
Unbiased Watermark for Large Language Models

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 The recent advancements in large language models (LLMs) have sparked a growing
2 apprehension regarding the potential misuse. One approach to mitigating this risk
3 is to incorporate watermarking techniques into LLMs, allowing for the tracking and
4 attribution of model outputs. This study examines a crucial aspect of watermarking:
5 how significantly watermarks impact the quality of model-generated outputs.
6 Previous studies have suggested a trade-off between watermark strength and output
7 quality. However, our research demonstrates that it is possible to integrate
8 watermarks without affecting the output probability distribution with appropriate
9 implementation. We refer to this type of watermark as an **unbiased watermark**.
10 This has significant implications for the use of LLMs, as it becomes impossible
11 for users to discern whether a service provider has incorporated watermarks or not.
12 Furthermore, the presence of watermarks does not compromise the performance
13 of the model in downstream tasks, ensuring that the overall utility of the language
14 model is preserved. Our findings contribute to the ongoing discussion around
15 responsible AI development, suggesting that unbiased watermarks can serve as
16 an effective means of tracking and attributing model outputs without sacrificing
17 output quality.

18 1 Introduction

19 In recent years, large language models (LLMs) [19, 39, 40] have become an indispensable tool for a
20 wide range of tasks, including text generation [27, 10], translation [7, 5], summarization [36], etc.
21 With the escalating misuse of LLMs, such as plagiarism, tracking the usage of text generated by
22 machines has become increasingly important. One viable method to monitor the usage of LLMs
23 is watermarking [20, 32, 59], which embeds imperceptible information within the generated text,
24 thereby allowing for efficient detection and tracking of the model’s potential abuse.

25 Watermarking techniques can serve multiple purposes, such as embedding ownership information
26 within the generated text to protect the intellectual property rights of the model. It can also help
27 mitigate potential harm caused by LLMs by monitoring where the model is being used and whether it
28 is being misused or abused.

29 A good watermarking method should not adversely affect the normal usage of the language model or
30 degrade the quality of the generated text. However, a prevailing belief holds that there is an inevitable
31 trade-off between the strength of the watermark and the quality of the output text. For instance,
32 recent work by Kirchenbauer et al. [32] introduced a method that augmented the logits of a randomly
33 selected set of "green" tokens. By tuning the "magnitude of logits adjustment", they demonstrated a
34 trade-off between watermark strength and text quality.

35 Our primary contribution is to challenge this conventional wisdom. We show that with the right
36 implementation, watermarking can be accomplished without affecting the output quality. We refer to
37 this particular type of watermark as an **unbiased watermark**. We approach the problem of output
38 quality degradation from the perspective of watermark detection. We posit that if the watermark

39 causes a decline in output quality, there should be a method to guess the presence of the watermark
40 based on the quality. Conversely, if the watermark cannot be detected, it implies that the output
41 quality remains unaffected. Specifically, we provide a proof that with a suitable implementation,
42 watermarking does not affect the output probability distribution. This has significant implications,
43 as users who do not have the private key are unable to discern whether a service provider has
44 applied watermarking to the model. Furthermore, the addition of watermarking does not affect
45 the performance of the generated text in any downstream tasks. **Our main contributions can be**
46 **summarized as follows:**

- 47 • We introduce *unbiased watermark*, an innovative family of watermark methods that guarantee the
48 non-degradation of text quality. In addition, we offer a comprehensive framework that facilitates
49 the design and detection of unbiased watermarks.
- 50 • We propose two innovative and practical watermarking techniques known as δ -reweight and
51 γ -reweight. Through extensive experimentation, we demonstrate that these techniques preserve
52 output quality in machine translation and text summarization tasks.
- 53 • We develop an advanced maximin variant of the original log-likelihood ratio test for watermark
54 detection. This novel detection method comes with theoretical guarantees, specifically an upper
55 bound on type I error, thus enhancing the reliability of watermark detection in language models.

56 2 Preliminary

57 In this section, we delve into the problem of watermarking in the context of LLMs. We begin by
58 setting up the problem and defining essential concepts.

59 **Problem Modeling:** We first introduce several notations to formalize the problem. Let Σ denote the
60 vocabulary set, which is the set of all possible tokens an LLM can generate in a single step. We then
61 define the set Σ^* as the collection of all possible strings of any length, including those of length zero.

62 An LLM generates a sequence of tokens conditioned on a given context. In a single step, the
63 probability of generating the next token $x_{n+1} \in \Sigma$ given the current context, x_1, x_2, \dots, x_n , can be
64 denoted as $P_M(x_{n+1} \mid x_1, x_2, \dots, x_n)$. The LLM operates in an autoregressive fashion, which means
65 the joint probability of generating multiple tokens x_{n+1}, \dots, x_{n+m} can be written as:

$$P_M(x_{n+1}, \dots, x_{n+m} \mid x_1, x_2, \dots, x_n) = \prod_{i=1}^m P_M(x_{n+i} \mid x_1, x_2, \dots, x_n, x_{n+1}, \dots, x_{n+i-1}).$$

66 For simplicity, we use the following notation: $P_M(\mathbf{x}_{n+1:n+m} \mid \mathbf{x}_{1:n})$, where $\mathbf{x}_{n+1:n+m} =$
67 $(x_{n+1}, \dots, x_{n+m}) \in \Sigma^*$.

68 In the context of watermarking, we introduce a service provider that holds a private key k from the key
69 space K . The key $k \in K$ is chosen at random from the prior distribution $P_K(k)$. The watermarked
70 output of the LLM follows distribution $P_{M,w}(x_{n+1} \mid x_1, x_2, \dots, x_n; k)$, which is conditioned on both
71 the key k and the context $\mathbf{x}_{1:n}$. Similarly, we use the notation $P_{M,w}(\mathbf{x}_{n+1:n+m} \mid \mathbf{x}_{1:n}; k)$ for the
72 probability of generating a sequence of tokens in a watermarked model.

73 **Objective.** Our goal is to devise a watermarking scheme that: a) is efficiently detectable by the
74 service provider; b) can't be detected by users and does not negatively impact the quality of the
75 output.

76 The reason we focus on the detection of watermarks by users is that it is closely related to the output
77 quality. If the watermark causes a degradation in the output quality, there should exist a method
78 to infer the presence of the watermark by examining the quality. Conversely, if the watermark is
79 undetectable, it implies that it does not impact the output quality.

80 From a statistical testing perspective, a watermark is considered strictly undetectable if the probability
81 distributions of the watermarked and non-watermarked outputs are identical. To capture this notion,
82 we define several desirable properties of watermarking schemes.

83 **Definition 1** (*n*-shot-undetectable). *For a fixed input sequence $\mathbf{a} \in \Sigma^*$, we say that watermarked*
84 *LLM and key prior pair $(P_{M,w}, P_K)$ is *n*-shot-undetectable compared to original LLM P_M if*

$$\prod_{i=1}^n P_M(\mathbf{x}^i \mid \mathbf{a}) = \sum_{k \in K} P_K(k) \prod_{i=1}^n P_{M,w}(\mathbf{x}^i \mid \mathbf{a}; k), \quad \text{for any } n \text{ number of strings } \mathbf{x}^i \in \Sigma^*.$$

85 **Definition 2** (downstream-invariant). We say the watermarked LLM and key prior pair $(P_{M,w}, P_K)$
 86 are invariant compared to original LLM P_M on downstream tasks iff

$$\mathbb{E}_{\mathbf{x} \sim P_{M,w}(\cdot | \mathbf{a}; k), k \sim P_K} [f(\mathbf{x})] = \mathbb{E}_{\mathbf{x} \sim P_M(\cdot | \mathbf{a})} [f(\mathbf{x})],$$

87 for any strings $\mathbf{x}, \mathbf{a} \in \Sigma^*$, and for any metric $f : \Sigma^* \rightarrow \mathbb{R}$.

88 Note that the one-shot-undetectable property implies the downstream invariant property. Interestingly,
 89 this implication does not require the n -shot-undetectable property for $n > 1$, which means a water-
 90 marking scheme that is one-shot-undetectable can still maintain the output quality for downstream
 91 tasks even if the user might discern the existence of the watermark through multiple generation
 92 requests.

93 In summary, we have outlined the preliminary concepts and objectives for developing a watermarking
 94 scheme for LLMs. We highlight the desired properties of n -shot-undetectability and downstream
 95 invariance, as they provide a rigorous theoretical guarantee of quality preservation and integrity in
 96 the deployment of watermark schema. In Section 4, we will present a watermark framework that is
 97 provably n -shot-undetectable for any given integer $n \geq 1$.

98 3 Warm up: undetectability in a simplified toy environment

99 In this subsection, we aim to prove the feasibility of undetectability in a highly simplified toy
 100 environment. This preliminary analysis serves as a foundation for understanding the more complex
 101 scenarios that follow.

102 **Settings.** Consider a service provider that offers a random number generation service. The service
 103 outputs a uniformly distributed random number in the set $\{0, 1\}$. The clean generation process can
 104 be represented as $P_M(x) = 1/2, \forall x \in \{0, 1\}$. We assume that the key k belongs to the set $\{0, 1\}$
 105 and is selected with equal probability. With the watermark added, the probability of the new output
 106 can be expressed as: $P_{M,w}(x | k) = \delta_k(x)$.

107 Recall that the one-shot-undetectable property can be represented as $P_M(x) = \sum_{k \in K} P_{M,w}(x |$
 108 $k)P_K(k)$. Suppose that a user can only make a single request to the service. If the user is unaware
 109 of the key, the user will be unable to distinguish whether the received result is watermarked or not.
 110 Therefore, in this simplified scenario, the undetectability of the watermark is achieved.

111 However, there is a considerable gap between this toy example and the practical implementation of
 112 watermarking in LLMs. Firstly, the symbol set Σ in LLMs is far more complex than the binary set
 113 $\{0, 1\}$, and the probability distribution is not uniform. Besides, the generation process in LLMs is
 114 autoregressive, which means that more than one symbol are generated iteratively. Furthermore, the
 115 toy example does not satisfy the n -shot-undetectable property for $n > 1$.

116 Despite these differences, this simple example provides essential insights that help in understanding
 117 the following sections where we address these challenges. The underlying principles of undetectability
 118 remain constant, while their application becomes more intricate in a more complex environment.

119 4 Watermarking with unbiased reweighting

120 In this section, we build upon the intuition from the previous section and extend the approach to
 121 LLMs' generation. The section is structured as follows: Section 4.1 introduces a fundamental
 122 mathematical tool for addressing the reweighting problem in general discrete probability distributions.
 123 Section 4.2 applies the reweighting technique to LLMs. Section 4.3 presents the final framework.

124 4.1 Distribution reweighting

125 In its most general form, we consider a random watermark code E and a reweight function $R_E :$
 126 $\Delta_\Sigma \rightarrow \Delta_\Sigma$, which depends on the random watermark code E . The set of all possible probability
 127 distributions on the symbol set Σ is denoted as Δ_Σ , which forms a simplex.

128 **Definition 3.** A **reweighting function** is a tuple (\mathcal{E}, P_E, R) where \mathcal{E} is called the watermark code
 129 space, P_E is a probability distribution on space \mathcal{E} , and R is a function $R : \mathcal{E} \times \Delta_\Sigma \rightarrow \Delta_\Sigma$.
 130 For a specific watermark code $E \in \mathcal{E}$, we denote the partially evaluated reweighting function as
 131 $R_E : \Delta_\Sigma \rightarrow \Delta_\Sigma$.

132 **Definition 4.** Given a random watermark code E and a reweighting function $R_E : \Delta_\Sigma \rightarrow \Delta_\Sigma$, we
 133 say that R is an **unbiased reweighting function** if and only if for all $P \in \Delta_\Sigma$, $\mathbb{E}_E[R_E(P)] = P$.

134 **4.1.1 Existing reweighting methods**

135 Kirchenbauer et al. [32] essentially comprise two reweighting methods in their work, but neither of
 136 them satisfies the unbiased property.

137 Both methods have \mathcal{E} as the set of mappings $f : \Sigma \rightarrow \{\text{red}, \text{green}\}$, such that f maps half of the
 138 tokens in Σ to ‘red’ and the other half to ‘green’, and P_E as a uniform distribution. Therefore, the
 139 random watermark code E assigns each symbol to either *red* or *green*. The ‘‘Hard Red List’’ method
 140 sets the probability of all red symbols to zero and renormalizes the probabilities of the remaining
 141 vocabulary. The second method is ‘‘Soft Red List’’ blocking, where they randomly select the same
 142 ‘‘Red List’’ as the first method and decrease the corresponding probability for red symbols by adding a
 143 constant δ to the logits of the green symbols, then apply softmax to obtain the final probabilities.

144 **4.1.2 Unbiased reweighting methods**

145 In this section, we present two reweighting methods that satisfy the unbiased property.

146 **δ -reweight:** Let the watermark code space \mathcal{E} be the interval $[0, 1]$, and let P_E be the uniform
 147 probability on \mathcal{E} . Leveraging *Inverse Transform Sampling*¹ [14], we can sample from distribution
 148 $P \in \Delta_\Sigma$ using a uniformly distributed random number in $[0, 1]$. Therefore, we have a mapping
 149 $\text{sampling}_P : \mathcal{E} \rightarrow \Sigma$. The δ -reweight just returns a delta distribution $R_E(P) = \delta_{\text{sampling}_P(E)}$.

150 It is important to note that while the reweighted distribution for each individual random event E
 151 is a delta distribution, the mean output token probabilities remain the original distribution P when
 152 considering the randomness of E .

153 **γ -reweight:** Let the watermark code space \mathcal{E} be the set of all bijective function between vocabularies
 154 set Σ and a set of indices $[\Sigma] = \{1, \dots, |\Sigma|\}$, where $|\Sigma|$ is the size of vocabularies set Σ . Essentially,
 155 any watermark code E is an indexing function for vocabularies set Σ , and is also equivalent to a total
 156 order on Σ . Let P_E be the uniform probability on \mathcal{E} , it is easy to sample a watermark code E by
 157 randomly shuffling the symbol list.

158 Assume the original distribution is $P_T(t) \in \Delta_\Sigma, \forall t \in \Sigma$. Given the watermark code $E : \Sigma \rightarrow [\Sigma]$,
 159 we construct auxiliary functions $F_I(i) = \sum_{t \in \Sigma} \mathbf{1}(E(t) \leq i) P_T(t)$, $F_S(s) = \max(2s - 1, 0)$,
 160 $F_{I'}(i) = F_S(F_I(i))$. The γ -reweight yields new distribution $P_{T'}(t) = F_{I'}(E(t)) - F_{I'}(E(t) - 1)$.

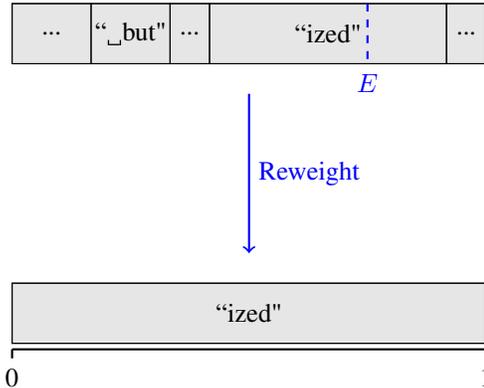


Figure 1: Illustration of δ -reweight.

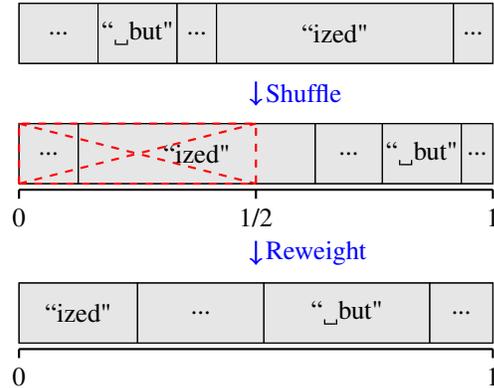


Figure 2: Illustration of γ -reweight.

161 We provide illustrations of the δ -reweight and γ -reweight methods in Figures 1 and 2. Each block
 162 represents a token, and the width represents the probability of that token, so the total length is 1. The
 163 left panel shows the δ -reweight method, where each individual random watermark code $E \in [0, 1]$
 164 uniformly sampled from interval $[0, 1]$ corresponds to a specific token according to the horizontal axis,
 165 and the reweighted distribution is just a δ distribution on that token, such that the selected token has 1
 166 probability, and all other vocabulary tokens have a probability of 0. The right panel demonstrates the
 167 γ -reweight method. First, the symbol set is shuffled. Then, the left half of the regions are rejected,
 168 and the remaining regions are amplified with a factor of 2.

169 Both methods are unbiased¹ when considering the randomness of the watermark code E . For δ -
 170 reweight, we can see that by noticing that the probability of returning a δ distribution on a token is

¹Detailed definition and rigorous proof can be found in Appendix B

171 just the original probability on that token, therefore the weighted average of all delta distributions is
 172 still the original probability. In the case of γ -reweight, although certain regions are rejected and the
 173 other regions are amplified, every token has the same probability to be in the rejected or amplified
 174 region, thus ensuring the unbiased property.

175 4.2 Reweighting for autoregressive model

176 The reweighting methods presented in the previous section can be applied to single token-generation
 177 directly. Given a prefix $\mathbf{x}_{1:n}$, the probability distribution for generating a new token without a
 178 watermark is denoted as $P_M(\cdot|\mathbf{x}_{1:n}) \in \Delta_\Sigma$. For a random watermark code E , we sample from a
 179 new distribution $P_{M,w}(\cdot|\mathbf{x}_{1:n}) = R_E(P_M(\cdot|\mathbf{x}_{1:n})) \in \Delta_\Sigma$. If the reweighting function is unbiased,
 180 we have $\mathbb{E}_E[R_E(P_M(\cdot|\mathbf{x}_{1:n}))] = P_M(\cdot|\mathbf{x}_{1:n})$. This ensures that, for an individual unaware of
 181 the watermark code, it is impossible to determine whether a new token is sampled directly from
 182 $P_M(\cdot|\mathbf{x}_{1:n})$ or from $P_{M,w}(\cdot|\mathbf{x}_{1:n}; E)$ for a random watermark E . However, if the watermark code is
 183 known, one can perform statistical hypothesis testing to determine the likelihood of a token being
 184 sampled from either distribution.

185 The main challenge now is constructing the watermark code E . Since the LLM generation task is
 186 autoregressive, multiple reweighting steps are required, with each step needing a watermark code E_i
 187 for reweighting the distribution of token x_i .

188 4.2.1 Independence of watermark codes

189 It is crucial that E_i values are independent to ensure the unbiased nature of the entire sequence, rather
 190 than just the single-token generation process.

191 **Theorem 5.** *Given an unbiased reweighting function (\mathcal{E}, P_E, R) , if E_i values are i.i.d. with the*
 192 *distribution P_E , we have: $\mathbb{E}_{E_1, \dots, E_n}[P_{M,w}(\mathbf{x}_{1:n}|\mathbf{a}_{1:m})] = P_M(\mathbf{x}_{1:n}|\mathbf{a}_{1:m})$.*

193 If the E_i values are not independent, we cannot guarantee that the generation probability of the entire
 194 sequence remains unbiased. As an extreme example, consider a case where all E_i values are identical.
 195 Referring to the random bit example in the previous section, assume that the correct distribution is
 196 a sequence where each token is a random 0 or 1 with equal probability. Identical E_i values would
 197 result in identical token outputs, ultimately producing sequences consisting solely of 0's or 1's, which
 198 is clearly biased.

199 4.2.2 Context code

200 To construct a large number of independent watermark codes E_i during watermarking and to know
 201 the used E_i values during watermark detection, we follow an approach similar to Kirchenbauer et al.
 202 [32] by combining the information from the prefix and a secret key to construct E_i .

203 For a single token generation process, given a prefix x_1, x_2, \dots, x_n , we consider an abstract context
 204 code space C and an abstract context code generation function $cc : \Sigma^* \rightarrow C$. Based on the prefix,
 205 we construct the context code $c_{n+1} = cc(x_1, x_2, \dots, x_n)$. Specific examples include using the entire
 206 prefix $c_{n+1} = (x_1, x_2, \dots, x_n)$, and using the m most recent prefixes $c_{n+1} = (x_{n-m+1}, \dots, x_n)$. Our
 207 comprehensive framework accommodates diverse context code generation approaches, particularly
 208 those that integrate error-correcting mechanisms to augment watermark resilience in the face of text
 209 manipulation attacks. Nevertheless, we refrain from delving into these strategies within the confines
 210 of this paper and consider it a subject for subsequent investigation.

211 The final watermark code is defined as $E_i = \hat{E}(c_i, k)$, using a watermark code generation function
 212 $\hat{E} : C \times K \rightarrow \mathcal{E}$.

213 **Definition 6.** *Given an unbiased reweighting function (\mathcal{E}, P_E, R) and a context code space C , an*
 214 *unbiased watermark code generation function is a tuple $(\mathcal{E}, P_E, R, C, K, P_K, \hat{E})$ that satisfies:*

- 215 1. *Unbiasedness:* $\mathbb{E}_{k \sim P_K}[R_{\hat{E}(c,k)}(P)] = P, \forall P \in \Delta_\Sigma, \forall c \in C$.
- 216 2. *Independence:* For any n distinct $c_1, \dots, c_n \in C$, the values $R_{\hat{E}(c_i,k)}(P)$ are mutually
 217 independent.

218 **Theorem 7.** *For any unbiased reweighting function and context code space, an unbiased watermark*
 219 *code generation function always exists.*

220 In practice, pseudorandom numbers can be used to implement the unbiased watermark code generation
 221 function in the above theorem. Specifically, the hash value $\text{hash}(c, k)$ can be used as a random seed

222 to sample E from P_E as an implementation of $E = \hat{E}(c, k)$. In this paper, we employ SHA-256 for
 223 hash function and a 1024-bit random bitstring as the key k .

224 An unbiased watermark code generation function ensures that watermark codes E_i are independent
 225 with each other if only their context codes are different. During the generation of a sequence,
 226 context codes may be repeated, although this is a rare event in practice. If c_i and c_j are equal,
 227 then E_i and E_j are also equal, violating the independence of E_i . A simple workaround is to skip
 228 reweighting for a token when encountering a previously used context code. In other words, we set
 229 $P_{M,w}(\cdot | \mathbf{a}_{1:m}, \mathbf{x}_{1:i-1}) = P_M(\cdot | \mathbf{a}_{1:m}, \mathbf{x}_{1:i-1})$ if the context code has appeared before.

230 4.3 Framework

Algorithm 1 Watermarking framework

```

1: Input: key for watermark  $k \in K$ , prompt  $\mathbf{a}_{1:m} \in \Sigma^*$ , generate length  $n \in \mathbb{N}$ , initial code
   history  $cch \in 2^C$ , context code function  $cc : \Sigma^* \rightarrow C$ , watermark code generation function
    $\hat{E} : C \times K \rightarrow \mathcal{E}$ , and reweighting function  $R : \mathcal{E} \times \Delta_\Sigma \rightarrow \Delta_\Sigma$ .
2: for  $t = 1, \dots, n$  do
3:    $P_i \leftarrow P_M(\cdot | \mathbf{a}_{1:m}, \mathbf{x}_{1:i-1})$  ▷ original distribution
4:    $c_i \leftarrow cc(\cdot | \mathbf{a}_{1:m}, \mathbf{x}_{1:i-1})$  ▷ context code
5:   if  $c_i \in cch$  then
6:      $Q_i \leftarrow P_i$  ▷ skip the reweighting
7:   else
8:      $cch \leftarrow cch \cup \{c_i\}$  ▷ record history
9:      $E_i \leftarrow \hat{E}(c_i, k)$  ▷ watermark code
10:     $Q_i \leftarrow R_{E_i}(P_i)$  ▷ reweighted distribution
11:    Sample the next token  $x_i$  using distribution  $Q_i$ 
12: return  $\mathbf{x}_{1:n}$ 

```

231 Integrating the tools discussed earlier, we present a general framework for watermarking here. The
 232 algorithm for this framework is outlined in Algorithm 1.

233 We note that our abstract framework requires the specification of two key components in order to be
 234 practically implemented: the unbiased reweight function R_E and the context code function cc .

235 5 Statistical hypothesis testing for watermark detection

236 In the previous section, we discussed the process of adding a watermark to a text based on a secret
 237 key k and a given prompt $\mathbf{a}_{1:m}$. The watermark-embedded text can be sampled from the distribution
 238 $P_{M,w}(\mathbf{x}_{1:n} | \mathbf{a}_{1:m}; k)$. In this section, we focus on the watermark detection task, which is the inverse
 239 problem of watermark embedding.

240 Given a text $\mathbf{x}_{1:n}$, the goal of watermark detection is to infer whether it is more likely to be generated
 241 from the unmarked distribution $P_M(\mathbf{x}_{1:n} | \mathbf{a}_{1:m})$ or the marked distribution $P_{M,w}(\mathbf{x}_{1:n} | \mathbf{a}_{1:m}; k)$.
 242 This problem can be formulated as a statistical hypothesis test between two competing hypotheses:
 243 H_0 , which posits that $\mathbf{x}_{1:n}$ follows the unmarked distribution, and H_1 , which posits that $\mathbf{x}_{1:n}$ follows
 244 the marked distribution.

245 5.1 Score-based testing

246 We focus on a particular kind of score-based testing, which assigns a score to each token in the text.
 247 The score can be interpreted as the confidence that the token was generated by the watermark model
 248 rather than the original model. Scores s_i can be computed based on $\mathbf{x}_{1:i}$, in accordance with the
 249 autoregressive manner of the generation process.

250 The total score S is given by $S = \sum_{i=1}^n s_i$. A threshold \hat{S} is set such that if $S < \hat{S}$, the null
 251 hypothesis H_0 is accepted, indicating insufficient evidence to conclude that the text contains a
 252 watermark. Otherwise, the null hypothesis is rejected. There are two types of error probabilities
 253 associated with this decision process: type I error, which is the probability of incorrectly rejecting

254 the null hypothesis under H_0 , denoted as $P_{H_0}(S \geq \hat{S})$, and type II error, which is the probability of
 255 incorrectly accepting the null hypothesis under H_1 , denoted as $P_{H_1}(S < \hat{S})$.

256 To derive theoretical results, we require the scores to have a specific property: under the null
 257 hypothesis H_0 , the exponential momentum of s_i is bounded, conditioned on the preceding context
 258 $\mathbf{x}_{1,i-1}$. This requirement leads to an upper bound on α , the type I error probability.

259 To derive theoretical results, we require that the scores have a particular property: the exponential
 260 moment of s_i under H_0 should be bounded, conditioned on the previous text $\mathbf{x}_{1,i-1}$. This requirement
 261 leads to an upper bound on the type I error rate.

262 **Theorem 8.** *Given a probability space (Ω, \mathcal{A}, P) and a Σ -valued stochastic process $x_i : 1 \leq i \leq n$,
 263 as well as an \mathbb{R} -valued stochastic process $s_i : 1 \leq i \leq n$, let $\mathcal{F}_i^x := \sigma(x_j \mid 1 \leq j \leq i)$ and
 264 $\mathcal{F}_i^s := \sigma(s_j \mid 1 \leq j \leq i)$ be the corresponding filtrations, where $\sigma(\cdot)$ denotes the σ -algebra
 265 generated by random variables. If $\mathcal{F}_i^s \subseteq \mathcal{F}_i^x$ and $\mathbb{E}[\exp(s_i) | \mathcal{F}_{i-1}^x] \leq 1$, then $P(\sum_{i=1}^n s_i \geq t) \leq e^{-t}$.*

266 Therefore, to ensure that the type I error probability has an upper bound α , we can set the threshold
 267 \hat{S} as $\hat{S} = -\log(\alpha)$. In the following, we discuss two special scores.

268 5.2 Log likelihood ratio (LLR) score

269 According to the Neyman-Pearson lemma, the likelihood ratio test is the most powerful test among
 270 all tests with the same type I error rate. Specifically, the log-likelihood ratio (LLR) score is defined as
 271 $s_i = \log \frac{P_{M,w}(x_i | \mathbf{a}_{1:m}, \mathbf{x}_{1:i-1}; k)}{P_M(x_i | \mathbf{a}_{1:m}, \mathbf{x}_{1:i-1})}$, and the total score becomes $S = \log \frac{P_{M,w}(\mathbf{x}_{1:n} | \mathbf{a}_{1:m}; k)}{P_M(\mathbf{x}_{1:n} | \mathbf{a}_{1:m})}$.

272 We now provide an optimization derivation of the above s_i to gain intuition and set the foundation
 273 for the maximin variant of the LLR score in the next section. Let $P_i = P_M(\cdot | \mathbf{a}_{1:m}, \mathbf{x}_{1:i-1})$,
 274 $Q_i = P_{M,w}(\cdot | \mathbf{a}_{1:m}, \mathbf{x}_{1:i-1}; k)$, and let $s_i = S_i(x_i) \in \mathbb{R}$ denote the score corresponding to different
 275 x_i . Note that P_i , Q_i , and S_i are all functions with signature $\Sigma \rightarrow \mathbb{R}$, therefore equivalent to vectors
 276 of dimension $|\Sigma|$. We can define the inner product as $\langle P_i, S_i \rangle = \sum_{x \in \Sigma} P_i(x) S_i(x)$.

277 The requirement $\mathbb{E}[\exp(s_i) | \mathcal{F}_{i-1}^x] \leq 1$ can be reformulated as $\langle P_i, \exp(S_i) \rangle \leq 1$, where the expo-
 278 nential function is applied element-wise. Instead of minimizing the type II error directly, we aim to
 279 maximize the average score under H_1 , i.e., $\langle Q_i, S_i \rangle$.

280 The optimization problem becomes $\max_{S_i} \langle Q_i, S_i \rangle$, s.t. $\langle P_i, \exp(S_i) \rangle \leq 1$. The optimal solution is
 281 given by $S_i(x) = \log \frac{Q_i(x)}{P_i(x)}$, which recovers the optimal log likelihood ratio score.

282 5.3 Maximin variant of LLR score

283 One major limitation of the LLR score described in the previous section is that when $Q_i(x) = 0$,
 284 $S_i(x) = -\infty$. This means that as long as a single token does not come from the watermark model
 285 $P_{M,w}$, the score becomes negative infinity, making it impossible to reject the null hypothesis H_0 .

286 A more general reason for this issue is that the watermark model $P_{M,w}$ used in the detection process
 287 may not exactly match the true distribution of the watermarked text. In practice, potential sources of
 288 discrepancy include editing (e.g., a text sampled from $P_{M,w}$ may undergo some degree of editing
 289 before being watermark detection) and imperfect estimation of the generation process (e.g., due to
 290 lack of knowledge of the exact prompt and temperature used during generation).

291 To address this problem, we consider a perturbed generation distribution. Instead of the original
 292 hypothesis H_1 , where $\mathbf{x}_{1:n}$ follows the watermark distribution $P_{M,w}$, we now assume that $\mathbf{x}_{1:n}$
 293 follows a distribution $P'_{M,w}$, which is similar to but not identical to $P_{M,w}$. Specifically, during the
 294 generation of each token, the total variation (TV) distance between Q'_i and Q_i is bounded by d .

295 The corresponding new optimization problem is

$$\max_{S_i} \min_{Q'_i \in \Delta_{\Sigma}, TV(Q'_i, Q_i) \leq d} \langle Q'_i, S_i \rangle, \quad \text{s.t. } \langle P_i, \exp(S_i) \rangle \leq 1.$$

296 Intuitively, the optimal solution for Q'_i in the inner optimization decreases $Q'_i(x)$ when $S_i(x)$ is large
 297 and increases $Q'_i(x)$ when $S_i(x)$ is small.

298 The computation of the maximin solution can be done efficiently in $\tilde{O}(|\Sigma|)$ time and the specific
 299 algorithm is shown in Appendix B.5.

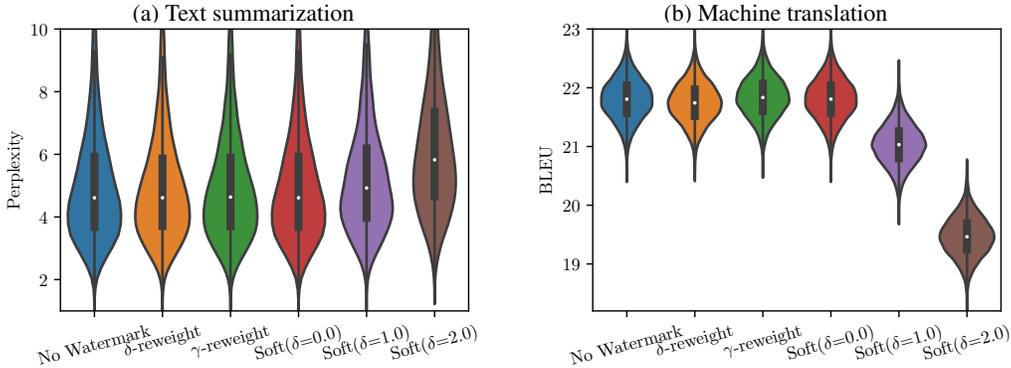


Figure 3: Distribution of perplexity of output for TS and BLEU score for MT.

300 It is important to note that the maximin variant of the LLR score is more robust than the standard
 301 LLR score, as it yields higher scores when the text has undergone some degree of editing. However,
 302 it is not specifically designed to defend against any attacks.

303 A hyperparameter $d \in [0, 1]$ that represent the perturbation strength is introduced in the score.
 304 Intuitively, if the text to be detected has undergone more editing and deviates further from the
 305 distribution $P_{M,w}$, d should be larger. In practice, we recommend using grid search to select the best
 306 value of d . Assuming there are A candidate values for d , corresponding to A different scores $s_i^{(a)}$
 307 ($1 \leq a \leq A$), we can modify Theorem 8 as follows.

308 **Theorem 9.** *Under the same conditions as Theorem 8, but with multiple scores $s_i^{(a)}$, we have*

$$P \left(\max_{1 \leq a \leq A} \left(\sum_{i=1}^n s_i^{(a)} \right) \geq t \right) \leq Ae^{-t}.$$

309 Thus, when using grid search, the final threshold should be adjusted as $\hat{S} = -\log(\alpha) + \log(A)$. This
 310 ensures that the upper bound of the type I error is still α .

311 6 Experiments

312 We evaluate the performance of our Unbiased Watermarks on two important applications of seq2seq
 313 models: text summarization (TS) and machine translation (MT). For the TS task, we use the
 314 BART-large model [37] and the CNN-DM [25] corpus as our testing dataset. The MT task involves
 315 translating English to Romanian, for which we employ the Multilingual BART (MBart) [37] model on
 316 the WMT’14 En-Ro corpus. For further details on the experiment setup, please refer to Appendix E.

Table 1: Performance of different watermarking methods on TS and MT. We use F1 scores of BERTScore and scale BERTScore and ROUGE-1 with a factor of 100.

	Text summarization			Machine translation	
	BERTScore \uparrow	ROUGE-1 \uparrow	Perplexity \downarrow	BERTScore \uparrow	BLEU \uparrow
No Watermark	32.70 ± 0.08	38.56 ± 0.09	5.024 ± 0.018	55.9 ± 0.3	21.8 ± 0.3
δ -reweight	32.71 ± 0.08	38.57 ± 0.09	5.022 ± 0.018	56.3 ± 0.3	21.7 ± 0.3
γ -reweight	32.69 ± 0.08	38.60 ± 0.09	5.019 ± 0.018	56.2 ± 0.3	21.8 ± 0.3
Soft($\delta=0.0$)	32.70 ± 0.08	38.56 ± 0.09	5.024 ± 0.018	55.9 ± 0.3	21.8 ± 0.3
Soft($\delta=1.0$)	32.35 ± 0.08	38.20 ± 0.09	5.313 ± 0.018	55.1 ± 0.3	21.0 ± 0.3
Soft($\delta=2.0$)	31.21 ± 0.08	37.17 ± 0.08	6.253 ± 0.022	53.8 ± 0.3	19.5 ± 0.3

317 Our primary focus is to compare the performance of our proposed unbiased watermarking methods
 318 including the δ -reweight and γ -reweight, with the soft-red-list method presented by Kirchenbauer
 319 et al. [32]. The strength of the watermark in the soft-red-list approach is controlled by a parameter δ .

320 The quality of output post-watermarking is presented in Table 1. We observed that the output quality
 321 remains unaffected by our unbiased watermark methods, both for the δ -reweight and γ -reweight,

Table 3: Text sampled from OPT-6.7B, with and without watermarks. For "No watermark" (NW), the score is computed based on δ -reweight. When watermarks are included, the corresponding reweighting function is used for computing score. The optimal perturbation strengths d obtained by grid search are 0.9, 0.0, 0.0 for three outputs respectively.

Prompt	What is a watermark? What's the purpose of it?	score	p-value ²
NW	Why don't you want it on there? I'm confused. I think he meant to say logo, since he wrote "watermark", so the first word must be a typo. Yes! Exactly typo. Sorry, English is not my native language. Thanks for the explanation!	0.30	8.14
δ -RW	It is supposed to be watermarking the pictures that you took with your phone. I think. So, so you can share your pictures and not take credit for them.	75.9	1.2e-32
γ -RW	A watermark is a small image or logo (often in square pixels) that is placed over the larger, original image. It serves primarily to distinguish copyright or ownership of large images (such as banners and logos) and, on rare occasion, to identify small images (such as thumbnail images for blog posts and pictures).	32.9	5.7e-14

irrespective of the task and metric. Conversely, the soft-red-list method, when $\delta = 0$, does not introduce any watermark and hence does not affect output quality. However, for $\delta > 0$, it significantly impairs the quality of output.

Figure 3 provides a more intuitive depiction of the score distributions. It is evident that our unbiased watermark methods not only ensure that the mean performance remains unaffected but also that the performance distribution is stable. Conversely, the soft-red-list method shows a notable performance decrease.

In terms of watermark detection, we compute score associated with each token. The mean and variance of score per token for TS and MT are presented in Table 2. As a heuristic, if the sum of the scores for all tokens in a sentence reaches 10, a p-value of less than 0.0005 is ensured. If the sum score hits 20, the p-value must be less than $3e-8$.

Table 2: Mean and variance of score per token for different reweighting methods and different tasks.

	Text summarization	Machine translation
δ -RW	0.8784 ± 1.4354	0.4192 ± 1.1361
γ -RW	0.2207 ± 0.3678	0.1056 ± 0.2916

Additionally, we provide an example of watermarking applied to a completion task in Table 3. It visually demonstrates the score distribution across tokens: positive scores are represented in green, and negative ones in red. The intensity of the color corresponds to the magnitude of the score, with darker shades representing larger absolute values.

7 Related work

The idea of watermarking text has been widely explored by many researchers [11, 31, 44, 45, 4, 28, 49, 43], even before the advent of large language models. Several techniques involve editing existing text to add a watermark, such as changing synonyms [54, 57, 9, 59, 66] or visually indistinguishable words [46], altering sentence structures [56, 55, 38], and employing neural networks [22, 23, 67].

Recent advancements in generative models have opened new possibilities for directly generating watermarked results. Two relevant works in this domain are by Kirchenbauer et al. [32] and Aaronson [1]. Due to space constraints, we moved the in-depth analysis and other related work to Section A.

8 Conclusion

Overall, this paper provides a novel framework of watermarking for language models, demonstrating that it is possible to use watermark to protect intellectual property and monitor potential misuse without compromising the quality of the generated text. This research serves as a valuable foundation for future work in the field of watermarking for large language models.

²This is an upper bound computed based on Theorem 9. The upper bound could be larger than 1, but this does not necessarily imply that the p-value exceeds 1.

354 References

- 355 [1] Scott Aaronson. My ai safety lecture for ut effective altruism. November 2022. URL [https://](https://scottaaronson.blog/?p=6823)
356 scottaaronson.blog/?p=6823.
- 357 [2] Sahar Abdelnabi and Mario Fritz. Adversarial watermarking transformer: Towards tracing text provenance
358 with data hiding. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 121–140. IEEE, 2021.
- 359 [3] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness
360 into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium*,
361 pages 1615–1631, 2018.
- 362 [4] Mikhail J Atallah, Victor Raskin, Michael Crogan, Christian Hempelmann, Florian Kerschbaum, Dina
363 Mohamed, and Sanket Naik. Natural language watermarking: Design, analysis, and a proof-of-concept
364 implementation. In *Information Hiding: 4th International Workshop, IH 2001 Pittsburgh, PA, USA, April*
365 *25–27, 2001 Proceedings 4*, pages 185–200. Springer, 2001.
- 366 [5] Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham,
367 Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu
368 Pal, Matt Post, and Marcos Zampieri. Findings of the 2019 conference on machine translation (WMT19).
369 In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*,
370 pages 1–61, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/
371 W19-5301.
- 372 [6] Franziska Boenisch. A systematic review on model watermarking for neural networks. *Frontiers in big*
373 *Data*, 4:729663, 2021.
- 374 [7] Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang,
375 Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post,
376 Raphael Rubino, Lucia Specia, and Marco Turchi. Findings of the 2017 conference on machine translation
377 (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214. Copenhagen,
378 Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4717.
- 379 [8] Nicholas Boucher, Iliia Shumailov, Ross Anderson, and Nicolas Papernot. Bad characters: Imperceptible
380 nlp attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1987–2004. IEEE, 2022.
- 381 [9] Yuei-Lin Chiang, Lu-Ping Chang, Wen-Tai Hsieh, and Wen-Chih Chen. Natural language watermarking
382 using semantic substitution for chinese text. In *Digital Watermarking: Second International Workshop,*
383 *IWDW 2003, Seoul, Korea, October 20-22, 2003. Revised Papers 2*, pages 129–140. Springer, 2004.
- 384 [10] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi
385 Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv*
386 *preprint arXiv:2210.11416*, 2022.
- 387 [11] Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker. *Digital watermarking and*
388 *steganography*. Morgan kaufmann, 2007.
- 389 [12] Evan Crothers, Nathalie Japkowicz, and Herna Viktor. Machine generated text: A comprehensive survey
390 of threat models and detection methods. *arXiv preprint arXiv:2210.07321*, 2022.
- 391 [13] Falcon Z Dai and Zheng Cai. Towards near-imperceptible steganographic text. *arXiv preprint*
392 *arXiv:1907.06679*, 2019.
- 393 [14] Luc Devroye. *Non-Uniform Random Variate Generation*. Springer New York, 1986.
- 394 [15] Tina Fang, Martin Jaggi, and Katerina Argyraki. Generating steganographic text with lstms. *arXiv preprint*
395 *arXiv:1705.10742*, 2017.
- 396 [16] Evgeniy Gabrilovich and Alex Gontmakher. The homograph attack. *Communications of the ACM*, 45(2):
397 128, 2002.
- 398 [17] Margherita Gambini, Tiziano Fagni, Fabrizio Falchi, and Maurizio Tesconi. On pushing deepfake tweet
399 detection capabilities to the limits. In *14th ACM Web Science Conference 2022*, pages 154–163, 2022.
- 400 [18] Riley Goodside. There are adversarial attacks for that proposal as well — in particular, generating
401 with emojis after words and then removing them before submitting defeats it,. January 2023. URL
402 <https://twitter.com/goodside/status/1610682909647671306>.
- 403 [19] Google. Palm-2-llm. <https://blog.google/technology/ai/google-palm-2-ai-large-language-model/>, 2023.

- 404 [20] Chenxi Gu, Chengsong Huang, Xiaoqing Zheng, Kai-Wei Chang, and Cho-Jui Hsieh. Watermarking
405 pre-trained language models with backdoorling. *arXiv preprint arXiv:2210.07543*, 2022.
- 406 [21] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine
407 learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.
- 408 [22] Xuanli He, Qiongkai Xu, Lingjuan Lyu, Fangzhao Wu, and Chenguang Wang. Protecting intellectual
409 property of language generation apis with lexical watermark. In *Proceedings of the AAAI Conference on*
410 *Artificial Intelligence*, volume 36, pages 10758–10766, 2022.
- 411 [23] Xuanli He, Qiongkai Xu, Yi Zeng, Lingjuan Lyu, Fangzhao Wu, Jiwei Li, and Ruoxi Jia. Cater: Intellectual
412 property protection on text generation apis via conditional watermarks. *arXiv preprint arXiv:2209.08773*,
413 2022.
- 414 [24] James N Helfrich and Rick Neff. Dual canonicalization: An answer to the homograph attack. In *2012*
415 *eCrime Researchers Summit*, pages 1–10. IEEE, 2012.
- 416 [25] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman,
417 and Phil Blunsom. Teaching machines to read and comprehend. In Corinna Cortes, Neil D. Lawrence,
418 Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing*
419 *Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015,*
420 *Montreal, Quebec, Canada*, pages 1693–1701, 2015.
- 421 [26] Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. Automatic detection of
422 generated text is easiest when humans are fooled. *arXiv preprint arXiv:1911.00650*, 2019.
- 423 [27] Srinivasan Iyer, Xi Victoria Lin, Ramakanth Pasunuru, Todor Mihaylov, Daniel Simig, Ping Yu, Kurt
424 Shuster, Tianlu Wang, Qing Liu, Punit Singh Koura, et al. Opt-impl: Scaling language model instruction
425 meta learning through the lens of generalization. *arXiv preprint arXiv:2212.12017*, 2022.
- 426 [28] Zunera Jalil and Anwar M Mirza. A review of digital watermarking techniques for text documents. In
427 *2009 International Conference on Information and Multimedia Technology*, pages 230–234. IEEE, 2009.
- 428 [29] Ganesh Jawahar, Muhammad Abdul-Mageed, and Laks VS Lakshmanan. Automatic detection of machine
429 generated text: A critical survey. *arXiv preprint arXiv:2011.01314*, 2020.
- 430 [30] Hengrui Jia, Christopher A Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled
431 watermarks as a defense against model extraction. In *USENIX Security Symposium*, pages 1937–1954,
432 2021.
- 433 [31] Nurul Shamimi Kamaruddin, Amirrudin Kamsin, Lip Yee Por, and Hameedur Rahman. A review of text
434 watermarking: theory, methods, and applications. *IEEE Access*, 6:8011–8028, 2018.
- 435 [32] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark
436 for large language models. *arXiv preprint arXiv:2301.10226*, 2023.
- 437 [33] Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. Paraphrasing evades
438 detectors of ai-generated text, but retrieval is an effective defense. *arXiv preprint arXiv:2303.13408*, 2023.
- 439 [34] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. How to prove your model belongs to you: A
440 blind-watermark based framework to protect intellectual property of dnn. In *Proceedings of the 35th*
441 *Annual Computer Security Applications Conference*, pages 126–137, 2019.
- 442 [35] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches*
443 *out*, pages 74–81, 2004.
- 444 [36] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. In *Proceedings of the*
445 *2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint*
446 *Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China,
447 November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1387.
- 448 [37] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and
449 Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation. *Transactions of the*
450 *Association for Computational Linguistics*, 8:726–742, 2020.
- 451 [38] Hasan Mesut Meral, Bulent Sankur, A Sumru Ozsoy, Tunga Gungor, and Emre Sevinç. Natural language
452 watermarking via morphosyntactic alterations. *Computer Speech & Language*, 23(1):107–125, 2009.
- 453 [39] OpenAI. Chatgpt. <https://openai.com/blog/chatgpt>, 2023a.

- 454 [40] OpenAI. Gpt-4 technical report. *arXiv*, 2023b.
- 455 [41] Luca Pajola and Mauro Conti. Fall of giants: How popular text-based mlaas fall against a simple evasion
456 attack. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 198–211. IEEE,
457 2021.
- 458 [42] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation
459 of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational
460 Linguistics*, pages 311–318, 2002.
- 461 [43] Fabien AP Petitcolas, Ross J Anderson, and Markus G Kuhn. Information hiding-a survey. *Proceedings of
462 the IEEE*, 87(7):1062–1078, 1999.
- 463 [44] Christine I Podilchuk and Edward J Delp. Digital watermarking: algorithms and applications. *IEEE signal
464 processing Magazine*, 18(4):33–46, 2001.
- 465 [45] Vidyasagar M Potdar, Song Han, and Elizabeth Chang. A survey of digital image watermarking techniques.
466 In *INDIN'05. 2005 3rd IEEE International Conference on Industrial Informatics, 2005.*, pages 709–716.
467 IEEE, 2005.
- 468 [46] Stefano Giovanni Rizzo, Flavio Bertini, and Danilo Montesi. Fine-grain watermarking for intellectual
469 property protection. *EURASIP Journal on Information Security*, 2019:1–20, 2019.
- 470 [47] Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. Can
471 ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*, 2023.
- 472 [48] M Hassan Shirali-Shahreza and Mohammad Shirali-Shahreza. A new synonym text steganography. In
473 *2008 international conference on intelligent information hiding and multimedia signal processing*, pages
474 1524–1526. IEEE, 2008.
- 475 [49] Katzenbeisser Stefan, A Petitcolas Fabien, et al. Information hiding techniques for steganography and
476 digital watermarking, 2000.
- 477 [50] Yuchen Sun, Tianpeng Liu, Panhe Hu, Qing Liao, Shouling Ji, Nenghai Yu, Deke Guo, and Li Liu. Deep
478 intellectual property: A survey. *arXiv preprint arXiv:2304.14613*, 2023.
- 479 [51] Reuben Tan, Bryan A Plummer, and Kate Saenko. Detecting cross-modal inconsistency to defend against
480 neural fake news. *arXiv preprint arXiv:2009.07698*, 2020.
- 481 [52] Ruixiang Tang, Yu-Neng Chuang, and Xia Hu. The science of detecting llm-generated texts. *arXiv preprint
482 arXiv:2303.07205*, 2023.
- 483 [53] Yi Tay, Dara Bahri, Che Zheng, Clifford Brunk, Donald Metzler, and Andrew Tomkins. Reverse engineering
484 configurations of neural text generation models. *arXiv preprint arXiv:2004.06201*, 2020.
- 485 [54] Mercan Topkara, Cuneyt M Taskiran, and Edward J Delp III. Natural language watermarking. In *Security,
486 Steganography, and Watermarking of Multimedia Contents VII*, volume 5681, pages 441–452. SPIE, 2005.
- 487 [55] Mercan Topkara, Giuseppe Riccardi, Dilek Hakkani-Tür, and Mikhail J Atallah. Natural language
488 watermarking: Challenges in building a practical system. In *Security, Steganography, and Watermarking of
489 Multimedia Contents VIII*, volume 6072, pages 106–117. SPIE, 2006.
- 490 [56] Mercan Topkara, Umut Topkara, and Mikhail J Atallah. Words are not enough: sentence level natural
491 language watermarking. In *Proceedings of the 4th ACM international workshop on Contents protection
492 and security*, pages 37–46, 2006.
- 493 [57] Umut Topkara, Mercan Topkara, and Mikhail J Atallah. The hiding virtues of ambiguity: quantifiably
494 resilient watermarking of natural language text through synonym substitutions. In *Proceedings of the 8th
495 workshop on Multimedia and security*, pages 164–174, 2006.
- 496 [58] Honai Ueoka, Yugo Murawaki, and Sadao Kurohashi. Frustratingly easy edit-based linguistic steganography
497 with a masked language model. *arXiv preprint arXiv:2104.09833*, 2021.
- 498 [59] Ashish Venugopal, Jakob Uszkoreit, David Talbot, Franz Josef Och, and Juri Ganitkevitch. Watermarking
499 the outputs of structured prediction with an application in statistical machine translation. In *Proceedings of
500 the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1363–1372, 2011.
- 501 [60] Hong Wang, Xuan Luo, Weizhi Wang, and Xifeng Yan. Bot or human? detecting chatgpt imposters with a
502 single question. *arXiv preprint arXiv:2305.06424*, 2023.

- 503 [61] Alex Wilson and Andrew D Ker. Avoiding detection on twitter: embedding strategies for linguistic
504 steganography. Society of Photo-optical Instrumentation Engineers, 2016.
- 505 [62] Alex Wilson, Phil Blunsom, and Andrew D Ker. Linguistic steganography on twitter: hierarchical language
506 modeling with manual interaction. In *Media Watermarking, Security, and Forensics 2014*, volume 9028,
507 pages 9–25. SPIE, 2014.
- 508 [63] Alex Wilson, Phil Blunsom, and Andrew Ker. Detection of steganographic techniques on twitter. In
509 *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages
510 2564–2569, 2015.
- 511 [64] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric
512 Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface’s transformers: State-of-the-art
513 natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.
- 514 [65] Max Wolff and Stuart Wolff. Attacking neural text detectors. *arXiv preprint arXiv:2002.11768*, 2020.
- 515 [66] Xi Yang, Jie Zhang, Kejiang Chen, Weiming Zhang, Zehua Ma, Feng Wang, and Nenghai Yu. Tracing text
516 provenance via context-aware lexical substitution. In *Proceedings of the AAAI Conference on Artificial
517 Intelligence*, volume 36, pages 11613–11621, 2022.
- 518 [67] KiYoon Yoo, Wonhyuk Ahn, Jiho Jang, and Nojun Kwak. Robust natural language watermarking through
519 invariant features. *arXiv preprint arXiv:2305.01904*, 2023.
- 520 [68] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin
521 Choi. Defending against neural fake news. *Advances in neural information processing systems*, 32, 2019.
- 522 [69] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy.
523 Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on
524 Asia Conference on Computer and Communications Security*, pages 159–172, 2018.
- 525 [70] Xuandong Zhao, Yu-Xiang Wang, and Lei Li. Protecting language generation models via invisible
526 watermarking. *arXiv preprint arXiv:2302.03162*, 2023.
- 527 [71] Zachary M Ziegler, Yuntian Deng, and Alexander M Rush. Neural linguistic steganography. *arXiv preprint
528 arXiv:1909.01496*, 2019.

529 **A Related works**

530 **A.1 Text watermarking**

531 The idea of watermarking text has been widely explored by many researchers [11, 31, 44, 45, 4, 28,
532 49, 43], even before the advent of large language models. Several techniques involve editing existing
533 text to add a watermark, such as changing synonyms [54, 57, 9, 59, 66] or visually indistinguishable
534 words [46], altering sentence structures [56, 55, 38], and employing neural networks [22, 23, 67].

535 Recent advancements in generative models have opened new possibilities for directly generating
536 watermarked results. Two relevant works in this domain are by Kirchenbauer et al. [32] and Aaronson
537 [1]. Kirchenbauer et al.’s pioneering work, which uses the previous context to generate watermarked
538 tokens, heavily influences our approach. However, their watermarking technique can introduce bias
539 to the output, leading to performance degradation. Our work addresses this limitation by applying
540 unbiased reweighting and recording context code history.

541 Aaronson [1] have talked about using a pseudo-random cryptographic function for watermarking,
542 but the details are not disclosed, making it challenging to conduct a direct comparison. Aaronson’s
543 “cryptographic pseudorandom function” could be a special case of reweighting function in this paper.
544 However, in his blog, there is no apparent structure akin to “context code history”, a mechanism
545 that plays a crucial role in our work to ensure n-shot-undetectability. Therefore, it remains uncertain
546 whether Aaronson’s technique could offer a similar theoretical guarantee of n-shot-undetectability
547 as ours. Additionally, it is not clear if their method provides an upper bound on type I error, like
548 Theorem 8.

549 **A.2 Attacks on watermarks**

550 Alongside the development of watermarking technologies, various methods to modify and remove
551 these watermarks and their countermeasures have also been explored. These include attacks based on
552 invisible characters and homoglyphs [16, 24, 41, 8], generative attacks such as those that prompted
553 the model to change its output in a predictable and easily reversible way [32], and specific instances
554 such as the emoji attack [18], and paraphrasing attacks [47, 33].

555 **A.3 Steganography in text**

556 Steganography hides information in text primarily for secret communication. It bears similarities to
557 watermarking in that it seeks to conceal information. However, while watermarking only needs to
558 detect the presence of a watermark, steganography must recover all embedded information. Many
559 approaches have tried to edit existing text, through rule-based transformations [62, 63, 61], synonym-
560 based methods [48], and more recently, neural network-based methods [2, 58]. Information can also
561 be embedded directly during generation [15, 13, 71].

562 **A.4 Watermarking models**

563 Watermarking has also been applied to models themselves to protect intellectual property rights and
564 to guard against model stealing or extraction [30, 6, 70]. The aim here is to gather evidence through
565 inference services [34, 69] and can be accomplished by adding backdoors to models [3, 21, 20].
566 While they are similar to text watermarking in that they embed information without impacting fair
567 use, the focus is on tracing the model rather than the text.

568 **A.5 Detecting machine-generated text**

569 The objective of detecting machine-generated text lies in discerning whether a given text has been
570 produced by an algorithm or written by a human. Such detection is crucial to prevent misuse and
571 a substantial body of research has explored this area [68, 26, 12, 29, 51, 53, 52, 60]. However, the
572 task has become increasingly challenging due to the continual improvement in language models and
573 the advent of adversarial attacks [17, 65, 47]. The difference between this and text watermarking is
574 that watermarking is employed to differentiate whether a text is generated by a particular model or
575 provider, yet the detection of machine-generated text is not concerned with a specific model.

576 **B Detailed definition and additional proofs**

577 **B.1 Detailed definition and additional proofs for Section 4.1**

578 **Definition 10** (hard/soft-red-list reweighting [32]). *Given two hyper-parameters $0 \leq \gamma \leq 1$ and*
 579 *$\delta \geq 0$, let the watermark code space be $\mathcal{E} = \{E \in \{0, 1\}^\Sigma \mid |E^{-1}(1)| = \lfloor \gamma |\Sigma| \rfloor\}$, such that f maps*
 580 *γ -portion of the tokens in Σ to 1 (which interpreted as “green”) and the other portion to 0 (which*
 581 *interpreted as “red”), and let P_E to be the uniform distribution on space \mathcal{E} . For any watermark code*
 582 *E , and for any token distribution $P \in \Delta_\Sigma$, the output distribution of the hard-red-list reweighting*
 583 *function on a token $t \in \Sigma$ is defined by $R_E(P)(t) = \frac{E(t)P(t)}{\sum_{t \in \Sigma} E(t)P(t)}$ assuming $\sum_{t \in \Sigma} E(t)P(t) > 0$.*
 584 *The soft-red-list reweighting function is defined by $R_E(P)(t) = \frac{\exp\{\log P(t) + \delta E(t)\}}{\sum_{t \in \Sigma} \exp\{\log P(t) + \delta E(t)\}}$, where*
 585 *$\delta > 0$ is a fixed constant.*

586 **Theorem 11.** *Hard-red-list and soft-red-list reweighting functions are biased.*

Proof. We first show the hard-red-list reweighting is biased. For $\gamma = 0.5$, consider $\Sigma = \{a, b\}$, $P(a) = 0.9, P(b) = 0.1$, we have

$$R_E(P)(a) = \frac{1}{2} \times \frac{P(a)}{P(a)} + 0 \times \frac{0}{P(b)} = 0.5 \neq 0.9 = P(a).$$

We then show the soft-red-list reweighting is biased. For $\gamma = 0.5$, consider $\Sigma = \{a, b\}$, $P(a) = 0.9, P(b) = 0.1$, we have

$$R_E(P)(a) = \frac{1}{2} \times \frac{e^\delta P(a)}{e^\delta P(a) + P(b)} + \frac{1}{2} \times \frac{P(a)}{P(a) + e^\delta P(b)}.$$

587 It is easy to verify that for any $\delta > 0$, we have $R_E(P)(a) < P(a)$.

588 Thus, hard/soft-red-list reweighting are both biased. □

589 **Definition 12** (δ -reweight). *Let the watermark code space \mathcal{E} be the interval $[0, 1]$, and let E be*
 590 *uniformly distributed on \mathcal{E} . Given an arbitrary token distribution $P \in \Delta_\Sigma$, let B be a bijection*
 591 *between Σ and $[\Sigma]$, we construct a cumulative density function of P w.r.t. B by $F_P(t; B) =$
 592 $\sum_{t' \in \Sigma} \mathbf{1}(B(t') \leq B(t))P(t'), \forall t \in \Sigma$. Then we can define a mapping $\text{sampling}_P : \mathcal{E} \rightarrow \Sigma$,*

$$\text{sampling}_P(E) = B^{-1}(I(E)),$$

593 *where*

$$I(E) = \min_t B(t) \text{ s.t. } E \leq F_P(t; B),$$

594 *The δ -reweight function is defined by $R_E(P) := \delta_{\text{sampling}_P(E)}$.*

Definition 13 (γ -reweight). *Let the watermark code space \mathcal{E} be the set of all bijective function*
between vocabularies set Σ and a set of indices $[\Sigma] = \{1, \dots, |\Sigma|\}$, where $|\Sigma|$ is the size of
vocabularies set Σ . Assume the original distribution is $P_T(t) \in \Delta_\Sigma, \forall t \in \Sigma$. Given the watermark
code $E : \Sigma \rightarrow [\Sigma]$, we define

$$A_E(i) := \max \left\{ 2 \left(\sum_{t \in \Sigma} \mathbf{1}(E(t) \leq i) P_T(t) \right) - 1, 0 \right\},$$

595 *where $\mathbf{1}(E(t) \leq i) = 1$ when $E(t) \leq i$ otherwise $\mathbf{1}(E(t) \leq i) = 0$. We define $P_{T'(E)}(t) :=$
 596 $A_E(E(t)) - A_E(E(t) - 1)$. It's easy to verify $P_{T'(E)}$ is a distribution by $\forall t \in \Sigma, P_{T'(E)}(t) \geq 0$
 597 *and $\sum_{t \in \Sigma} P_{T'(E)}(t) = 1$. Thus, γ -reweight function is defined by $R_E(P_T) := P_{T'(E)}$.**

598 **Theorem 14.** *Both δ -reweight and γ -reweight are unbiased reweighting functions.*

599 *Proof.* According to Definition 4, we need to show $\mathbb{E}_E[R_E(P)] = P$ for arbitrary $P \in \Delta_\Sigma$.

600 For δ -reweight, we have $R_E(P) = \delta_{\text{sampling}_P(E)}$ and E is uniformly distributed on $[0, 1]$. Thus, we
 601 only need to show $\forall t \in \Sigma, \mathbb{E}_E[\delta_{\text{sampling}_P(E)}(t)] = P(t)$.

$$\begin{aligned} \mathbb{E}_E[\delta_{\text{sampling}_P(E)}(t)] &= \int_0^1 \mathbf{1}(\text{sampling}_P(e) = t) de, \\ &= \int_0^1 \mathbf{1}(I(e) = B(t)) de, \\ &= \begin{cases} F_P(t; B) - F_P(B^{-1}(B(t) - 1); B) & B(t) > 1 \\ F_P(t; B) & B(t) = 1 \end{cases} \\ &= P(t). \end{aligned} \tag{1}$$

602 For γ -reweight, we need to show $\forall t \in \Sigma, \mathbb{E}_E[R_E(P_T)(t)] = P_T(t)$

$$\begin{aligned} \mathbb{E}_E[R_E(P_T)(t)] &= \mathbb{E}_E[P_{T'(E)}(t)] \\ &= \mathbb{E}_E[A_E(E(t)) - A_E(E(t) - 1)]. \end{aligned} \tag{2}$$

Denoted by $g_E(i) = 2 \left(\sum_{t' \in \Sigma} \mathbf{1}(E(t') \leq i) P_T(t') \right) - 1$. $\forall E \in \mathcal{E}$, we consider the reserved order E^r of E , we have $E(t) + E^r(t) = n + 1$ and

$$g_E(E(t)) + g_{E^r}(E^r(t) - 1) = 2 \left(\sum_{t' \in \Sigma} [\mathbf{1}(E(t') \leq E(t)) + \mathbf{1}(E(t') \geq E(t) + 1)] P_T(t') \right) - 2 = 0.$$

603 So we have

$$\begin{aligned} &A_E(E(t)) - A_E(E(t) - 1) + A_{E^r}(E^r(t)) - A_{E^r}(E^r(t) - 1) \\ &= \max\{g_E(E(t)), 0\} - \max\{g_E(E(t) - 1), 0\} + \max\{g_{E^r}^r(E^r(t)), 0\} - \max\{g_{E^r}^r(E^r(t) - 1), 0\} \\ &= g_E(E(t)) \mathbf{1}(g_E(E(t)) > 0) - g_{E^r}(E^r(t) - 1) \mathbf{1}(g_{E^r}(E^r(t) - 1) > 0) + \\ &\quad g_{E^r}(E^r(t)) \mathbf{1}(g_{E^r}(E^r(t)) > 0) - g_E(E(t) - 1) \mathbf{1}(g_E(E(t) - 1) > 0) \\ &= g_E(E(t)) \mathbf{1}(g_E(E(t)) > 0) + g_E(E(t)) \mathbf{1}(g_E(E(t)) < 0) - \\ &\quad g_E(E(t) - 1) \mathbf{1}(g_E(E(t) - 1) < 0) - g_E(E(t) - 1) \mathbf{1}(g_E(E(t) - 1) > 0) \\ &= g_E(E(t)) - g_E(E(t) - 1) \\ &= 2P_T(t), \end{aligned} \tag{3}$$

604 which yields

$$\begin{aligned} \mathbb{E}_E[R_E(P_T)](t) &= \mathbb{E}_E[A_E(E(t)) - A_E(E(t) - 1)]. \\ &= \frac{1}{2} (\mathbb{E}_E[A_E(E(t)) - A_E(E(t) - 1)] + \mathbb{E}_{E^r}[A_{E^r}(E^r(t)) - A_{E^r}(E^r(t) - 1)]). \\ &= \frac{1}{2} \mathbb{E}_E[2P_T(t)] \\ &= P_T(t). \end{aligned} \tag{4}$$

605 □

606 **B.2 Additional proofs for Section 4.2**

607 *Proof of Theorem 5.* We have

$$\begin{aligned} &\mathbb{E}_{E_1, \dots, E_n} [P_{M,w}(\mathbf{x}_{1:n} | \mathbf{a}_{1:m})] \\ &= \mathbb{E}_{E_1, \dots, E_{n-1}} [\mathbb{E}_{E_n} [P_{M,w}(\mathbf{x}_{1:n} | \mathbf{a}_{1:m})]] \\ &= \mathbb{E}_{E_1, \dots, E_{n-1}} [\mathbb{E}_{E_n} [P_{M,w}(x_n | \mathbf{a}_{1:m}, \mathbf{x}_{1:n-1})] P_{M,w}(\mathbf{x}_{1:n-1} | \mathbf{a}_{1:m})] \\ &= \mathbb{E}_{E_n} [P_{M,w}(x_n | \mathbf{a}_{1:m}, \mathbf{x}_{1:n-1})] \mathbb{E}_{E_1, \dots, E_{n-1}} [P_{M,w}(\mathbf{x}_{1:n-1} | \mathbf{a}_{1:m})] \\ &= P_M(x_n | \mathbf{a}_{1:m}, \mathbf{x}_{1:n-1}) \mathbb{E}_{E_1, \dots, E_{n-1}} [P_{M,w}(\mathbf{x}_{1:n-1} | \mathbf{a}_{1:m})], \end{aligned}$$

608 where the second last step uses the independence of the E_i values and the last step uses the unbi-
 609 asedness of the reweighting function. Repeating the same argument for the remaining E_i values, we
 610 obtain

$$\mathbb{E}_{E_1, \dots, E_n} [P_{M,w}(\mathbf{x}_{1:n} | \mathbf{a}_{1:m})] = P_M(\mathbf{x}_{1:n} | \mathbf{a}_{1:m}).$$

611

□

612 *Proof of Theorem 7.* Given a watermark code space \mathcal{E} and a watermark code distribution $P_E(e)$,
 613 we construct a key space $K = \mathcal{E}^C$, where each key k is a function from the context code space to
 614 the watermark code space. The random key probability density function is defined as $P_K(k) =$
 615 $\prod_{c \in C} P_E(k(c))$.

616 This construction forms a particular instance of an unbiased watermark code generation function. □

617 B.3 Detailed theory for Section 4.3

618 **Corollary 15.** *For every generation request by a user, Algorithm 1 can provide a generation result.*
 619 *This generation service is n -shot undetectability for any $n \in \mathbb{N}^+$ if the unbiased watermark code*
 620 *generation function is employed, and the context code history is continuously recorded. Specifically,*
 621 *the context code history cch is updated after each invocation of Algorithm 1, and the resulting context*
 622 *code history is used as the initial context code history for the next invocation.*

623 *On the other hand, if the context code history is reset after every generation task, the generation*
 624 *service can only guarantee 1-shot undetectability.*

625 *Proof.* The key design element in this service is the context code history. By maintaining the context
 626 code history throughout the generation process, we can ensure that each time the reweighting is
 627 performed, the context code is unique, i.e., it has not appeared in any previous generation tasks.
 628 According to the properties of the unbiased watermark code generation function in Definition 6, this
 629 guarantees that the watermark codes generated during each reweighting are independent of previously
 630 generated watermark codes. As a result, the final distribution is unbiased, and n -shot undetectability
 631 is achieved.

632 However, if the context code history is reset after every generation task, it is possible for two
 633 invocations of Algorithm 1 to produce the same context code, leading to the same watermark code.
 634 Consequently, n -shot undetectability cannot be guaranteed for $n > 1$, and the generation service can
 635 only provide 1-shot undetectability. □

636 A straightforward variant of the above approach exists in the form of a batch variant. If the batch
 637 size is set to b and the context code history is reset after each batch, the system can ensure b -shot
 638 undetectability.

639 B.4 Proof of tailed bounds in Section 5

Proof of Theorem 8.

$$\begin{aligned} \mathbb{E} \left[\exp \left(\sum_{i=1}^n s_i \right) \right] &= \mathbb{E} \left[\exp \left(\sum_{i=1}^{n-1} s_i \right) \mathbb{E}[\exp(s_n) | \mathcal{F}_{n-1}^x] \right] \\ &\leq \mathbb{E} \left[\exp \left(\sum_{i=1}^{n-1} s_i \right) \right] \leq \dots \leq 1, \end{aligned}$$

640 where the abbreviation in the last step means applying similar inequalities multiple times.

641 By applying the Chernoff bound, we obtain the desired result. □

642 *Proof of Theorem 9.* From Theorem 3, we know that

$$P \left(\sum_{i=1}^n s_i^{(a)} \geq t \right) \leq e^{-t}.$$

643 Thus,

$$P \left(\max_{1 \leq a \leq A} \left(\sum_{i=1}^n s_i^{(a)} \right) \geq t \right) \leq \sum_{1 \leq a \leq A} P \left(\sum_{i=1}^n s_i^{(a)} \geq t \right) \leq A e^{-t}.$$

644

□

645 B.5 Details on maximin variant of LLR score

646 B.5.1 Derivation of the solution

647 Recall that we are dealing with the maximin problem given as:

$$\begin{aligned} \max_{S_i} \quad & \min_{Q'_i \in \Delta_\Sigma, TV(Q'_i, Q_i) \leq d} \langle Q'_i, S_i \rangle \\ \text{s.t.} \quad & \langle P_i, \exp(S_i) \rangle \leq 1. \end{aligned}$$

648 We can find a relaxation by replacing the constraint $Q'_i \in \Delta_\Sigma$ with $\sum_{x \in \Sigma} Q'_i(x) = 1$ and no longer
649 requiring $Q'_i(x) \geq 0$. Thus, we obtain the following inequality:

$$\min_{Q'_i \in \Delta_\Sigma, TV(Q'_i, Q_i) \leq d} \langle Q'_i, S_i \rangle \geq \min_{Q'_i, \sum_{x \in \Sigma} Q'_i(x) = 1, TV(Q'_i, Q_i) \leq d} \langle Q'_i, S_i \rangle.$$

650 The new maximin problem becomes:

$$\begin{aligned} \max_{S_i} \quad & \min_{Q'_i, \sum_{x \in \Sigma} Q'_i(x) = 1, TV(Q'_i, Q_i) \leq d} \langle Q'_i, S_i \rangle \\ \text{s.t.} \quad & \langle P_i, \exp(S_i) \rangle \leq 1. \end{aligned}$$

651 This relaxation is tight, meaning it does not affect the final maximin optimal solution. This is because,
652 even though the relaxed problem does not require $Q'_i(x) \geq 0$, the maximin problem's optimal solution
653 S_i^* and Q'_i^* must satisfy $Q'_i^*(x) \geq 0$. Otherwise, $S_i^*(x)$ could be further reduced, implying that
654 $S_i^*(x)$ is not an optimal solution and leading to a contradiction.

655 The inner optimization of the relaxed problem can be solved directly:

$$\min_{Q'_i, \sum_{x \in \Sigma} Q'_i(x) = 1, TV(Q'_i, Q_i) \leq d} \langle Q'_i, S_i \rangle = \langle Q_i, S_i \rangle + d \left(\min_x S_i(x) - \max_x S_i(x) \right).$$

656 This leads to the new maximization optimization problem:

$$\begin{aligned} \max_{S_i} \quad & \langle Q_i, S_i \rangle + d \left(\min_x S_i(x) - \max_x S_i(x) \right) \\ \text{s.t.} \quad & \langle P_i, \exp(S_i) \rangle \leq 1. \end{aligned}$$

657 We can find the KKT conditions for this optimization problem by rewriting it as follows:

$$\begin{aligned} \max_{S_i} \quad & \langle Q_i, S_i \rangle + d(\max S_i - \min S_i) \\ \text{s.t.} \quad & \langle P_i, \exp(S_i) \rangle \leq 1, \\ & \max S_i \geq S_i(x), \\ & \min S_i \leq S_i(x). \end{aligned}$$

658 Let the Lagrangian be

$$\begin{aligned} L = \max_{S_i} & \langle Q_i, S_i \rangle + d(\min S_i - \max S_i) \\ & + \lambda(1 - \langle P_i, \exp(S_i) \rangle) \\ & + \langle u, \max S_i - S_i \rangle \\ & + \langle v, S_i - \min S_i \rangle. \end{aligned}$$

659 Then, the KKT conditions are:

$$\begin{aligned}\frac{\partial L}{\partial S_i(x)} &= [Q_i(x) - u(x) + v(x)] - \lambda P_i(x) \exp(S_i(x)) = 0, \\ \frac{\partial L}{\partial \max S_i} &= -d + \sum_{x \in \Sigma} u(x) = 0, \\ \frac{\partial L}{\partial \min S_i} &= d - \sum_{x \in \Sigma} v(x) = 0, \\ \lambda(1 - \langle P_i, \exp(S_i) \rangle) &= 0, \\ \langle u, \max S_i - S_i \rangle &= 0, \\ \langle v, S_i - \min S_i \rangle &= 0.\end{aligned}$$

660 We can solve for the value of λ :

$$\sum_{x \in \Sigma} \frac{\partial L}{\partial S_i(x)} = [1 - d + d] - \lambda \sum_{x \in \Sigma} P_i(x) \exp(S_i(x)) = 0.$$

661 Note that λ cannot be 0, so the fourth KKT condition implies $\langle P_i, \exp(S_i) \rangle = 1$. Consequently, the
662 above equation implies $\lambda = 1$.

663 The final solution is given by:

$$\begin{aligned}S_i(x) &= \log \frac{Q_i(x) - u(x) + v(x)}{P_i(x)}, \\ u(x) \neq 0 &\text{ iff } S_i(x) = \max_x S_i(x), \\ v(x) \neq 0 &\text{ iff } S_i(x) = \min_x S_i(x), \\ \sum_{x \in \Sigma} u(x) &= \sum_{x \in \Sigma} v(x) = d.\end{aligned}$$

664 B.5.2 Computing the solution

665 Let

$$\begin{aligned}X_{\max} &= \{x \in \Sigma \mid S_i(x) = \max_x S_i(x)\}, \\ X_{\min} &= \{x \in \Sigma \mid S_i(x) = \min_x S_i(x)\}.\end{aligned}$$

666 If $x \notin X_{\max} \cup X_{\min}$, then we have

$$S_i(x) = \log \frac{Q_i(x)}{P_i(x)}.$$

667 If $x \in X_{\max}$, then we have

$$\max_x S_i(x) = S_i(x) = \log \frac{Q_i(x) - u(x) + v(x)}{P_i(x)}.$$

668 Summing over all $x \in X_{\max}$, and noting that $\sum_{x \in X_{\max}} u(x) = d$, we obtain:

$$\max_x S_i(x) = \log \frac{\sum_{x \in X_{\max}} Q_i(x) - d + \sum_{x \in X_{\max}} v(x)}{\sum_{x \in X_{\max}} P_i(x)}.$$

669 Similarly,

$$\min_x S_i(x) = \log \frac{\sum_{x \in X_{\min}} Q_i(x) - \sum_{x \in X_{\min}} u(x) + d}{\sum_{x \in X_{\min}} P_i(x)}.$$

670 When $\sum_{x \in X_{\min}} u(x) \neq 0$, it implies that there exists an $x \in X_{\min}$ such that $x \in X_{\max}$, which in
671 turn implies that $\max_x S_i(x) = S_i(x) = \min_x S_i(x)$. In this case, the score is trivial, with $S_i(x) = 0$
672 for all $x \in \Sigma$.

673 Thus, the computation of the maximin solution reduces to finding X_{\max} and X_{\min} , which can be
674 computed in $\tilde{O}(|\Sigma|)$ time. A pseudocode is shown in Algorithm 2.

675 Note that the provided pseudocode is not a real implementation but serves as a schematic representa-
676 tion of the algorithm. In our experimental implementation, we took into consideration the effective
677 precision of computer floating-point numbers. To ensure numerical stability and prevent NaNs, we
678 implemented the algorithm in log space. This makes the algorithm more complex, and additionally,
679 we designed the algorithm with grid search by reusing previous computation results for acceleration.
680 We also implemented such algorithm with tensor operator for efficient computation on GPU. For
681 more details, please refer to the source code.

682 **Algorithm 2** Computation of maximin variant of LLR score

```
import numpy as np

def get_max_lr(P: np.ndarray, Q: np.ndarray, d: float) -> float:
    """Get  $\max_x \exp(S(x))$ """
    indexes = sorted(range(len(P)), key=lambda i: Q[i] / P[i], reverse=True)

    sum_Q = 0.0
    sum_P = 0.0

    def _lr():
        nonlocal sum_Q, sum_P
        if sum_Q <= d:
            return 0.0
        else:
            return (sum_Q - d) / sum_P

    lr = _lr()

    for i in indexes:
        if Q[i] / P[i] < lr:
            break
        sum_Q += Q[i]
        sum_P += P[i]
        lr = _lr()
    return lr

def get_min_lr(P: np.ndarray, Q: np.ndarray, d: float) -> float:
    """Get  $\min_x \exp(S(x))$ """
    indexes = sorted(range(len(P)), key=lambda i: Q[i] / P[i])

    sum_Q = 0.0
    sum_P = 0.0

    def _lr():
        nonlocal sum_Q, sum_P
        return (sum_Q + d) / sum_P

    lr = _lr()

    for i in indexes:
        if Q[i] / P[i] > lr:
            break
        sum_Q += Q[i]
        sum_P += P[i]
        lr = _lr()
    return lr
```

```

    return lr

def get_S(P: np.ndarray, Q: np.ndarray, d: float) -> np.ndarray:
    max_lr = get_max_lr(P, Q, d)
    min_lr = get_min_lr(P, Q, d)
    lr = Q / P
    if max_lr <= min_lr:
        return np.zeros_like(p)
    return np.log(np.clip(lr, min_lr, max_lr))

```

683 C Additional discussion

684 **Performance without context code history** Despite that “context code history” is necessary to
685 ensure n -shot-undetectable, it’s possible to bypass this requirement, and always execute steps 9 and
686 10 in Algorithm 1. In many instances, this won’t significantly degrade the performance of downstream
687 tasks, as the probability of context code collision is low. However, if one chooses to neglect the
688 context code history, they effectively waive the theoretical guarantee of n -shot-undetectability and
689 potentially expose themselves to corner cases that could notably undermine the task performance.
690 Moreover, users could specifically construct test cases that check for the existence of watermarks.
691 For instance, prompts like “Generate a random bit (0 or 1):” or “Generate a random bit sequence,
692 with five dots between every two digits:” would yield incorrect results in the absence of context code
693 history.

694 **Computation of logits during detection** The watermark detection methods in Sections 5.2 and 5.3
695 relies on the output probability distribution P_M . Ideally, the P_M used during detection should be
696 the same as the one during generation. However, this may not always be possible. Language model
697 logits depend on various parameters like the prompt, the temperature and sampling policy used
698 during generation, etc., which might not be accessible during watermark detection. For instance, P_M
699 depends on the prompt, but during detection, we might only have the text to be examined and not the
700 prompt from which it was generated.

701 In such circumstances, we can only resort to using another distribution P'_M as an estimation of P_M .
702 For instance, if the prompt is missing during detection, we can set the prompt to an empty string and
703 then calculate the language model probabilities. In a machine translation task, one could translate the
704 output back to the input language and use that as input. In practice, there’s likely to be a disparity
705 between P'_M and P_M , which could lead to a drop in score. We discuss in detail how the score is
706 affected by changes in logits in Appendix F.2.

707 **Cost of likelihood computation** The detection methods in Sections 5.2 and 5.3 require the output
708 probability distribution P_M . This comes at a computational cost: it’s more computationally expensive
709 than red list-based methods proposed by Kirchenbauer et al. [32], as it involves a language model.
710 However, the cost is much less than a generation, as it only requires a single forward pass.

711 On the other hand, our framework also supports likelihood-agnostic detection methods, which have
712 their own pros and cons. We present a detailed comparison of likelihood-based and likelihood-
713 agnostic methods and provide an example in Appendix D.

714 **Perturbation of P** The method in Section 5.3 introduces a variation of the log likelihood ratio
715 test where the watermarked distribution $P_{M,w}$ is perturbed, resulting in a new optimization problem.
716 Similarly, we could introduce a perturbation to the original distribution P_M . Specifically, we would
717 adjust the original constraint of $\langle P_i, \exp(S_i) \rangle \leq 1$ to be $\langle P'_i, \exp(S_i) \rangle \leq 1, \forall P'_i$, s.t. $TV(P_i, P'_i) \leq$
718 d' , where $TV(P_i, P'_i)$ denotes the total variation distance between P_i and P'_i and d' is a small positive
719 number.

720 This new optimization problem can be solved using similar methods as those in Appendix B.5.2. We
721 have implemented this computation in our codebase. However, for the experiments in this paper, we
722 only used the case where $d' = 0$.

723 **D Likelihood-agnostic watermark score**

724 Our unbiased watermark can also be detected in a likelihood-agnostic way such that it does not rely
 725 on a language model and its output logits to compute the score.

726 **D.1 Method**

727 **D.1.1 Reweighting function**

728 We use the same δ -reweighting as in Section 4.1.2, but with a different implementation. Instead
 729 of using inverse sampling, we can also use Gumbel trick. Specifically, each watermark code is
 730 a list of $|\Sigma|$ number of independent and identically distributed standard Gumbel variables. The
 731 watermark code space is $\mathcal{E} = \mathbb{R}^\Sigma$. The probability density function of the watermark code is given by
 732 $P_E(E) = \prod_{a \in \Sigma} e^{-E(a) + e^{E(a)}}$.

733 To sample a token using the Gumbel trick, we compute $a^* = \operatorname{argmax}_a \{\log P(a) + E(a)\}$, and the
 734 reweighted distribution becomes $Q = \delta_{a^*}$. Gumbel variables allow us to guess the likelihood of a
 735 token coming from the watermark model without relying on logits, as tokens with larger Gumbel
 736 variables are more likely to be picked by the watermark model.

737 **D.1.2 Score design and tail bound**

738 Similar to Section 5, we calculate scores for each token, but without relying on the original and
 739 reweighted distribution P and Q . Thus, the design of the likelihood-agnostic score has a certain
 740 degree of arbitrariness, unlike the method in Sections 5.2 and 5.3 which was derived in a principled
 741 way.

742 We choose the score to be $s_i = \ln 2 - \exp(-E(a^*))$. One of the main concerns of this construction
 743 is that it can yield a tail bound similar to Theorem 8.

744 **Theorem 16.** *For n independent random variables $G_i \sim \text{Gumbel}(0, 1)$, if we define $s_i = \ln 2 -$
 745 $\exp(-G_i)$, we have $\mathbb{E}[\exp(s_i)] \leq 1$ and $P(\sum_{i=1}^n s_i \leq t) \leq e^{-t}$.*

746 For a token with watermark, the average score is $\mathbb{E}[\ln 2 - \exp(-G_i(a^*))] = \ln 2 - \sum_{a \in \Sigma} P(a)^2 =$
 747 $\ln 2 - \exp(-H_2(P))$, where $H_2(P)$ is the Rényi entropy of order 2. Therefore, the average score is
 748 positive only when the entropy is high.

749 Note that Theorem 16 requires independence of s_i , unlike Theorem 8 where the s_i can be a random
 750 process. In practice, the Gumbel variables depend on the watermark code, and the watermark code
 751 might repeat, leading to dependencies between Gumbel variables and thus between scores. To address
 752 this issue, for repeating context codes, we set the score to zero, ensuring that Theorem 16 remains
 753 applicable.

The detection process is as follows: given a text $\mathbf{x}_{1:n} = (x_1, \dots, x_n)$, we obtain a series of context
 codes (cc_1, \dots, cc_n) and watermark codes (E_1, \dots, E_n) . The final scores are computed as

$$s_i = \begin{cases} \ln 2 - \exp(-E_i(x_i)) & \text{if } cc_i \notin cc_1, \dots, cc_{i-1}, \\ 0 & \text{if } cc_i \in cc_1, \dots, cc_{i-1}. \end{cases}$$

754 **D.2 Comparison between likelihood-based score and likelihood-agnostic score**

755 Compared to the likelihood-based score, the likelihood-agnostic score has some notable drawbacks.

756 As it does not rely on logits, it cannot distinguish between high and low entropy situations. In low
 757 entropy cases, the likelihood-agnostic score still tends to have a large absolute value, even though it
 758 does not provide any signal and only contributes noise, lowering the score. In extreme cases, when
 759 the entropy is zero, the generation result is deterministic, and the ideal detection algorithm should
 760 output a zero score, as there is no evidence for or against the presence of the watermark. However,
 761 the likelihood-agnostic score would output a negative average score, giving a false indication that the
 762 text was not generated by a model with watermark.

763 Moreover, in cases where the original distribution P_M is known, the likelihood-agnostic score is
 764 much smaller than the log likelihood ratio based score. According to the Neyman-Pearson lemma,

765 the log likelihood ratio test is the most powerful statistical test, and its maximin variant also retains
766 this property to a certain degree, thus providing a higher score than likelihood-agnostic score.
767 On the other hand, the likelihood-agnostic score has a lower computational cost, as it does not depend
768 on the logits computed by a large language model. Furthermore, the fact that likelihood-agnostic
769 score is independent of logits from the language model makes it more appealing when the original
770 distribution P_M is hard to estimate during detection.

771 E Detailed experiment setup

772 We evaluate the performance of our Unbiased Watermarks on two important applications of seq2seq
773 models: text summarization(TS) and machine translation(MT).

774 **Text summarization.** In the TS task, we adopt the test set of of CNN-DM [25] corpus, which consists
775 of 11,490 examples. The model applied is BART-large, which contains 400 million parameters.

776 **Machine translation.** For the MT task, we employ the WMT’14 English (En) to Romanian (Ro)
777 dataset, which has a test set size of 1,999 examples. The Multilingual Bart (MBart) [37] model and
778 its official tokenizer is applied.

779 **Watermark setup.** We evaluate two reweighting functions in our experiment: δ -reweight and
780 γ -reweight. For context code generation, we employ the most recent five tokens as context code.
781 For example, if the current input to the decoder is (x_1, x_2, x_3) , the context code used in generating
782 x_4 would be (x_1, x_2, x_3) , considering only three tokens are available. Context code history is reset
783 before generating each batch, thereby making our method b -shot-undetectable given a batch size of b .
784 For the unbiased watermark code generation function, we use SHA-256 as the hash function and a
785 1024-bit random bitstring as the key k . The watermark code E is sampled from P_E using $\text{hash}(c, k)$
786 as the random seed.

787 In addition, we compared our method with the soft-red-list watermarking method from Kirchenbauer
788 et al. [32]. Their method depends on two parameters δ , controlling the size of the change in logits,
789 and γ , which is the proportion of the green list in the total vocabulary. We test δ with three values:
790 0.0, 1.0, 2.0, and fix γ to be $\frac{1}{2}$. It is important to clarify that the δ and γ in our δ -reweight and
791 γ -reweight are different from those in Kirchenbauer et al.’s method. In the latter, δ and γ are
792 hyperparameters, while in our method, δ -reweight and γ -reweight are names of two reweighting
793 strategies.

794 **Watermark detection.** We employ the maximin variant of LLR score for watermark detection. The
795 score depends on a perturbation strength d and is optimized by performing a grid search over the set
796 $\{0, 0.1, \dots, 0.9, 1.0\}$, which consists of 11 points. The optimal perturbation strength is the one that
797 yields the highest score sum.

798 **Evaluation metrics.** For the TS task, we employ the ROUGE score [35], which measures the overlap
799 in terms of n-grams to assess the effectiveness of the summary in capturing the essential content from
800 the reference summaries. For the MT task, we use the BLEU score [42] that emphasizes the lexical
801 similarity between the machine-generated translations and the human reference translations. We
802 estimated the distribution and standard error of BLEU score based on bootstrapping. In both tasks,
803 we also apply BERTScore and Perplexity as auxiliary metrics.

804 **Computational costs.** Our experiments are carried out on a machine equipped with 2x AMD EPYC
805 7513 32-Core Processor and 8x A6000 GPUs. All experiments can be completed within 4 hours.

806 **Implementation.** The experiments are implemented based on the Huggingface library [64], a popular
807 platform for developing and sharing models in the NLP community.

808 F More experiment

809 F.1 Adding watermark

810 Tables 4 and 5 shows more result under the same setup as Table 1.

Table 4: Additional result about the performance of different watermarking methods on TS. We scale BERTScore and ROUGE with a factor of 100.

	BERTScore.Precision \uparrow	BERTScore.Recall \uparrow	ROUGE-2 \uparrow	ROUGE-L \uparrow
No Watermark	0.3180 \pm 0.0009	0.3361 \pm 0.0010	0.1388 \pm 0.0008	0.2445 \pm 0.0008
δ -reweight	0.3180 \pm 0.0009	0.3365 \pm 0.0010	0.1392 \pm 0.0008	0.2451 \pm 0.0008
γ -reweight	0.3180 \pm 0.0009	0.3360 \pm 0.0010	0.1397 \pm 0.0008	0.2451 \pm 0.0008
Soft($\delta=0.0$)	0.3180 \pm 0.0009	0.3361 \pm 0.0010	0.1388 \pm 0.0008	0.2445 \pm 0.0008
Soft($\delta=1.0$)	0.3092 \pm 0.0009	0.3382 \pm 0.0009	0.1344 \pm 0.0007	0.2400 \pm 0.0007
Soft($\delta=2.0$)	0.2908 \pm 0.0008	0.3339 \pm 0.0009	0.1238 \pm 0.0007	0.2293 \pm 0.0007

Table 5: Additional result about the performance of different watermarking methods on MT. We scale BERTScore with a factor of 100.

	BERTScore.Precision \uparrow	BERTScore.Recall \uparrow	Perplexity \downarrow
No Watermark	0.546 \pm 0.003	0.575 \pm 0.003	2.31 \pm 0.07
δ -reweight	0.550 \pm 0.003	0.579 \pm 0.003	2.20 \pm 0.05
γ -reweight	0.549 \pm 0.003	0.577 \pm 0.003	2.24 \pm 0.04
Soft($\delta=0.0$)	0.546 \pm 0.003	0.575 \pm 0.003	2.31 \pm 0.07
Soft($\delta=1.0$)	0.537 \pm 0.003	0.568 \pm 0.003	2.43 \pm 0.07
Soft($\delta=2.0$)	0.523 \pm 0.003	0.555 \pm 0.003	2.81 \pm 0.07

811 **F.2 Sensitivity of scores**

812 The detection methods in Sections 5.2 and 5.3 rely on the output logits of the language models,
 813 which in turn depend on various factors such as the prompt, the temperature and sampling policy
 814 used during the generation process, and the language model itself. In this section, we measure the
 815 sensitivity of the scores to changes in these parameters.

816 Watermarked samples are generated from the distribution $P_{M,w}$, which comes from reweighting of
 817 the original distribution P_M . However, during detection, we modify some parameters, including
 818 temperature, sampling policy (top_k), input, and model, resulting in a new probability distribution
 819 P'_M .

820 The following table demonstrates the decrease in scores under different changes, showing that when
 821 P'_M is not equal to P_M , the scores decline. This implies that more tokens are required to accumulate
 822 sufficient evidence to prove the existence of the watermark.

Table 6: Score per token when the estimated token distribution is computed from a different temperature than the real token distribution.

temperature	Text summarization		Machine translation	
	δ -reweight	γ -reweight	δ -reweight	γ -reweight
0.5	0.049 \pm 0.407	0.133 \pm 0.309	0.041 \pm 0.303	0.084 \pm 0.241
1.0 (groundtruth)	0.878 \pm 1.435	0.220 \pm 0.367	0.420 \pm 1.135	0.105 \pm 0.291
1.5	0.036 \pm 0.498	0.166 \pm 0.455	0.019 \pm 0.324	0.088 \pm 0.335

Table 7: Score per token when the estimated token distribution is computed from a different top_k than the real token distribution.

top_k	Text summarization		Machine translation	
	δ -reweight	γ -reweight	δ -reweight	γ -reweight
20	0.520 \pm 1.144	0.212 \pm 0.362	0.274 \pm 0.859	0.101 \pm 0.284
50 (groundtruth)	0.878 \pm 1.435	0.220 \pm 0.367	0.420 \pm 1.135	0.105 \pm 0.291
100	0.582 \pm 1.262	0.219 \pm 0.369	0.288 \pm 0.930	0.105 \pm 0.292
No top_k sampling	0.377 \pm 1.124	0.216 \pm 0.373	0.022 \pm 0.349	0.097 \pm 0.324

823 Comparing the two reweight functions, we find that when P'_M is equal to P_M , the δ -reweight always
 824 yields a higher score than the γ -reweight. However, when P'_M is different from P_M , the scores
 825 obtained from the δ -reweight exhibit a significant drop, whereas the decline in scores for the γ -

Table 8: Score per token when the estimated token distribution is computed with and without input.

	Text summarization		Machine translation	
	δ -reweight	γ -reweight	δ -reweight	γ -reweight
with input (groundtruth)	0.8783 ± 1.4353	0.2206 ± 0.3677	0.4201 ± 1.1355	0.1058 ± 0.2916
without input	0.0108 ± 0.2170	0.0244 ± 0.2417	0.0096 ± 0.2004	0.0186 ± 0.1904

Table 9: Score per token when the estimated token distribution is computed from a different model than the real token distribution.

model	Text summarization	
	δ -reweight	γ -reweight
"philschmid/bart-large-cnn-samsum" (groundtruth)	0.878 ± 1.435	0.220 ± 0.367
"facebook/bart-large-cnn"	0.041 ± 0.447	0.091 ± 0.412

826 reweight is always more gradual than that of the δ -reweight. This indicates that the γ -reweight is less
 827 sensitive to the differences between P'_M and P_M .

828 F.3 Likelihood-agnostic score

829 When applied to text summarization, which is a task with relatively high entropy, the likelihood-
 830 agnostic score is positive on average but an order of magnitude lower than the likelihood-based score.
 831 For machine translation, which is a low entropy task, the average score is negative, and thus cannot
 832 be used to detect watermark in this case.

Table 10: Mean and variance of score per token for δ -reweight based on Gumbel trick on different tasks.

	Text summarization	Machine translation
Maximin variant of LLR score	0.876 ± 1.444	0.429 ± 1.172
Likelihood-agnostic score	0.078 ± 0.776	-0.104 ± 0.891

833 G Limitations

834 G.1 Major Limitations

- 835 • First, we note that our unbiased watermarking technique only works for generative processes with
 836 high entropy. In an extreme case, when entropy is 0 and output of the original model is fixed, any
 837 unbiased watermarking method will always yield the same result as the original model. As a result,
 838 it is challenging to integrate our unbiased watermarking approach with beam search algorithms
 839 due to their intrinsic deterministic nature.
- 840 • Second, our study does not address the attacks to the watermark. Numerous ways of watermark
 841 removal have been explored, ranging from simple text insertion to more sophisticated methods like
 842 paraphrasing attacks. While these topics are beyond the scope of this paper, they are nonetheless
 843 crucial to consider for a comprehensive understanding of the watermarking problem.

844 G.2 Minor Limitations

- 845 • Even though we have proposed a watermarking framework, there is considerable design space left
 846 unexplored. Many reweighting functions and context codes may be applicable, but it is unclear
 847 which one is optimal in practice, particularly since we currently lack standard evaluation metrics.
 848 We expect that continued research in this area could possibly shed more light on this subject.
- 849 • In Algorithm 1, the introduction of context code history strictly ensures n-shot-undetectable
 850 watermarking at the expense of additional storage requirements, as the context code history from
 851 past generation processes needs to be retained. This presents a trade-off between storage and
 852 undetectability. For instance, if we store all context codes in the previous n generated outputs,
 853 we can ensure n -shot-undetectability. However, the greater the value of n , the larger the required
 854 storage space, though this does provide stronger undetectability. Generally, storage complexity
 855 increases with $O(n)$.

856 **H Broader impacts**

857 Our unbiased watermark has removed major hurdles for large-scale application of watermarks. The
858 two primary obstacles previously were the potential for watermarks to degrade the quality of output
859 and the possibility for users to discern the presence of watermarks. Our method addresses both of
860 these issues thoroughly.

861 **H.1 Impact analysis**

862 **Traceability and accountability** Traceability refers to the ability to trace back the origin of a text.
863 Any watermarking method, including method in this paper, contribute to traceability. In an era of
864 misinformation and disinformation, this allows for holding providers accountable for the content
865 generated by their models.

866 **Identifying model-generated texts** Watermarking method can be used to distinguish which texts
867 are generated by the models. This prevents unnecessary training on the data generated by the models
868 themselves.

869 **Ownership** Watermarking method can help provide evidence in situations where a provider claims
870 ownership over a generated text [50].

871 **Data privacy concerns** The use of different watermarks, if applied discretionarily, could potentially
872 link generated text back to a specific user or request. This could be seen as a breach of users' privacy,
873 raising important data privacy concerns.

874 **Manipulation and removal of watermarks** The ongoing development of techniques to manipulate
875 or remove watermarks could lead to an "arms race" between providers attempting to secure their
876 watermarks and users trying to remove them.

877 **H.2 Ethical considerations**

878 There are several ethical considerations in the pursuit of watermarking technology.

879 **Consent** Users have the right to be informed about the use of watermarks and should have the
880 option to opt-out.

881 **Transparency** Providers should be transparent about the use of watermarks, including information
882 on what is embedded within these watermarks and how it's used. If the watermarks contain identifying
883 information, providers should clearly state who can access this information and take appropriate
884 measures to prevent potential misuse.

885 **Fair use** The application of our watermarking technique should not interfere with the legitimate
886 use of the service by users.

887 Our watermarking method does not degrade the quality of the output, ensuring the values of fair use
888 are upheld. However, it also introduces a potentially challenging issue.

889 Due to the undetectable nature of our technique, every user might have to assume that the service
890 they are using has been watermarked, as it cannot be disproved. This raises challenging questions on
891 how to ensure consent and transparency.

892 **H.3 Conclusion**

893 Our unbiased watermarking method brings improved traceability and attribution and ensures that fair
894 use is not compromised. However, it also poses significant challenges in data privacy, transparency,
895 and consent. Any implementation of this system needs to be done thoughtfully and ethically, with
896 clear communication to users about how it works and what it means for them.