in high perplexity, unexpected tokens that degrade the quality of the text. Later, we rigorously define sentence entropy, and analyze its impact on watermark detection.

2. A simple proof of concept

We start out by describing a simple “hard” red list watermark in Algorithm 1 that is easy to analyze, easy to detect and hard to remove. The simplicity of this approach comes at the cost of poor generation quality on low entropy sequences. We will discuss more sophisticated strategies later.

**Algorithm 1** Text Generation with Hard Red List

Input: prompt, \( s_{-N_p}, \ldots, s_{-1} \)

for \( t = 0, 1, \ldots \) do

1. Apply the language model to prior tokens \( s_{-N_p}, \ldots, s_{t-1} \) to get a probability vector \( p(t) \) over the vocabulary.

2. Compute a hash of token \( s_{t-1} \), and use it to seed a random number generator.

3. Using this seed, randomly partition the vocabulary into a “green list” \( G \) and a “red list” \( R \) of equal size.

4. Sample \( s(t) \) from \( G \), never generating any token in the red list.

end for
we reject the null hypothesis, 

\[ H_0: \text{The text sequence is generated with no knowledge of the red list rule.} \] (1)

Because the red list is chosen at random, a natural writer is expected to violate the red list rule with half of their tokens, while the watermarked model produces no violations. The probability that a natural source produces \( T \) tokens without violating the red list rule is only \( 1/2^T \), which is vanishingly small even for short text fragments with a dozen words. This enables detection of the watermark (rejection of \( H_0 \)) for, e.g., a synthetic tweet.

A more robust detection approach uses a one proportion z-test to evaluate the null hypothesis. If the null hypothesis is true, then the number of green list tokens, denoted \( |s|_G \), has expected value \( T/2 \) and variance \( T/4 \). The \( z \)-statistic for this test is

\[ z = 2(\frac{|s|_G - T/2}{\sqrt{T}}). \] (2)

We reject the null hypothesis and detect the watermark if \( z \) is above a chosen threshold. Suppose we choose to reject the null hypothesis if \( z > 4 \). In this case, the probability of a false positive is \( 3 \times 10^{-5} \), which is the one-sided p-value corresponding to \( z > 4 \). At the same time, we will detect any watermarked sequence with 16 or more tokens (the minimum value of \( T \) that produces \( z = 4 \) when \( |s|_G = T \)).

**How hard is it to remove the watermark?** The use of the one proportion z-test makes removal of the watermark difficult. Consider the case of a watermarked sequence of length \( T = 1000 \). Suppose an adversary modifies 200 tokens in the sequence to add red list words and scrub the watermark. A modified token at position \( t \) can violate the red list rule at position \( t \). Furthermore, the value of \( s_t \) determines the red list for token \( s_{t+1} \), and a maximally adversarial choice of \( s_t \) will put \( s_{t+1} \) in violation of the red list rule as well. For this reason, 200 token flips can create at most 400 violations of the red list rule. Unfortunately for the attacker, this maximally adversarial sequence with 600 remaining green list tokens still produces a z-statistic of \( 2(600 - 1000/2)/\sqrt{1000} \approx 6.3 \), and a p-value of \( \approx 10^{-10} \), leaving the watermark readily detectable with extremely high confidence. In general, removing the watermark of a long sequence requires modifying roughly one quarter of the tokens or more.

Note the analysis above assumes the attacker has complete knowledge of the watermark, and each selected token is maximally adversarial (which likely has a negative impact on quality). Without knowledge of the watermark algorithm, each flipped token has only a 50% chance of being in the red list, as does the adjacent token. In this case, the attacker above only creates 200 red list words (in expectation) by modifying 200 tokens. Methods for keeping the watermark algorithm secret but available via API are discussed in Section 5.

**Drawbacks of the hard red list rule.** The hard red list rule handles low entropy sequences in a simple way; it prevents the language model from producing them. For example, the token “Barack” is almost deterministically followed by “Obama” in many text datasets, yet “Obama” may be disallowed by the red list.

A better behavior is to use a “soft” watermarking rule that is only active for high-entropy text that can be imperceptibly watermarked. As long as low-entropy sequences are wrapped inside a passage with enough total entropy, the passage will still easily trigger a watermark detector, solving the problem described in Section 1.2. Further, one can combine the watermark with a beam search decoder that “irons-in” the watermark. By searching the hypothesis space of likely token sequences, candidates sequences with a high density of green list tokens are found, resulting in a high strength watermark with minimal perplexity cost.

### 3. A more sophisticated watermark

We now discuss the “soft” watermark that promotes the use of the green list for high entropy tokens when many good choices are available, while having little impact on the choice of low-entropy tokens that are nearly deterministic.

To derive this watermark, we examine what happens in the language model just before it produces a probability vector. The last layer of the language model outputs a vector of logits \( l^{(t)} \). These logits get converted into a probability vector \( p^{(t)} \) using the softmax operator

\[ p^{(t)}_k = \exp(l^{(t)}_k) / \sum_i \exp(l^{(t)}_i). \]

Rather than strictly prohibiting the red list tokens, Algorithm 2 adds a constant \( \delta \) to the logits of the green list tokens.

The soft red list rule adaptively enforces the watermark in situations where doing so will have little impact on quality, while almost ignoring the watermark rule in the low entropy case where there is a clear and unique choice of the “best” word. A highly likely word with \( p^{(t)}_k \approx 1 \) has a much larger logit than other candidates, and this will remain the largest regardless of whether it is in the red list. But when the entropy is high, there are many comparably large logits to choose from, and the \( \delta \) rule has a large impact on the sampling distribution, strongly biasing the output towards the green list.
Algorithm 2 Text Generation with Soft Red List

**Input:** prompt, \( s(-N_p) \ldots s(-1) \),
- green list size, \( \gamma \in (0, 1) \),
- hardness parameter, \( \delta > 0 \)

for \( t = 0, 1, \ldots \) do
1. Apply the language model to prior tokens \( s(-N_p) \ldots s(t-1) \) to get a logit vector \( l^{(t)} \) over the vocabulary.
2. Compute a hash of token \( s(t-1) \), and use it to seed a random number generator.
3. Using this random number generator, randomly partition the vocabulary into a “green list” \( G \) of size \( \gamma |V| \), and a “red list” \( R \) of size \( (1 - \gamma)|V| \).
4. Add \( \delta \) to each green list logit. Apply the softmax operator to these modified logits to get a probability distribution over the vocabulary.
   \[
   \hat{p}^{(t)}_k = \frac{\exp(l^{(t)}_k + \delta)}{\sum_{l \in G} \exp(l^{(t)}_l + \delta) + \sum_{l \in R} \exp(l^{(t)}_l) + \sum_{l \in G} \exp(l^{(t)}_l + \delta)}, \quad k \in G
   \]
   \[
   \hat{p}^{(t)}_k = \frac{\exp(l^{(t)}_k + \delta)}{\sum_{l \in R} \exp(l^{(t)}_l) + \sum_{l \in G} \exp(l^{(t)}_l + \delta)}, \quad k \in R.
   \]
5. Sample the next token, \( s^{(t)} \), using the watermarked distribution \( \hat{p}^{(t)} \).
end for

3.1. Detecting the soft watermark

The process for detecting the soft watermark is identical to that for the hard watermark. We assume the null hypothesis (1) and compute a z-statistic using Equation (2). We reject the null hypothesis and detect the watermark if \( z \) is greater than a threshold. For arbitrary \( \gamma \) we have
\[
z = (|s|_G - \gamma T) / \sqrt{T(1 - \gamma)}.
\] (3)

Consider again the case in which we detect the watermark for \( z > 4 \). Just like in the case of the hard watermark, we get false positives with rate \( 3 \times 10^{-5} \). In the case of the hard watermark, we could detect any watermarked sequence of length 16 tokens or more, regardless of the properties of the text. However, in the case of the soft watermark our ability to detect synthetic text depends on the entropy of the sequence. High entropy sequences are detected with relatively few tokens, while low entropy sequences require more tokens for detection. Below, we rigorously analyze the detection sensitivity of the soft watermark, and its dependence on entropy.

4. Analysis of the soft watermark

In this section, we examine the expected number of green list tokens used by a watermarked language model and analyze the dependence of this quantity on the entropy of a generated text fragment. Our analysis assumes the red list is sampled uniformly at random. This is a deviation from the method used in practice, which generates red lists using a pseudo-random number generator seeded with previous tokens. The consequences of pseudo-random sampling are explored in Section 5. We analyze the case in which text is generated by multinomial random sampling. In our experiments, we consider two more sampling schemes, greedy decoding and beam search.

We need a definition of entropy that is appropriate for our analysis. The strength of our watermark is weak when the distribution over tokens has a large “spike” concentrated on one or several tokens. We define the following type of entropy to quantify this phenomenon.

**Definition 4.1.** Given a discrete probability vector \( p \) and a scalar \( z \), we define the spike entropy of \( p \) with modulus \( z \) as
\[
S(p, z) = \sum_k \frac{p_k}{1 + zp_k}.
\]

Like the classical Shannon entropy, the spike entropy is a measure of how spread out a distribution is; The spike entropy assumes its minimal value of \( \frac{1}{1 + z} \) when the entire mass of \( p \) is concentrated at a single location, and its maximal value of \( \frac{N}{N + z} \) when the mass of \( p \) is uniformly distributed. For large \( z \), the value of \( \frac{p_k}{1 + zp_k} \approx 1/z \) when \( p_k > 1/z \) and \( \approx 0 \) for \( p_k < 1/z \). For this reason, one can interpret the spike entropy as a softened measure of the number of entries in \( p \) greater than \( 1/z \).

The following theorem predicts the number of green list tokens that appear in a sequence with the watermark.

**Theorem 4.2.** Consider watermarked text sequences of \( T \) tokens. Each sequence is produced by sequentially sampling a raw probability vector \( p^{(t)} \) from the language model, sampling a random green list of size \( \gamma N \), and boosting the green list logits by \( \delta \) using Equation 4 before sampling each token. Define \( \alpha = \exp(\delta) \), and let \( |s|_G \) denote the number of green list tokens in sequence \( s \).

If a randomly generated watermarked sequence has average spike entropy at least \( S^* \), i.e.,
\[
\frac{1}{T} \sum_t S(p^{(t)}, \frac{(1 - \gamma)(\alpha - 1)}{1 + (\alpha - 1)\gamma}) \geq S^*,
\]
then the number of green list tokens in the sequence has expected value at least
\[
\mathbb{E} |s|_G \geq \frac{\gamma \alpha T}{1 + (\alpha - 1)\gamma} S^*.
\]

Furthermore, the number of green list tokens has variance at most
\[
\text{Var} |s|_G \leq T \frac{\gamma \alpha S^*}{1 + (\alpha - 1)\gamma} \left( 1 - \frac{\gamma \alpha S^*}{1 + (\alpha - 1)\gamma} \right).
\]
If we have chosen $\gamma \geq .5$, then we can use the strictly looser but simpler bound 
\[ \text{Var} |s|_G \leq T\gamma (1 - \gamma). \]

**Remark.** It may seem like there are a lot of messy constants floating around in this bound. However, when we choose $\gamma = \frac{1}{2}$ and $\delta = \ln(2) \approx 0.7$, this bound simplifies to 
\[ \mathbb{E} |s|_G \geq \frac{2}{3} TS^*, \quad \text{Var} |s|_G \leq \frac{2}{3} TS^* \left(1 - \frac{2}{3} S^* \right) \]
where $S^*$ is a bound on spike entropy with modulus 1/3. If we study the “hard” red list rules by choosing $\gamma = \frac{1}{2}$ and letting $\delta \to \infty$, we have 
\[ \mathbb{E} |s|_G \geq TS^*, \quad \text{Var} |s|_G \leq TS^* (1 - S^*) \]
where $S^*$ is a bound on spike entropy with modulus 1.

### 4.1. Sensitivity of the watermark test

The sensitivity of the soft watermark can be computed using standard type-II error analysis. For illustrative purposes, we estimate the type-II (false negative) error rate of a soft watermark with $\gamma = .5$ and $\delta = 2$. We assume 200 tokens are generated using OPT-1.3B (Zhang et al., 2022) using prompts from the C4 dataset’s RealNewsLike subset (Raffel et al., 2019). We also assume a detection threshold of $z = 4$ (which occurs at $\approx 128.2/100$ tokens) which gives us a type-I error (false positive) rate of $3 \times 10^{-5}$.

**Theoretical bound.** Our generations have an average spike entropy per sample of $S = 0.807$ over $\sim 500$ generations. Theorem 4.2 says that the expected number of green list tokens per generation is at least 142.2. Indeed, the empirical average is 159.5. For sequences with entropy equal to the mean ($S = 0.807$) we get $\sigma \leq 6.41$ tokens, and 98.6% sensitivity (1.4% type-II error rate), using a standard Gaussian approximation for the green list count. Note, this is a lower bound on the sensitivity for this particular entropy. If we use the true empirical mean of 159.5 rather than the theoretical bound, we get a $5.3 \times 10^{-7}$ type-II error rate, a realistic approximation but not a rigorous lower bound.

**Empirical sensitivity.** Empirically, 98.4% of generations are detected at the $z = 4$ (128 token) threshold when multinomial sampling is used. When 4-way beam search over a greedy decoding is used, we get 99.6% empirical sensitivity. Unlike the theoretical bounds, these are computed over all generations, which have the same length but vary in their individual entropies. Here, the primary source of type-II errors is low entropy sequences, as calculations above show that we expect a very low error rate when the entropy lies near the mean. To validate this, we examine the subset of 375/500 generations that have spike entropy above the 25th percentile, of which we detect 100% of generations at the $z = 4$ threshold.

**4.2. Impact on quality of generated text**

When the distribution produced by the language model is uniform (maximal entropy), the randomness of the green list results in tokens being uniformly sampled, and the perplexity remains untouched. Conversely, in the case of minimal entropy, where all probability mass is concentrated on a single token, the soft watermark rule has no effect and there is once again no impact on perplexity. However, the watermark rule does impact perplexity for tokens of moderate entropy. We provide a bound on this perplexity change in the supplementary material.

### 5. Private Watermarking

The watermark algorithms above are designed to be public. A watermark can also be operated in private mode, in which the algorithm uses a random key that is kept secret and hosted behind a secure API. If the attacker has no knowledge of the key used to produce the red list, it becomes more difficult for the attacker to remove the watermark as the attacker does not know which tokens are in the red list. However, testing for the presence of the watermark now requires using the same secure API and, if this API is public, access needs to be monitored to prevent an adversary from making too many queries using minor variants of the same sequence.

A range of more complex mechanisms are possible with different efficiency and security tradeoffs, and we discuss a few of these in the supplementary material.

### 6. Experiments

In this section we explore the behavior of the watermark using the OPT-1.3B model (Zhang et al., 2022). We measure watermark strength using the rate of type-I errors (human text falsely flagged as watermarked) and type-II errors (watermarked text not detected). We provide basic implementation details in Appendix A.1.

**Datasets and Prompts.** To simulate a variety of realistic language modeling scenarios we slice and dice a random selection of texts from the news-like subset of the C4 dataset (Raffel et al., 2019). For each random string, we trim a fixed
A Watermark for Large Language Models

Table 1. Selected outputs from non-watermarked (NW) and watermarked (W) multinomial sampling using $\gamma = 0.5$ and $\delta = 2.0$. The example in the first row has high entropy and correspondingly high z-scores, without any perceptible degradation in output quality. The lower row is a failure case where the watermark is too weak to be detected – it has low entropy and corresponding low z-scores.

<table>
<thead>
<tr>
<th>prompt</th>
<th>real completion</th>
<th>no watermark (NW)</th>
<th>watermarked (W)</th>
<th>$S$ (W)</th>
<th>$\alpha$ (NW)</th>
<th>$\alpha$ (W)</th>
<th>$\text{PPL}$ (W)</th>
<th>$\text{PPL}$ (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;...died out of court and publicly reconciled &quot;\n</td>
<td>returned with some moderate hits in the late 1990s and were inducted into the Rock and Roll Hall of Fame in 1997. With his brothers, Mr. Gibb won six Grammys. \n</td>
<td>continued to tour, and Barry became a television producer. \n</td>
<td>1990 album, “Spirits of the Century,” was a mixed critical and commercial success. \n</td>
<td>0.68</td>
<td>12.72</td>
<td>3.15</td>
<td>1.93</td>
<td></td>
</tr>
<tr>
<td>...cond season at Hall Bros Oval. \n</td>
<td>we coped a few floggings with all those junior blokes coming in; Galvin said. \n</td>
<td>we coped a few floggings with all those junior blokes coming in; Galvin said. \n</td>
<td>we coped a few floggings with all those junior blokes coming in; Galvin said. \n</td>
<td>0.58</td>
<td>3.11</td>
<td>1.05</td>
<td>1.04</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2. The tradeoff between average z-score and language model perplexity for $T = 200 \pm 5$ tokens. (left) Multinomial sampling. (right) Greedy and beam search with 4 and 8 beams for $\gamma = 0.5$. Beam search promotes higher green list usage and thus larger z-scores with smaller impact to model quality (perplexity, PPL).

length of tokens from the end and treat them as a “baseline” completion. The remaining tokens are a prompt. A larger oracle language model (OPT-2.7B) is used to compute perplexity (PPL) for the generated completions and for the human baseline.

**Watermark Strength vs Text Quality.** One can achieve a very strong watermark for short sequences by choosing a small green list size $\gamma$ and a large green list bias $\delta$. However, creating a stronger watermark may distort generated text. Figure 2 (left) shows the tradeoff between watermark strength (z-score) and text quality (perplexity) for various combinations of watermarking parameters. We compute results using $500 \pm 10$ sequences of length $T = 200 \pm 5$ tokens for each parameter choice. Interestingly, we see that a small green list, $\gamma = 0.1$ is pareto-optimal.

In addition to these quantitative results, we show examples of real prompts and watermarked outputs in Table 1 to provide a qualitative sense for the behavior of the test statistic and quality measurement on different kinds of prompts. Additional examples are compiled in Appendix A.3.

**Ironing in the Watermark with Beam Search.** Figure 2 (right) shows the tradeoff between watermark strength and accuracy when beam search is used. Beam search has a synergistic interaction with the soft watermarking rule. Particularly when 8 beams are used, the points in Figure 2 form an almost vertical line, showing very little perplexity cost to achieve strong watermarking.

**Watermark Strength vs Number of Tokens.** Theory predicts that the type I and type II error rates of the watermark should decay to zero as the sequence length $T$ increases. Figure 3 shows the strength of the watermark, measured using the average z-score over samples, as $T$ sweeps from 2 to 200. Curves are shown for various values of $\delta$ and $\gamma$. The left two charts use multinomial sampling, while the right chart uses 8-way beam search and $\gamma = 0.25$. Once again, we see the power of the beam search in achieving high green list ratios; even for the moderate bias of $\delta = 3$, an average z-score greater than 5 is achieved for as few as 35 tokens.

**Performance and Sensitivity for Multinomial Sampling.** To show the sensitivity of the resulting hypothesis test based...
on the observed z-scores, we provide a table of error rate for various watermarking parameters in Table 2. We also sweep a range of thresholds in ROC charts in Figure 4. We further report detection performance and error rates for various cutoffs in Appendix C, and provide a comparison between empirical z-scores and theoretical predictions. Note that no type-I (false positive) errors were observed for any run shown in the error tables (see the columns of 0.0’s).

7. Attacking the watermark

Like any software tool, care must be taken when implementing a watermark and watermark detector so that security is maintained. Otherwise, an adversarial user may modify text to add red list tokens, and thus avoid detection. In many cases, simple attacks can be avoided by properly normalizing text before hashes are computed. In Appendix B.1, we discuss a range of attacks that we are currently aware of, and methods to mitigate them. In the next section, we implement and evaluate an example of a representative attack using a second, smaller language model.


We study a realistic black-box attack by attempting to remove the presence of the watermark by replacing spans in the original output text using another language model. We treat the watermark algorithm as if it is private, mocking seclusion behind an API. The attacker does not have access to the locations of green list tokens and instead tries to modify the text through token replacement at random indices until a certain word replacement budget, $\varepsilon$, is reached. The budget constraint maintains a level semantic similarity between the original watermarked text and the attacked text, otherwise the “utility” of the original text for its intended task may be lost. Also, each span replacement in the attack is performed via inference using a multi-million parameter language model. While this is roughly a third the size of the target model, it means that the attack incurs an associated cost per step implying that a base level of efficiency with respect to model calls would be desired in practice.

In our experiment, we adopt T5-Large (Raffel et al., 2020) as the replacement model and iteratively select and replace tokens until the attacker either reaches the budget, or no
Table 2. Empirical error rates for watermark detection using multinomial sampling and beam search. Each row is averaged over \( \sim 500(−4/+)7 \) generated sequences of length \( T = 200\pm5 \). A maximum of one type-I (false positive) error was observed for any given run. All soft watermarks at \( \delta = 2.0 \) incur at most 1.6% (8/500) type-II error at \( z = 4 \). No type-II errors occurred for the hardest watermarks with \( \delta = 5.0 \) and \( \gamma = 0.25 \). A more extensive set of error rates is provided in Table 8.

<table>
<thead>
<tr>
<th>( z=4.0 )</th>
<th>sampling</th>
<th>( \delta )</th>
<th>( \gamma )</th>
<th>FPR</th>
<th>TNR</th>
<th>TPR</th>
<th>FNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>m-nom.</td>
<td>1.0</td>
<td>0.50</td>
<td>0.0</td>
<td>1.0</td>
<td>0.767</td>
<td>0.233</td>
<td></td>
</tr>
<tr>
<td>m-nom.</td>
<td>1.0</td>
<td>0.25</td>
<td>0.0</td>
<td>1.0</td>
<td>0.729</td>
<td>0.271</td>
<td></td>
</tr>
<tr>
<td>m-nom.</td>
<td>2.0</td>
<td>0.50</td>
<td>0.0</td>
<td>1.0</td>
<td>0.984</td>
<td>0.016</td>
<td></td>
</tr>
<tr>
<td>m-nom.</td>
<td>2.0</td>
<td>0.25</td>
<td>0.0</td>
<td>1.0</td>
<td>0.994</td>
<td>0.006</td>
<td></td>
</tr>
<tr>
<td>m-nom.</td>
<td>5.0</td>
<td>0.50</td>
<td>0.0</td>
<td>1.0</td>
<td>0.996</td>
<td>0.004</td>
<td></td>
</tr>
<tr>
<td>m-nom.</td>
<td>5.0</td>
<td>0.25</td>
<td>0.0</td>
<td>1.0</td>
<td>1.000</td>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>8-beams</td>
<td>1.0</td>
<td>0.25</td>
<td>0.0</td>
<td>1.0</td>
<td>0.819</td>
<td>0.181</td>
<td></td>
</tr>
<tr>
<td>8-beams</td>
<td>2.0</td>
<td>0.25</td>
<td>0.0</td>
<td>1.0</td>
<td>0.994</td>
<td>0.006</td>
<td></td>
</tr>
<tr>
<td>8-beams</td>
<td>5.0</td>
<td>0.25</td>
<td>0.0</td>
<td>1.0</td>
<td>1.000</td>
<td>0.000</td>
<td></td>
</tr>
</tbody>
</table>

more suitable replacement candidates are returned.

**Details of the T5 Span Attack.** We tokenize the watermarked text using the T5 tokenizer. Then, while fewer than \( \varepsilon T \) successful replacements have been performed or a maximal iteration count is reached:

1. Randomly replace one word from the tokenization with a `<mask>`.
2. Pass the region of text surrounding the `mask` token to T5 to obtain a list of \( k = 20 \) candidate replacement token sequences via a 50-way beam search, with associated scores corresponding to their likelihood.
3. Each candidate is decoded into a string. If one of the \( k \) candidates returned by the model is not equal to the original string corresponding to the masked span, then the attack succeeds, and the span is replaced with the new text.

After attacking a set of 500 sequences of length \( T = 200\pm5 \) token sequences this way, we compute updated \( z \)-scores and tabulate error rates (Table 9 in the Appendix). We also generate ROC plots for a range of \( \varepsilon \) budgets. While this attack is effective at increasing the number of red list tokens in the text, as shown in Figure 5, we only measure a decrease in watermark strength of 0.01 AUC when \( \varepsilon = 0.1 \). While the watermark removal is more successful at a larger budget of 0.3, the average PPL of attacked sequences increases by \( 3 \times \), in addition to requiring more model calls.

**8. Related Work**

The idea of watermarking has a long history. Watermarking of digital text has been considered challenging, due to its discrete nature (Katzenbeisser & Petitcolas, 2000). Watermarking is considered easier for continuous data, where a variety of effective strategies are known (Petitcolas et al., 1999; Zhu et al., 2018; Lu et al., 2021; Boenisch, 2021). Note that watermarking is a special case of steganography, the task of embedding arbitrary hidden information into data.

**Watermarking Natural Language.** Early approaches to watermarking natural text in digital form (Atallah et al., 2001; 2003; Venugopal et al., 2011) pose similar desiderata as in Section 1, except for classical models.

Rule-based watermarks were further developed in a series of works (Chiang et al., 2004; Topkara et al., 2006a;b; Meral et al., 2009) with variants also embedding watermarks based on synonym tables instead of only parse trees. Early developments were summarized in Jalil & Mirza (2009), but strong watermarks significantly degraded the text quality due to the limited flexibility of language models of the time.

While approaches via hierarchical language models in Wilson et al. (2014) still required human interactions, the emergence of modern neural language models (Vaswani et al., 2017; Devlin et al., 2019) also allowed for improved watermarking/steganography (Fang et al., 2017; Ziegler et al., 2019; Dai & Cai, 2019; He et al., 2022). Fang et al. (2017) propose such a neural steganography approach where, to encode a message of \( w \) bits, the message is first separated into blocks of length \( b \). Then, the vocabulary \( V \) of a language model is partitioned at random into disjoint sets of size \( |V|/2^p \). A generative LM can then encode the message...
by generating only a token from the “allowed” set at each position. However, this hard rule reduces the quality of generated text, boxing the LM into a small selection of tokens for each step.

In a somewhat similar approach to our proposal, Kaptchuk et al. (2021) constructs a framework that adapts traditional public-key cryptographic steganography specifically for “natural” communication channels like text using generative models. However, their method, Meteor, relies on a synchronized model framework where the sender and receiver agree on a shared generative model used to embed and decode the hidden bits of information being sent.

Another approach (Ueoka et al., 2021) uses mask-infilling via BERT to edit already-generated text for steganography. Finally, Abdelnabi & Fritz (2021) design an end-to-end system where both encoding and decoding are handled by text-to-text language models that are trained adversarially for this purpose, and Gu et al. (2022) implants a series of backdoor triggers through a finetuning process to cause biased responses to specific inputs, a behavior detectable at verification time.

Recently, Aaronson (2022) announced that he is also studying cryptographic approaches to watermarking in collaboration with OpenAI. Their preliminary method is based on biasing of the LM output, as opposed to complete determination as in Fang et al. (2017). While details are not currently available, the description suggests that hashing of n-gram sequences is involved. We hope to extend our comparison to this work when more information becomes available.

In contrast to other currently published works, we want to focus on strategies that are simultaneously minimally restrictive to a language model, leverage the LMs own understanding of natural text, require no usage of the LM to decode the watermark, and can be theoretically analyzed and validated.

Post-hoc Detection. An alternative to watermarking is post-hoc analysis of machine-generated text, for example using language model features or finetuning existing large language models to behave as detectors (Zellers et al., 2019; Tan et al., 2020), see an overview in Jawahar et al. (2020). Implementation details, such as sampling strategies, can be reverse-engineered from text (Tay et al., 2020). However, detection approaches are slowly losing ground for this purpose, and Gu et al. (2022) note that detection strategies for GPT-2 struggle with GPT-3. Further, known detectors are vulnerable to adversarial attacks that degrade their functionality (Wolff & Wolff, 2022).

While efforts to provide strong detectors continue, as in Tian (2023a), ultimately language model progress may make detection infeasible. All post-hoc detection methods require the LM to be significantly biased away from human text in some measurable way, such as low variation in perplexity across sentences (Tian, 2023b). Even for current LLMs, this margin might be small. Detection schemes that operate within this small margin are susceptible to labeling human text as false-positive, a concern that is especially pressing for non-native speakers and people who use computer writing assistance tools. Such populations might be especially at risk for false-positives, which could lead to academic problems if these detectors are used in schools (Butoi, 2023).

9. Conclusion

There is one more important property of the proposed method: The z-statistic used for detection depends only on the green list size parameter γ and the hash function. There is no dependence on δ or any other factor related to how the green list is enforced. For this reason, one can deploy the watermark using context-specific δ or green list enforcement rules for different kinds of text (e.g., prose vs code) while using the same downstream detector. One can also change a proprietary implementation of the watermarked sampling algorithm without any need to change the detector. Finally, the watermarking method could be turned on only in certain contexts, for example when a specific user seems to exhibit suspicious behavior.

There are still a number of open questions regarding watermarking. For example, what is the best way to test for the watermark in a streaming context, or in a context where a short span of watermarked text lives inside a longer non-watermarked span? We hope our present results are enough to convince readers that watermarks could be a practical tool for combating malicious uses of generative models, and we leave these additional questions for future research.

10. Acknowledgements

This work was made possible by the ONR MURI program, DARPA GARD (HR00112020007), the Office of Naval Research (N000142112557), and the AFOSR MURI program. Commercial support was provided by Capital One Bank, the Amazon Research Award program, and Open Philanthropy. Further support was provided by the National Science Foundation (IIS-2212182), and by the NSF TRAILS Institute (2229885).

References


Abdelnabi, S. and Fritz, M. Adversarial Watermarking Transformer: Towards Tracing Text Provenance
A Watermark for Large Language Models


Butoi, V. Things like GPTZero are scary. I’m sure the creator didn’t have the intention, but the fact that it’s marketed as “a solution to detecting AI written responses” even though there’s no evidence to show it works consistently and is nevertheless being EMPLOYED by schools, is crazy., January 2023. URL https://twitter.com/VictorButoi/status/1614779846210752512.


Goodside, R. There are adversarial attacks for that proposal as well — in particular, generating with emojis after words and then removing them before submitting defeats it., January 2023. URL https://twitter.com/goodside/status/1610682909647671306.


We provide series of representative outputs from different ranges in the sample space for model generations under a soft watermark with parameters $\delta = 2.0$, $\gamma = 0.5$ under the multinomial sampling scheme. To tabulate these outputs, the ~500 generations collected at this setting are either sorted by the average spike entropy of the watermarked model’s output distribution at generation time, or the measured test statistic, the $z$-score for that sequence. The top and bottom 5 samples according to these orderings are shown for both entropy (Table 4 and Table 3) and $z$-score (Table 6 and Table 5).
A Watermark for Large Language Models

Table 4. Low spike entropy examples under a $\delta = 2.0$, $\gamma = 0.5$ watermark with multinomial sampling.

<table>
<thead>
<tr>
<th>idx</th>
<th>prompt</th>
<th>real completion</th>
<th>no watermark (NW)</th>
<th>watermarked (W)</th>
<th>$S$ (NW)</th>
<th>$W$</th>
<th>PPL (NW)</th>
<th>PPL (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>132</td>
<td>[...second season at Hall Bros Oval.](\text{Galvin said,})</td>
<td>we copped a few floggings with all those junior blacks coming in.</td>
<td>we copped a few floggings with all those junior blacks coming in.</td>
<td>we copped a few floggings with all those junior blacks coming in.</td>
<td>0.59</td>
<td>-1.27</td>
<td>-1.13</td>
<td>1.05</td>
</tr>
<tr>
<td>850</td>
<td>[...][...]</td>
<td>Village: (\text{Copper Ski Resort})</td>
<td>Village: (\text{Copper Mountain Resort})</td>
<td>Village: (\text{Copper Mountain Resort})</td>
<td>0.61</td>
<td>0.00</td>
<td>5.37</td>
<td>4.58</td>
</tr>
<tr>
<td>939</td>
<td>[...jugs.(\text{Trump said}), now we have</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>444</td>
<td>[...31 on the season with a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1177</td>
<td>[...South Elliott Place, near</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


To compute perplexity, the larger, Oracle Language Model is fed the original prompt as input, as described in the main body, and perplexity is computed via taking the exponential of the average token-wise loss according to the oracle's next token distribution at every output index. Note that loss is computed for only the generated tokens produced by either a watermarked or non-watermarked model.

B. Detailed Threat Model

For completeness, we formally define the threat model and then enumerate a selection of possible attacks. As described, attacks may occur when malicious users operate bots/sock-puppets on social media, try to fool a CAPTCHA, or complete an academic assignment (Foltýnek et al., 2019). In this work we formally define adversarial behavior as all efforts by a party the using machine-generated text to remove the watermark. It is ultimately important to remember that we describe a watermark on the tokens of the generated text, i.e. on its form and style, and not on its semantic content. For example, a completely new essay written based on an outline or initial draft provided by a LM could not be detected. Such semantic watermarks may be possible, but we do not study this setting here.

Threat Model. We assume two parties, a model owner providing a text generation API, and an attacker attempting to remove the watermark from the API output. The attacker moves second, and is aware that the API contains a watermark. In public mode, the attacker is aware of all details of the hashing scheme and initial seed. In private mode, the attacker is aware...
of the watermark implementation, but has no knowledge of the key of the pseudo-random function $F$. The attacker attempts to reduce the number of green-listed occurrences in the text, reducing the $z$-score computed by a defender. In public mode, any party can evaluate the watermark. In private mode, only the model owner can evaluate the watermark and provides a text detection API. We assume that this API is rate-limited. The attacker assumes this access to other non-watermarked language models, but these models are weaker than the API under attack. The attacker is allowed to modify the generated text in any way.

Note that removing the watermark is always a trivial task if language model quality is disregarded – one can simply replace the entire text with random characters. For this reason only attacks that result in a reasonable language quality trade-off for the attacker are relevant. A defense is hence also successful if any watermark removal by the attacker reduces the quality of generated text to that of generated text achievable using a public model.

### B.1. Types of Attacks

Three types of attacks are possible. **Text insertion** attacks add additional tokens after generation that may be in the red list and may alter the red list composition of downstream tokens. **Text deletion** removes tokens from the generated text, potentially removing green list tokens and modifying downstream red lists. This attack increases the monetary costs of generation, as the attacker is “wasting” tokens, and may reduce text quality due to effectively decreased LM context width. **Text substitution** swaps one token with another, potentially introducing one red list token, and possibly causing downstream red listing. This attack can be automated through dictionary or LM substitution, but may reduce the quality of the generated text.

Below we catalog a range of attacks that fall into these categories.

**Paraphrasing Attacks.** A baseline substitution attack is manual paraphrasing by the human attacker. This attack is
Table 6. Low z-score examples under a $\delta = 2.0, \gamma = 0.5$ watermark with multinomial sampling.

<table>
<thead>
<tr>
<th>idx</th>
<th>prompt</th>
<th>real completion</th>
<th>no watermark (NW)</th>
<th>watermarked (W)</th>
<th>$S$ (NW)</th>
<th>$z$ (NW)</th>
<th>$S$ (W)</th>
<th>$z$ (W)</th>
<th>PPL (NW)</th>
<th>PPL (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>132</td>
<td>[...] second at Hall Bros Oval. (FP) The defender also admitted his surprise at Young's run to the finals but credited the injection of youth into the side. (TN) We were really in a building phase last year and we copped a few foggings with all these junior blokes coming in. (FP) Galvin said. &quot;We don't keep that core group together for two years and I think we've come along quicker than we anticipated.{truncated}</td>
<td>we copped a few foggings with all these junior blokes coming in. (FP) Galvin said. &quot;We don't keep that core group together for two years and I think we've come along quicker than we anticipated.{truncated}</td>
<td>we copped a few foggings with all these junior blokes coming in. (FP) Galvin said. &quot;We don't keep that core group together for two years and I think we've come along quicker than we anticipated.{truncated}</td>
<td>0.59</td>
<td>-1.27</td>
<td>-1.13</td>
<td>1.05</td>
<td>1.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>444</td>
<td>[...] on the season with a 2.76 GAA and 906 save percentage. (TN) Now, we've kept that core group together for two years and I think we've come along quicker than we anticipated. (FP) ...</td>
<td>stopped 18 of 19 shots in the Blue Jackets' 2-0 loss to the Islanders on Monday. The defeat dropped him to 29-22-8 on the season with a 2.74 GAA and 907 save percentage. (TN) Now, we've kept that core group together for two years and I think we've come along quicker than we anticipated. (FP) ...</td>
<td>stopped 28 of 30 shots in the Blue Jackets' 3-2 shutout loss to the Islanders on Sunday. The loss dropped him to 30-22-1 on the season with a 2.76 GAA and 906 save percentage. (TN) Now, we've kept that core group together for two years and I think we've come along quicker than we anticipated. (FP) ...</td>
<td>0.62</td>
<td>-0.99</td>
<td>2.40</td>
<td>1.33</td>
<td>1.45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>302</td>
<td>[...] rating. The price target for Sol Gel Technologies is set to $20. Sol Gel Technologies shares closed at $11.13 on Friday. (FP) Analysts at Wells Fargo initiated coverage on Landmark Infrastructure Partners. L.P. (NASDAQ: LMRK) with a Market Perform rating. The price target for Landmark Infrastructure Partners is set to $18.50. Landmark Infrastructure Partners closed at $17.75 on Friday. (TN) ...</td>
<td>L.P. (NASDAQ: LMRK) with an overweight rating. The price target for Landmark Infrastructure Partners is set to $12. Landmark Infrastructure Partners shares closed at $16.02 on Friday. (FP) Analysts at Jef...</td>
<td>L.P. (NASDAQ: LMRK) with an overweight rating. The price target for Landmark Infrastructure Partners is set to $12. Landmark Infrastructure Partners shares closed at $16.02 on Friday. (FP) Analysts at Jef...</td>
<td>0.66</td>
<td>-2.55</td>
<td>2.83</td>
<td>1.75</td>
<td>2.08</td>
<td></td>
<td></td>
</tr>
<tr>
<td>482</td>
<td>[...] jinika Aigner, with sister Elisabeth as her guide, sprang a surprise in women's visually impaired event on the fourth day of the World Para Alpine Skiing World Cup at the Spanish resort of La Molina. (FP) ...</td>
<td>itzlerter's Theo Gimur, Paralympic champion in the men's standing giant slalom, succeeded at the third attempt in beating France's 18-year-old world champion Arthur Bunchet at the World Para Alpine Ski...</td>
<td>eden's Chris Vos won gold in the men's super-G at the World Para Snowboard World Cup Finals in Klovjio in Sweden. (FP) The final day of action in Klovjio concludes today with the men's super-G and women's...</td>
<td>0.70</td>
<td>-0.71</td>
<td>2.83</td>
<td>2.86</td>
<td>3.34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>939</td>
<td>[...] jins. (FP) And now we have Trump calling on Bee to be fired. You know who else hates comedy at their expense? Authoritarian leader's President Abdel Fattah el-Sisi, a man Trump loves, banned Egypt's</td>
<td>version of The Daily Show because of the way its comedy mocked the government. The show's star, Bassem Yousef, had to flee the country and is now living in exile with his family. (FP) Turkey's strongman...</td>
<td>top comedy show after the host criticized the military's coup over a year ago. and then President Vladimir Putin banned comedy on Russian television over the same topic. (FP) It's not hard to see why Trump...</td>
<td>0.61</td>
<td>0.99</td>
<td>2.83</td>
<td>4.86</td>
<td>1.27</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7. Reference table for possible outcomes of the hypothesis test. Type-I errors, false positives, are improbable by construction of the watermarking approach, but type-II errors, false negatives, appear naturally for low-entropy sequences that cannot be watermarked.

Text is watermarked, i.e. machine-generated

<table>
<thead>
<tr>
<th>Hypothesis Test is Rejected</th>
<th>TP - Watermarked text is correctly flagged.</th>
<th>FN - Text is watermarked, but cannot be detected.</th>
<th>FP - Text that is not watermarked is flagged.</th>
<th>TN - Text is not watermarked and is not flagged.</th>
</tr>
</thead>
</table>

A more scalable version of this attack is to use automated paraphrasing. An attacker that has access to a public language model can use this model to rephrase the output of the generated model. We provide an experimental evaluation of this attack in Section 7.1 in the main body of the work.

Discreet Alterations. An attacker could make small alterations, adding additional whitespaces, or misspelling a few words to impact the computation of the hash. A well-constructed watermark should normalize text to ignore explicit whitespaces when computing the hash. Changing the spelling of many words is likely to severely degrade the quality of text. When implemented carefully, surface level alterations should not pose a serious threat to a watermark.

Tokenization Attacks. An attacker can modify text so that sub-word tokenization of a subsequent word changes. For example (again with BPE), if the text fragment `life, \text{\textbackslash Verri\textbackslash l\textbackslash i\textbackslash i\textbackslash u\textbackslash s}` is modified to `life, \text{\textbackslash Verr\textbackslash i\textbackslash l\textbackslash i\textbackslash i\textbackslash u\textbackslash s}` (i.e. `\textbackslash n` is replaced), then the tokenization of the succeeding word also switches from \texttt{Verr\_ll\_ll\_us} to \texttt{Ver\_ll\_ll\_us}. This results in more red list tokens than one would expect from a single insertion. The attack can contribute to the effectiveness of a more powerful attack, but most tokens in a default sentence will not be vulnerable.
Homoglyph and Zero-Width Attacks. This is a special case of the discreet alteration attack. The effect of tokenization attacks can be multiplied through homoglyph attacks (Gabrilovich & Gontmakher, 2002). Homoglyphs attacks are based on the fact that unicode characters are not unique, with multiple unicode IDs resolving to the same (or a very similar-looking) letter. This breaks tokenization, for example the word Lighthouse (two token) expands to 9 different tokens if i and s are replaced with their equivalent Cyrillic unicode characters. Security against Homoglyph and tokenization attacks can be maintained using input normalization before the text is tested for watermarks, for example via canonicalization as in Helfrich & Neff (2012). Otherwise, simple replacements of characters with their homoglyphs could break enough tokens to remove the watermark.

Likewise, there are Zero-width joiner/non-joiner unicode characters that encode zero-width whitespace and hence are effectively invisible in most languages. Like homoglyphs, these characters must be removed through canonicalization (Pajola & Conti, 2021; Boucher et al., 2022).

Generative Attacks.

Figure 6. Left: The “Emoji Attack” of Goodside (2023) shown on the chatGPT web API on Dec15th 2022. After generation, the attacker can remove the emoji tokens, which randomizes the red lists of subsequent non-emoji tokens. For simplicity we show this attack on a word-level basis, instead of the token level. Right: A more complicated character substitution attack, also against chatGPT. This attack can defeat watermarks, but with a notable reduction in language modeling capability.

Generative attacks abuse the capability of large language models for in-context learning, and prompt the model to change its output in a predictable and easily reversible way. For example, the Emoji attack of Goodside (2023) proceeds by prompting the model to generate an emoji after every token, see Figure 6, left. These emojis can be removed, randomizing the red list for subsequent tokens. More broadly, all attacks that prompt the model to change its output “language” in a predictable way can potentially cause this, for example prompting the model to replace all letters a with e, see Figure 6, right. Or, as a reverse homoglyph attack, prompting the model to “switch the letter i with i”, where the second i is a Cyrillic letter.

These attacks are the strongest tools against watermarking to our knowledge, but also require a strong LM with the capacity to follow the prompted rule without a loss in output quality. Additionally, this increases the cost of text generation by requiring more tokens than usual to be generated and reducing effective context width.

A defense against these attacks is to include negative examples of such prompts during finetuning, training the model to reject these requests. Note that instruction finetuning is already common (for example in ChatGPT) for other categories of malicious prompts, using reinforcement learning protocols (RLHF) (Christiano et al., 2017; Ouyang et al., 2022; Bai et al., 2022).

Evaluating Repetitive Text. A subtlety of the proposed approach is that tokens in the green list are only pseudo-random, and n-grams of text that are repeated will always be scored in the same manner. Assume a 2-gram, such as “Barack Obama” happens to green-list “Obama”. Repetitive usage of this 2-gram would result in a higher than expected number of green tokens. In a worst-case scenario, human-generated text with a high number of repetitions of this 2-gram may be erroneously flagged as machine-generated.
Two remedies are possible: The first is to simply increase the length $h$ of the PRNG function, thereby increasing the variability of the green-listed words, as larger $(h + 1)$-grams are much less likely to be repeated. A better remedy (possibly used in conjunction with the first) is not to count repeated $n$-grams when checking for the watermark. In the example above, the 2-gram “Barack Obama” would be counted on its first occurrence, and then subsequently ignored when it appears again; it is counted as neither green nor red, and the token counter $T$ is not incremented.

In addition to preventing false positives, skipping repeated $n$-grams can also make the detector more sensitive. A repeated $n$-gram is likely to be low-entropy, and so we cannot avoid its use when it contains red list words. By excluding these from the count, we can keep the green list fraction high and maintain high sensitivity.

C. Detection Accuracy of Multinomial Sampling

When a multinomial sampler is used (which is assumed by Theorem 4.2), we use the softmax output with standard temperature hyperparameter $\text{temp}=0.7$. We analyze the alignment between the empirical strength of the watermark and the theoretical lower bound for $\gamma = .5$ in Figure 7. We find that the theoretical bound is quite tight for smaller values of $\delta$, but the theorem under-estimates watermark sensitivity for larger $\delta$.

ROC curves for multinomial sampling, and greedy decoding with 8-way beam search in the 200 token case are depicted in Figure 8 and Figure 9 (Subsets of Figure 4 from the main work.). Tables with error rates and accuracy numbers at selected $z$ values are provided in Table 8, a subset of which was shown in Table 2.

![Figure 7. Empirical green list fraction vs bias parameter $\delta$. We compare to the theoretical bound predicted by Theorem 4.2.](image)

D. Impact of Watermarking on Model Factuality

A key benefit of the soft watermarking scheme is that it (passively) adapts to the current entropy in the model’s output distribution. If the model is highly confident on its next few token predictions, say those representing a specific named entity, then a soft watermark will not affect those predictions regardless of their factuality or groundedness. On the other hand, if the model is not confident on any particular tokens, then under standard decoding schemes, whether or not the final decoded output is hallucinatory will be a random event, and the watermark has an equal chance of upweighting tokens that result in more factual or more hallucinatory utterances.

To illustrate this, we present a small experiment that isolates this behavior. We take a model with reasonable competency in knowledge-intensive, closed-book question answering and evaluate its performance on the validation set of the TriviaQA question answering dataset (Joshi et al., 2017). Hypothesis: since answers to factoid questions should be short, low entropy sequences, a soft watermark will yield low detection statistics, however, task performance will not degrade much under application of the watermark. The results for this experiment are shown in Table 10 and provide some evidence that in factuality critical generation scenarios, a softly watermarked model is unlikely to deviate that much from its unwatermarked behavior (for better or worse). We observe less than a 4 point drop in Exact Match performance under the application.
of a standard soft watermark ($\gamma, \delta = 0.5, 2.0$) for both Google’s FLAN-UL2 model (Tay et al., 2022) and Huggingface BigScience’s BLOOMZ model (Muennighoff et al., 2022).

However, we note that this particular experimental setup is not a situation where we would actually deploy the watermark or expect it to work very well. Generating 5 to 10 tokens per question under greedy decoding and then testing those tokens for exact correctness, is something of a worst case estimate on the cost of watermarking. In this scenario the prompt is highly constraining and the only things the watermark can do are either nothing, or directly cause the model to deviate from the argmax. Such deviations would be detrimental on any question where the model “knows” the correct answer but isn’t overwhelmingly confident in the token sequence required to represent it (especially the surface form). We leave a more comprehensive study of the impacts of watermarking strategies on the factuality of LLMs in question answering and other knowledge intensive settings to future research.

E. Minor Variations

Multiple Watermarks. A company might also apply multiple watermarks to generated text, taking the union of all red lists at each token. This is a compromise in terms of watermark effectiveness, compared to a single watermark, however it allows additional flexibility. A company could run a public/private watermarking scheme, giving the public access to one of the watermarks to provide transparency and independent verification that text was machine-generated. At the same time, the company can keep the second watermark private and test text against both watermarks, to verify cases reported by the public watermark, or again to provide a stronger detection API. Such a setup would be especially effective in detecting whether an attack took place that attempted to remove the public watermark.

Selective Watermarks in response to malicious activity Watermarks could also be used selectively. An API owner could turn on watermarking (or dial up its strength considerably via increased $\delta$) only when faced with suspicious API usage by some accounts, for example if a request appears to be part of malicious activity like creating synthetic tweets. This would give more leeway to benign API usages, but allow for improved tracing of malicious API utilization.

Discovering A Watermarking Scheme So far we assumed that an attacker is aware that a watermark is present. Could the attack discover this fact only by analyzing generated text? For a hard watermark, this would be easy: Some combinations of tokens will never be generated by the model, no matter how strongly they are prompted. Yet, for a soft watermark (especially with small $\delta$), that depends on, e.g. $h = 10$ tokens via a more robust watermarking scheme, this becomes harder. The
A Watermark for Large Language Models

Figure 9. (a) ROC curve with AUC values for watermark detection. Curves for several choices of watermark parameter $\delta$ are shown - greedy decoding and 8-way beam search is used to generate tokens in all settings. (b) The same chart, but with different axes to make detail visible. Similarly to Figure 8, higher $\delta$ values achieve stronger performance, but additionally we see that for a given $\delta$, the beam search allows the watermark to capture slightly more AUC than the corresponding parameters under the multinomial sampling scheme.

attacker would need to distinguish the modification of green list logits via $\delta$ from naturally occurring biases of the LM.

F. Proof of Theorem 4.2

We begin our proof with a useful lemma. Using the spike entropy, we can predict how often a watermarked language model will spit out a green list token. When the entropy is high, the language model has a lot of freedom and we expect the model to use green list tokens aggressively. When the entropy is low, the model is more constrained and it is more likely to use a red list token.

**Lemma F.1.** Suppose a language model produces a raw (pre-watermark) probability vector $p \in (0, 1)^N$. Randomly partition $p$ into a green list of size $\gamma N$ and a red list of size $(1 - \gamma)N$ for some $\gamma \in (0, 1)$. Form the corresponding watermarked distribution by boosting the green list logits by $\delta$, as in Equation (4). Define $\alpha = \exp(\delta)$.

Sample a token index $k$ from the watermarked distribution. The probability that the token is sampled from the green list is at least

$$\mathbb{P}[k \in G] \geq \frac{\gamma \alpha}{1 + (\alpha - 1)\gamma} S \left( p, \frac{(1 - \gamma)(\alpha - 1)}{1 + (\alpha - 1)\gamma} \right).$$

**Proof.** When we add $\delta$ to the logits corresponding to the green list words, we increase their probabilities of being sampled. We replace the raw probability $p_k$ for each green list word with the enlarged probability

$$p_k^w \triangleq \frac{\alpha p_k}{\sum_{i \in R} p_i + \alpha \sum_{i \in G} p_i},$$

where $G$ is the set of green list indices and $R$ is the complementary set of red list indices. We denote the sizes of these sets as $N_G$ and $N_R$, respectively.

We begin our proof by bounding the size of a randomly chosen green list probability after it has been enlarged. Consider the following process for creating the lists. First, choose a random entry $p_k$ and place it in the green list. Then, randomly sample the remaining entries in the green list. The expected value of a randomly chosen probability from the green list can be written

$$\mathbb{E}_{k \in N_G} \mathbb{E}_{\sum_{i \in R} p_i + \alpha \sum_{i \in G} p_i} \frac{\alpha p_k}{\alpha \sum_{i \in G} p_i} \frac{p_k}{\sum_{i \in G} p_i + \alpha \sum_{i \in R} p_i},$$

$$\mathbb{E}_{k \in N_G, R} \sum_{i \in R} p_i + \alpha \sum_{i \in G} p_i,$$
Table 8. Empirical error rates for watermark detection using multinomial sampling and beam search. Each row is averaged over \( \sim 500 \) generated sequences of length \( T = 200 \pm 5 \). A maximum of one type-I (false positive) error was observed for any given run. All soft watermarks at \( \delta = 2.0 \) incur at most 1.6% (8/500) type-II error at \( z = 4 \). No type-II errors occurred for the hardest watermarks with \( \delta = 5.0 \) and \( \gamma = 0.25 \).

<table>
<thead>
<tr>
<th>( z = 4.0 )</th>
<th>( z = 5.0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \delta )</td>
<td>( \gamma )</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>m-nom.</td>
<td>1.0</td>
</tr>
<tr>
<td>m-nom.</td>
<td>1.0</td>
</tr>
<tr>
<td>m-nom.</td>
<td>2.0</td>
</tr>
<tr>
<td>m-nom.</td>
<td>2.0</td>
</tr>
<tr>
<td>m-nom.</td>
<td>5.0</td>
</tr>
<tr>
<td>m-nom.</td>
<td>5.0</td>
</tr>
<tr>
<td>8-beams</td>
<td>1.0</td>
</tr>
<tr>
<td>8-beams</td>
<td>1.0</td>
</tr>
<tr>
<td>8-beams</td>
<td>2.0</td>
</tr>
<tr>
<td>8-beams</td>
<td>2.0</td>
</tr>
<tr>
<td>8-beams</td>
<td>5.0</td>
</tr>
<tr>
<td>8-beams</td>
<td>5.0</td>
</tr>
</tbody>
</table>

Table 9. Error rates for watermarked text before and after attack (w/attck) for generations of length \( T = 200 \pm 5 \). For all settings we use \( (\delta, \gamma) = (2.0, 0.5) \). Results are shown for both multinomial sampling and greedy 8-way beam search. The TPR and FNR rates without the attack are shown for reference, but they have no dependence on the attack budget \( \varepsilon \). For all experiments, no false positives were observed and so FPR = 0 and TPR = 1.

<table>
<thead>
<tr>
<th>( \varepsilon )</th>
<th>count</th>
<th>TPR@4.0 FNR@4.0</th>
<th>w/attck TPR@4.0 FNR@4.0</th>
<th>w/attck TPR@5.0 FNR@5.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>m-nom.</td>
<td>0.1</td>
<td>487</td>
<td>0.984 0.016</td>
<td>0.819 0.181</td>
</tr>
<tr>
<td>m-nom.</td>
<td>0.3</td>
<td>487</td>
<td>0.984 0.016</td>
<td>0.353 0.647</td>
</tr>
<tr>
<td>m-nom.</td>
<td>0.5</td>
<td>487</td>
<td>0.984 0.016</td>
<td>0.094 0.906</td>
</tr>
<tr>
<td>m-nom.</td>
<td>0.7</td>
<td>487</td>
<td>0.984 0.016</td>
<td>0.039 0.961</td>
</tr>
<tr>
<td>beams</td>
<td>0.1</td>
<td>489</td>
<td>0.998 0.002</td>
<td>0.834 0.166</td>
</tr>
<tr>
<td>beams</td>
<td>0.3</td>
<td>489</td>
<td>0.998 0.002</td>
<td>0.652 0.348</td>
</tr>
<tr>
<td>beams</td>
<td>0.5</td>
<td>489</td>
<td>0.998 0.002</td>
<td>0.464 0.536</td>
</tr>
<tr>
<td>beams</td>
<td>0.7</td>
<td>489</td>
<td>0.998 0.002</td>
<td>0.299 0.701</td>
</tr>
</tbody>
</table>

Table 10. Performance measured using standard metrics Exact Match (EM) and whitespace-tokenized F1 score against each question’s answer alias list. "(W)" indicates generation with the watermark. Data is 50,000 samples from the validation split of the unfiltered version of TriviaQA dataset. Questions are posed to the model in a zero-shot manner with no in-context demonstrations. Prompt template used: "The following is a trivia question with a single correct factual answer. Please provide the answer to the question. \{q\}, \{n\}Answer: ". Generation is performed using greedy decoding to maximize baseline/unwatermarked performance.

<table>
<thead>
<tr>
<th>Model</th>
<th>EM</th>
<th>EM (W)</th>
<th>F1</th>
<th>F1 (W)</th>
<th>( z )</th>
<th>( z ) (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>google/flan-ul2</td>
<td>0.374</td>
<td>0.336</td>
<td>0.415</td>
<td>0.378</td>
<td>-0.007</td>
<td>0.402</td>
</tr>
<tr>
<td>bigscience/bloomz</td>
<td>0.296</td>
<td>0.259</td>
<td>0.343</td>
<td>0.312</td>
<td>0.008</td>
<td>0.255</td>
</tr>
</tbody>
</table>

where the inner expectation is over uniformly random red/green partitions that satisfy \( k \in G \).

Now let’s bound the inner expectation on the right. Consider the helper function

\[
f_k(p) = \mathbb{E}_{G,R} \frac{\alpha p_k}{\sum_{i \in R} p_i + \alpha \sum_{i \in G} p_i},
\]

where \( G \) and \( R \) are sampled at random from the set of partitions that satisfy \( k \in G \). The value of \( f_k \) is invariant to permutations in the order of the indices \( \{p_i, i \neq k\} \). For this reason \( f(p) = \mathbb{E}_{\Pi} f(\Pi p) \), where \( \Pi \) is a random permutation that leaves \( p_k \) in place. Also, \( f_k \) is convex in \( p_{-k} \). By Jensen’s inequality,

\[
f(p) = \mathbb{E}_{\Pi} f(\Pi p) \geq f(\mathbb{E}_{\Pi} p).
\]
The expectation on the right involves a probability vector $\bar{p} \triangleq \mathbb{E}_i \Pi p$ in which $\bar{p}_i = (1 - p_0)/(N - 1)$ for $i \neq k$. We now have

$$f_k(p) \geq f_k(\bar{p}) = \frac{\alpha p_k}{N_R (1 - p_k)/(N - 1) + \alpha (N_G - 1)(1 - p_0)/(N - 1) + \alpha p_0}$$

$$= \frac{\alpha p_k (N - 1)}{(N_R + \alpha N_G - \alpha)(1 - p_k) + \alpha p_0 (N - 1)}$$

$$= \frac{\alpha p_k (N - 1)}{N_R + \alpha N_G - \alpha + (\alpha N - N_R - \alpha N_G)p_k}$$

$$= \frac{p_k}{N_R + \alpha N_G - \alpha + (\alpha N - N_R - \alpha N_G)p_k} \geq \frac{p_k}{N_R + \alpha N_G + (\alpha N_R - N_R)p_k}.$$ (9)

In the last step we used the fact that the numerator is larger than the denominator, and so adding $\alpha$ to the numerator and denominator results in a small decrease in the bound. Also, note that the fraction on the right side of (9) is strictly greater than 1 for any value of $p_k \in (0, 1)$ and $\alpha \geq 1$. For this reason the bound is never vacuous, as $f_k(p) > p_k$.

Now let $\gamma = N_G/N$. This simplifies the notation of our intermediate result to

$$f_k(p) \geq \frac{\alpha p_k}{(1 - \gamma) + \alpha \gamma + (\alpha - 1)(1 - \gamma)p_k}.$$ (10)

Using this expression to simplify (4) we get

$$\mathbb{E}_{k \leq N} \mathbb{E}_{G,R} \frac{\alpha p_k}{\sum_{i \in R} p_i + \alpha \sum_{i \in G} p_i} = \mathbb{E}_{k \leq N} f_k(p) \geq \frac{\alpha N^{-1}}{1 + (\alpha - 1)\gamma} S \left( p, \frac{(1 - \gamma)(\alpha - 1)}{1 + (\alpha - 1)\gamma} \right).$$

The probability of sampling a token from the green list is exactly $N_G$ times larger than an average green list probability. The probability of sampling from the green list is thus given by

$$N_G \mathbb{E}_{k \leq N} \mathbb{E}_{G,R} \frac{\alpha p_k}{\sum_{i \in R} p_i + \alpha \sum_{i \in G} p_i} \geq \frac{\gamma^\alpha}{1 + (\alpha - 1)\gamma} S \left( p, \frac{(1 - \gamma)(\alpha - 1)}{1 + (\alpha - 1)\gamma} \right).$$

It can be observed that the bound in Lemma F.1 is never vacuous; The probability of choosing a token from the green list is trivially at least $\gamma$, and for any combination of finite logits the bound in Lemma F.1 is strictly greater than this trivial lower bound. See the proof for a discussion of why.

Using this lemma, it’s now fairly straightforward to prove the main theorem.

**Proof.** Lemma F.1 bounds the probability of a single token being in the green list. To compute the total number of green list tokens in the sequence, we simply sum this bound over all the tokens to get.

$$\mathbb{E} |s|_G = \sum_t \frac{\gamma^\alpha}{1 + (\alpha - 1)\gamma} S^t = T \mathbb{E} \frac{\gamma^\alpha}{1 + (\alpha - 1)\gamma} S^t \geq \frac{\gamma^\alpha T}{1 + (\alpha - 1)\gamma} S^*.$$ 

where $S^{(t)}$ represents the entropy of the distribution of token $t$.

To get the variance bound, we begin by noting that the variance of a Bernoulli random variable with success probability $p$ is $p(1 - p)$. The expected number of green list tokens is a sum of independent random Bernoulli variables, each representing one token. These variables are not identically distributed, but rather each has a success probability given by Lemma F.1. The variance of the sum is the sum of the variances, which is

$$\text{Var} |s|_G = \sum_t \frac{\gamma^\alpha S^{(t)}}{1 + (\alpha - 1)\gamma} \left( 1 - \frac{\gamma^\alpha S^{(t)}}{1 + (\alpha - 1)\gamma} \right) = T \mathbb{E} \frac{\gamma^\alpha S^{(t)}}{1 + (\alpha - 1)\gamma} \left( 1 - \frac{\gamma^\alpha S^{(t)}}{1 + (\alpha - 1)\gamma} \right).$$
The expectation on the right contains a concave function of \( S^t \). By Jensen’s inequality, we can pass the expectation inside the function to get

\[
\text{Var} \ |s|_G \leq T \frac{\gamma \alpha E_t S^t}{1 + (\alpha - 1)\gamma} \left(1 - \frac{\gamma \alpha E_t S^t}{1 + (\alpha - 1)\gamma}\right).
\]

Finally, note that the probability of a token being in the green list is always at least \( \gamma \), regardless of the distribution coming from the language model. Lemma F.1 is never vacuous, and the success probability predicted by the Lemma is always at least \( \gamma \). If \( \gamma \geq 0.5 \), then the variance of each Bernoulli trial is at most the variance of a Bernoulli trial with success probability \( \gamma \), which is given by \( \gamma (1 - \gamma) \).

Plugging this into our bound gives

\[
\text{Var} \ |s|_G \leq T \frac{\gamma \alpha E_t S^t}{1 + (\alpha - 1)\gamma} \left(1 - \frac{\gamma \alpha E_t S^t}{1 + (\alpha - 1)\gamma}\right) \leq T \gamma (1 - \gamma).
\]

\( \square \)

**G. Bound on Perplexity of Watermarked Text**

**Theorem G.1.** Consider a sequence \( s^i, -N_p < i < T \). Suppose the (non-watermarked) language model produces a probability vector \( p(T) \) for the token at position \( T \). The watermarked model predicts the token at position \( T \) using modified probability vector \( \hat{p}(T) \). The expected perplexity of the \( T \)th token with respect to the randomness of the red list partition is

\[
\mathbb{E}_{W,B} \sum_k \hat{p}(T)_k \ln(p_k(T)) \leq (1 + (\alpha - 1)\gamma) P^*,
\]

where \( P^* = \sum_k p_k(T) \ln(p_k(T)) \) is the perplexity of the original model.

**Proof.** The probability of sampling token \( k \) from the modified distribution is

\[
\hat{p}_k = \frac{\alpha p_k}{\sum_{i \in R} p_i + \alpha \sum_{i \in G} p_i},
\]

where \( G \) and \( R \) are random partitions of the vocabulary indices. We can write this expected value as the sum of a contribution from the case in which \( k \in G \), and one in which \( k \in R \). We get

\[
\mathbb{E}_{G,R} \sum_{i \in R} \frac{\alpha p_k}{\sum_{i \in R} p_i + \alpha \sum_{i \in G} p_i} = \mathbb{E}_{G,R,k \in G} \frac{\alpha p_k}{\sum_{i \in R} p_i + \alpha \sum_{i \in G} p_i} \leq \gamma \alpha p_k + (1 - \gamma)p_k = (1 + (\alpha - 1)\gamma)p_k.
\]

The expected perplexity is then given by

\[
\mathbb{E}_{G,R} \sum_k \hat{p}(T)_k \ln(p_k(T)) = \sum_k \mathbb{E}_{G,R} \hat{p}(T)_k \ln(p_k(T)) \leq (1 + (\alpha - 1)\gamma) P^*.
\]

\( \square \)