AI-generated Text Detection via Watermarking

Neil Gong

Text watermarks

- Non-learning-based
- Learning-based

Watermark Generation

Algorithm 1 Text Generation with Hard Red List

Input: prompt, $s^{(-N_p)} \dots s^{(-1)}$

for $t = 0, 1, \cdots$ **do**

1. Apply the language model to prior tokens $s^{(-N_p)} \dots s^{(t-1)}$ to get a probability vector $p^{(t)}$ over the vocabulary.

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

3. Using this seed, randomly partition the vocabulary into a "green list" G and a "red list" R of equal size.

4. Sample $s^{(t)}$ from G , never generating any token in the red list.

end for

Slides credit: Weihang Li

Watermark Detection

Basis: We can detect the watermark by testing the following null hypothesis:

H_0: *The text sequence is generated with no knowledge of the red list rule.*

Naive approach:
The probability that a natural source produces T tokens without violating the red list rule is only $1/2^T$, which is vanishingly small.

Watermark Detection

Alternative approach: *one proportion z-test*
If the null hypothesis is true, then the number of green list tokens, denoted $|s|_G$, has expected value $T/2$ and variance $T/4$. The z-statistic for this test is:

$$z = 2(|s|_G - T/2)/\sqrt{T}$$

We **reject the null hypothesis** and detect the watermark if z is above a chosen threshold.

Example:
Suppose we choose to reject the null hypothesis if $z > 4$
- The probability of a false positive is 3×10^{-5}
- We will detect any watermarked sequence with 16 or more tokens (the minimum value of T that produces $z = 4$ when $|s|_G = T$)

Watermark Generation - Soft Watermark

Algorithm 2 Text Generation with Soft Red List

Input: prompt, $s^{(-N_p)} \cdots s^{(-1)}$
green list size, $\gamma \in (0, 1)$
hardness parameter, $\delta > 0$

for $t = 0, 1, \cdots$ **do**

1. Apply the language model to prior tokens $s^{(-N_p)} \cdots s^{(t-1)}$ to get a logit vector $l^{(t)}$ over the vocabulary.

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

3. Using this random number generator, randomly partition the vocabulary into a "green list" G of size $\gamma|V|$, and a "red list" R of size $(1 - \gamma)|V|$.

4. Add δ to each green list logit. Apply the softmax operator to these modified logits to get a probability distribution over the vocabulary.

$$\hat{p}_k^{(t)} = \begin{cases} \frac{\exp(l_k^{(t)}+\delta)}{\sum_{i\in R}\exp(l_i^{(t)})+\sum_{i\in G}\exp(l_i^{(t)}+\delta)}, & k \in G \\ \frac{\exp(l_k^{(t)})}{\sum_{i\in R}\exp(l_i^{(t)})+\sum_{i\in G}\exp(l_i^{(t)}+\delta)}, & k \in R. \end{cases}$$

5. Sample the next token, $s^{(t)}$, using the watermarked distribution $\hat{p}^{(t)}$.

end for

Watermark Detection - Soft Watermark

(Identical to that for the hard watermark) We reject the null hypothesis and detect the watermark if z is greater than a threshold. For arbitrary γ we have:

$$z = (|s|_G - \gamma T)/\sqrt{T\gamma(1-\gamma)}$$

Example: z > 4, we get false positives with rate $3 * 10^{-5}$

Scott Aaronson's text watermark
https://www.youtube.com/watch?v=2Kx9jbSMZqA

Given: Tokens $w_1, \ldots, w_{t-1}$, and a probability distribution $D_t = (p_{t,1}, \ldots, p_{t,K})$ over t^{th} token w_t

Also: Pseudorandom function $f(w_{t-c+1}, \ldots, w_{t-1}, i)$, which maps the latest c tokens to (say) $r_{t,i} \in [0,1]$

Goal: Choose a t^{th} token i that looks like it's drawn from D, but also secretly boosts $r_{t,i}$

In detection phase: We have access to a document $w_1, \ldots, w_n$, and hence the $r_{t,i}$'s, but **not** the $p_{t,i}$'s

The Gumbel Softmax Scheme

At each position t, choose the token
$i = i(t)$ that maximizes $r_{t,i}^{1/p_{t,i}}$

Intuition: The smaller is $p_{t,i}$, the larger the exponent, which means the closer $r_{t,i}$ must be to 1 for i to have a chance of being chosen

In detection phase: Calculate $\sum_{t=1}^{n} \ln \frac{1}{1-r_{t,i(t)}}$.

Iff this sum exceeds a threshold, say that GPT probably wrote the thing.

Google’s SynthID- Scalable watermarking for identifying large language model outputs

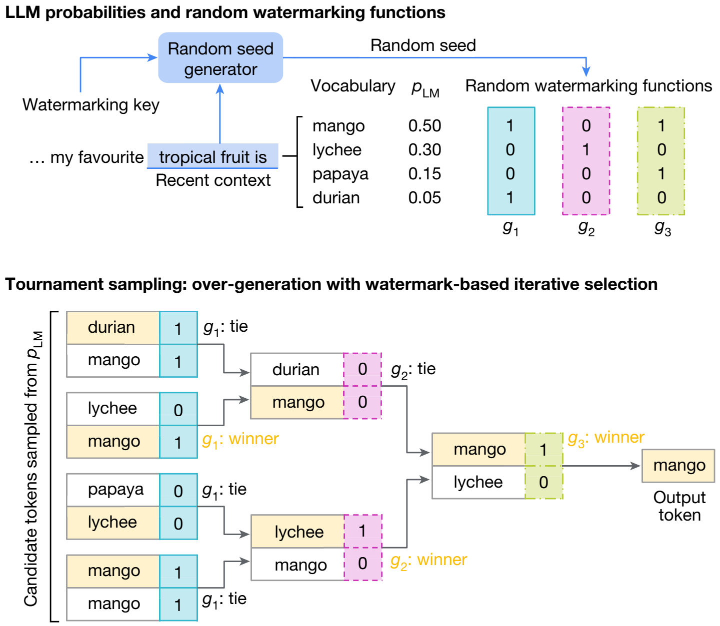

Fig. 2 | SynthID-Text’s Tournament-based watermarking. Top: to generate a new token x_t, we first score each token in the vocabulary using multiple (in this case, $m = 3$) random watermarking functions $g_1, \ldots, g_m$. These assign random values using a random seed, which is generated based on both the recent context and a watermarking key. Bottom: then, we choose the next token using a tournament process. First, we sample $2^m = 8$ (possibly non-unique) tokens from $p_{LM}(\cdot|x_{<t})$. These are split into pairs of competing tokens; in each pair, the highest scoring one (based on g_1) is chosen, breaking ties randomly. The resulting tokens compete in the next layer, where winners are chosen based on g_2, until in the last tournament layer the final winner is selected based on g_m: this becomes the next generated token x_t.

$$\text{Score}(x) = \frac{1}{mT} \sum_{t=1}^{T} \sum_{\ell=1}^{m} g_\ell(x_t, r_t).$$

Learning-based watermark - Adversarial Watermarking Transformer: Towards Tracing Text Provenance with Data Hiding