Simple Probability Truncation Improves Soft Red List Watermarks

Henrique Da Silva Gameiro [1 2] **Andrei Kucharavy** [3]

Abstract

Watermarking, whereby LLM outputs are steered to encode an easily identifiable digital signature, has recently gained attention as a potential solution for detecting synthetically generated text. However, watermarking schemes require tradeoffs between detectability (i.e., how easily the watermark can be identified by an algorithm) and quality of the generated text (i.e., the stylistic and semantic disruption to the normal generation of the LLM). In this work, we propose a simple extension to the Soft Red List watermark, *Softer Red List*, which enables higher detectability while maintaining text quality on par with non-watermarked text. Specifically, Softer Red List improves the classical red/green token algorithm by adding a probability truncation filter before boosting the probabilities tokens in the green list. Despite its simplicity, Softer Red List matches or exceeds the performance of previously published LLM watermarking schemes, notably achieving a better detection rate at a low false positive rate (FPR) than SynthID in the disinformation detection setting, all while maintaining comparable perplexity and better reasoning capacities. We provide the watermark implementation for the `Transformers` library here: `https://github.com/Reliable-Information-Lab-HEVS/softer-red-list-llm-watermark`.

1. Introduction

AI-generated content is becoming increasingly prevalent online. As examples, 5% of newly created content in Wikipedia is suspected (through algorithmic detection) of being AI-generated (Brooks et al., 2024), and at least 15.8 % of the reviews at the most recent ICLR 2024 conference were also detected to be AI written (Latona et al., 2024). As humans struggle to differentiate between authentic human and LLM-generated content (Ippolito et al., 2020), algorithmic detectors of synthetic text provide a potential solution for identifying LLM-generated text (Solaiman et al., 2019; Ippolito et al., 2020; Zellers et al., 2019). However, such detectors were rapidly shown to suffer from a variety of robustness issues, such as generalizing to outputs of new or fine-tuned models (Adelani et al., 2020), complex prompts (Bakhtin et al., 2019), modifications to the sampling strategy (Ippolito et al., 2020; Stiff & Johansson, 2022), and single character/word rephrasings (Gagiano et al., 2021). These challenges still plague modern detectors (Dugan et al., 2024; Gameiro et al., 2024; Wang et al., 2024) and suggest that LLM-generated text may just not be sufficiently different from human-generated text to allow for a consistent and reliable detection.

Watermarking LLM outputs addresses these challenges by injecting a more easily detectable *digital signature* into the texts generated by LLMs. For example, the *Soft Red List* (Kirchenbauer et al., 2024a) watermarking algorithm amplifies the generation of a fixed set of tokens by the LLM, which achieves a near-perfect detectability (when the watermark is known and the sequence is long enough for the watermark to be detected), and is robust against the vast majority of adversarial attacks (with the exception of independent LLM paraphrasing) (Kirchenbauer et al., 2024b; Krishna et al., 2023). However, this high degree of detectability introduces a *quality-detectability* tradeoff as the generated text can become unnatural when the watermark amplifies unlikely tokens (Pan et al., 2024; Lee et al., 2024). Most subsequent proposed watermarks establish better tradeoffs (e.g., SynthID achieves both better detectability and naturalness as measured by perplexity; Dathathri et al., 2024b).

However, the quality-detectability tradeoff is difficult to manage in real-world settings. First, for some of the most concerning misuse scenarios, such as viral social media disinformation posts (Goldstein et al., 2023), the length of the watermarked text does not reach the thresholds tested for

[*]Equal contribution [1]Lakera [2](work done while at) Section of Computer Science, EPFL, Lausanne, Switzerland [3]Institute of Informatics, HES-SO Valais-Wallis, Sierre, Switzerland. Correspondence to: Andrei Kucharavy <first.last@hevs.ch>.

Proceedings of the Swiss AI Days 2026, Martigny, VS & Fribourg, FR, Switzerland PMLR 309, 2026.

detecting many LLM watermarks. Effectively watermarking these outputs would require a more aggressive watermarking strategy while still maintaining acceptable text quality for users. Second, even in responsible use scenarios, modern LLMs are increasingly used to perform tasks such as code generation or reasoning, requiring watermarks to not only be linguistically acceptable, but also preserve more complex semantics (Lee et al., 2024).

Motivated by these challenges, we introduce *Softer Red List*,[1] a modification to the Soft Red List algorithm that improves the Pareto frontier of the detectability-quality tradeoff. Specifically, *Softer Red List* only amplifies watermark tokens that are below a chosen threshold p of the cumulative probability distribution, enabling fine-grained control of the quality-detectability tradeoff by setting the p-threshold based on the user context.

We evaluate *Softer Red List* on multiple benchmarks. First, we extend the dynamic social media disinformation benchmark introduced by Gameiro et al. (2024) that best approximates real-world disinformation detection scenarios. We show that in this short-text benchmark, Softer Red List is the only watermark allowing over 95% TPR @ 1% FPR at an undetectable ($\sim 1\%$) PPL increase. Second, to simulate the application of watermarks in many real-world settings, we also evaluate our watermark in the context of additional generation quality metrics, Human-eval+ (Liu et al., 2023) for code generation and GSM8K (Cobbe et al., 2021) for mathematical reasoning. Our results on these benchmarks show a more nuanced picture of the quality-detectability tradeoff, with *Softer Red List* matching or exceeding the performance of most state-of-the-art watermarking algorithms. Finally, we show that *Softer Red List* is also the watermark most resilient to the paraphrasing attack, although not yet matching SotA LLM detectors.

2. Background and Related Work

2.1. LLM Detectors

Detecting synthetic text produced by LLMs was traditionally framed as a classification problem (Solaiman et al., 2019; Zellers et al., 2019), where detectors would trained to distinguish between human-written and machine-generated text. However, subsequent studies have shown that such detectors often fail to generalize across models, domains, or prompts (Dugan et al., 2024; Gameiro et al., 2024; Wang et al., 2024), limiting their utility in real-world applications. To address the robustness limitations of trained detectors, zero-shot detectors (Mitchell et al., 2023; Bao et al., 2024) are not directly trained on dedicated detection datasets, but instead rely on differences in the statistical signatures of human and model-generated text. Proprietary systems like GPTZero also demonstrate strong performance (Tian & Cui, 2024), though they are vulnerable to both false positives—particularly with texts written by non-native speakers—and adversarial inputs that trivially bypass detection (Dugan et al., 2024; Gameiro et al., 2024).

[1]We openly release our implementation of *Softer Red List* watermarking here: https://github.com/Reliable-Information-Lab-HEVS/softer-red-list-llm-watermarke.

2.2. LLM watermarking

While Zero-shot detectors have shown encouraging results, they rely on a distribution shift between human-written text and LLM-generated text due to the LLM objective and decoding used. However, such a shift might not be present for future LLMs, and looking for one can lead to false positives, for instance, for non-English speakers and neurodivergent people (Liang et al., 2023). Digital watermarking, an already widely used technique for marking online content such as images and videos, doesn't suffer from these issues.

Kirchenbauer et al. Green/Red tokens Kirchenbauer et al. (2024a)'s green/red token algorithm (Soft Red List algorithm) is the first watermarking algorithm that implements the idea of softly promoting randomly selected tokens (green tokens). They also provide an algorithm extension to Soft Red List that achieves security. They report that the watermarked texts remain of similar quality to non-watermarked texts and exhibit near-perfect detectability, especially when the number of tokens generated is at least 800 (Kirchenbauer et al., 2024b).

Extensions of this algorithm have been proposed. For example, Lee et al. (2024) have adapted the original algorithm to code generation by adding an entropy threshold condition to generate watermarked text.

Distortion-free watermarking One of the desirable properties of watermarked text is that the quality of the text remains similar to that of non-watermarked text. This similarity has been the focus of various papers claiming distortion-free watermarking, such as by Kuditipudi et al. (2024), Christ et al. (2023), and Hu et al. (2023).

Watermarking tradeoff Although the approaches we presented in the previous paragraph claim to provide simultaneously strong detectability and unperturbed text quality with their watermark schemes, other recent papers have challenged this claim by showing a tradeoff between text quality and detectability when watermarking the LLM's outputs. As already hinted at by Kirchenbauer et al. (2024a), it is currently established that there exists a fundamental tradeoff between text quality, detectability, and watermark robustness. Pang et al. (2024) show simple attacks that work against supposedly robust watermark detectors. Pan et al.

(2024) also reports a drop of perplexity for most currently developed watermark schemes with their benchmark.

Lee et al. (2024) use a Pareto frontier view of the problem between detectability and code capability when using watermarking. We will use a similar view for our experiments.

Watermarking robustness Another issue reported initially by Kirchenbauer et al. (2024a) and analyzed subsequently by Kirchenbauer et al. (2024b) is the robustness of watermarked text detection against evasion attacks such as paraphrasing and other similar text edition attacks - or also against spoofing attacks. This vulnerability is similar to those reported for LLM detectors that do not rely on watermarks, as reported by Krishna et al. (2023) and Sadasivan et al. (2024). A promising yet underdeveloped approach is semantic invariant watermarking. Liu et al. (2024) propose a possible implementation of this idea.

SynthID SynthID (Dathathri et al., 2024a;b) is a watermarking scheme developed by Google DeepMind and released publicly recently as of the time of writing. By testing with the Google's Gemma LLM, they claim to achieve a better detection/text quality tradeoff than previous approaches. They also show that users prefer SynthID watermarked and non-watermarked text equally.

However, the original implementation of SynthID published in the Hugging Face's Transformers library contained an implementation error, leading to significantly degraded performance, which we refer to as `SynthID (old version)`. We reported the implementation error[2], leading to a patched version, which we report as `SynthID (new version)`.

While their scheme potentially achieves our text quality/detection target after the fix, only the non-distortionary variant of their scheme is publicly available. They do not provide a straightforwards way to tune the FPR threshold for SynthID further, and do not share the code for their experiments mentioned. Moreover, follow-up research on intrinsic limitation of SynthID schema suggests a fundamental limitation on the detectability (Omidi et al., 2026), suggesting a drastic TPR drop for lower FPR targets, which are to be expected in realistic settings.

3. Methodology and experiments

All results can be replicated with the code provided in the benchmark repository: `https://github.com/Reliable-Information-Lab-HEVS/softer-red-list-llm-watermark`, published under the MIT license. English is the only language considered here, with all datasets, prompts, and fine-tuning data in English.

[2]issue link: `https://github.com/huggingface/transformers/issues/34630`

Unless stated otherwise, the generation was performed with a LLaMA-3.1-8B-Instruct model with generation and watermarking parameters specified in Appendix C.1.

3.1. Problem formulation and objective

Let:

- The generated token sequence be $y_1, y_2, \cdots, y_T$ where T is the total number of tokens and the first Q tokens are prefix tokens.
- $p(y_t|y_1, \cdots, y_{t-1}) = p(y_t|y_{<t}) = p^{(t)}$ is the vector of probability for each possible next token for token t knowing the context tokens 1 up to $t-1$. We denote $p_k^{(t)}$ the k^{th} component of this vector of next-token probability
- This next token probability vector corresponds to the softmax of the logits: $p^{(t)} = \frac{exp(l^{(t)})}{\sum_{i=1}^{|V|} exp(l_i^{(t)})}$ where $l_i^{(t)}$ is the logits for token i in the vocabulary V as given by the LLM.
- The biased probability distribution, as per the green-red token algorithm A is:

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

where δ is the watermark strength and G, R is a pseudo-random partition of the vocabulary V (partition seeded using a cryptographic key) such that $|G| = \gamma|V|$ and $|R| = (1-\gamma)R$

The objective for the watermarking scheme is then the following:

- Making the text detectable as watermarked (using the statistic test described in B.1), i.e. the expected number of generated green tokens should be higher for watermarked text than non-watermarked text. This expected number should be at least as high such that it is implausible that it is due to randomness. The higher this expected green token count is for a sequence, the better the detectability.
- The modified distribution for the next token $\hat{p}^{(t)}$ should be close enough to the original distribution $p^{(t)}$ so that the intervention is imperceptible. We usually use perplexity and its difference between two distributions as a proxy to measure this distribution shift.
- Achieve security as described in Appendix B.2.

3.2. Softer Red List Algorithm

We outline the original "Soft Red List" algorithm in Appendix A.

The proposed algorithm is similar to the original green/red token watermark algorithm with one fundamental change. Instead of considering all the tokens in the vocabulary for the green tokens, we only consider those with a large enough probability according to a cutoff method ("top-p" or "probability ratio"). We still have a random selection for the green tokens among these selected tokens depending on the parameter γ. However, the number of tokens considered by the new algorithm for the biasing is usually much smaller than in the original algorithm.

Algorithm 1 Text generation with Softer Red List

Input: prompt, $s^{(-N_p)}, \ldots, s^{(-1)}$
green list size $\gamma \in (0, 1)$
hardness parameter $\delta > 0$
Parametrized probability cutoff function $f : V \to V$

0: **for** $t = 0, 1, \ldots$ **do**
0:

1. Apply the language model to prior tokens $s^{(-N_p)}, \ldots, s^{(t-1)}$ to get a logit vector $l^{(t)}$ over the vocabulary.
2. Compute a hash of token $s^{(t-1)}$, and use it to seed a random number generator.
3. Sort the probabilities of next token such that $p_1^{(t)} \geq p_2^{(t)} \geq \cdots p_{|V|}^{(t)}$. Apply the probability "cutoff" function f to obtain the eligible tokens. These tokens will constitute the new vocabulary V_p.
4. Add each token in $V \setminus V_p$ (tokens in V not in V_p) to the red list R.
5. Using the random number generator, randomly partition the vocabulary V_p into a "green list" G_p of size $\gamma|V_p|$, and a "red list" R_p that we add to the red list R now of size $(1-\gamma)|V_p| + |V \setminus V_p|$.
6. Add δ to each green list logit. Apply the softmax operator to these modified logits to get a probability distribution over the vocabulary:
$$\hat{p}_k^{(t)} = \begin{cases} \frac{\exp(l_k^{(t)}+\delta)}{\sum_{i\in R}\exp(l_i^{(t)})+\sum_{i\in G}\exp(l_i^{(t)}+\delta)}, & \text{if } k \in G, \\ \frac{\exp(l_k^{(t)})}{\sum_{i\in R}\exp(l_i^{(t)})+\sum_{i\in G}\exp(l_i^{(t)}+\delta)}, & \text{if } k \in R. \end{cases}$$
7. Sample the next token $s^{(t)}$ using the watermarked distribution $\hat{p}^{(t)}$.

0: **end for**=0

Steps 3 to 5 differ from the original algorithm.

This addition is motivated by the observation that the original algorithm may boost any token, including highly unlikely ones. While the impact on a single token is negligible, the generated string will accumulate out-of-distribution tokens, leading to a an eventual noticeable drop in the generated text quality (**?**Holtzman et al., 2020). In this context, it is preferable to restrict the set of tokens we boost to positions presenting a high diversity of tokens that can be sampled at similar probabilities, preserving the text quality. The original algorithm circumvents this issue by only considering a small bias to the logits. Our method makes this bias parameter irrelevant and focuses on this probability threshold decision.

For the detection of watermarks, we rely on the same hypothesis test as proposed by Kirchenbauer et al. (2024a) that we repeat in appendix B. We did not consider in our work the issue of guaranteeing the confidentiality of the secret key used to seed the green/red token choices, but the proposed "Robust private watermarking" extension proposed by Kirchenbauer et al. (2024a) can be used together with our "Softer Red List" algorithm.

3.3. Probability cutoff functions

The proposed cutoff method in the new algorithm is a function $f : V \to V$. It shrinks the vocabulary V to a vocabulary V_p where usually $|V_p| \ll |V|$. The token eligibility depends on the token's probability after softmax. In our experiments, we consider two possible cutoff functions.

Probability ratio The first one - "probability ratio" (or p-ratio for short) - takes the probability of the most probable next token and multiplies it by the p-ratio to obtain the lower bound on the probability of the token the algorithm considers.

Top-p The second method, "top-p", sorts the token in reverse order of probability and considers all tokens until the cumulative probability reaches the top p-value - much like top-p sampling.

3.4. Generating the datasets for the experiments

We chose Llama-3.1-8B-instruct as the representative LLM for generating text samples and added Qwen2.5-Coder-7B-Instruct for code-generation tests (see Appendix D for the model links). For the general text detection, we use a dynamic news-like short texts detection benchmark, presented in Gameiro et al. (2024). The 500-character news-like content corresponds to one of the highest-risk use cases for LLMs in information operations, short length poses a challenge to other detectors and watermarks (Kirchenbauer et al., 2024b), while the benchmark's dynamic nature ensures that no tested detection method has overfit it.

To create LLM-generated natural texts, consistently with Gameiro et al. (2024) we leverage the CNN Dailymail news

dataset (see Appendix D), representative of American English news. To obtain the LLM-generated articles, we take a news article from the dataset, clean the beginning of the article to remove header content, and then pick the 10 first words of the article as a prefix. We use this prefix together with a chat prompt (see Appendix C.3). We let the model generate up to 200 tokens but cut the generation to have only the first 500 characters. We also cut the original articles to 500 characters to obtain the reference human samples. Using this procedure, human and generated samples are indistinguishable in length (see examples in appendix C.4). Using this method, we create for each watermarking scheme and Softer Red List parameter values in our parametric sweep (see section 4.4) a dataset of about 1000 eval and 1000 test samples (500 fake samples and 500 human samples for each split).

3.5. Evaluations

Evaluating watermarked text detection We generate datasets of 500 eval and 500 test samples for each watermarking scheme and specific parameter in the parametric sweep. This allows us to compute the threshold to obtain the target false positive rate (FPR) using the evaluation set and then compute the true positive rate (TPR) on the test set using this same threshold.

Evaluating watermarked text quality We perform several experiments to measure the distortion due to the different watermarking schemes. We use PPL as measured using an LLM different than the one generating the text to avoid a bias towards its own generations (we use Qwen-2-7B-Instruct). We use HumanEval+ (Liu et al., 2023) for generated code evaluation and GSM8K (Cobbe et al., 2021) for reasoning evaluation.

Evaluating watermark's robustness We consider paraphrasing as the attack of choice to test the watermark's robustness. Kirchenbauer et al. (2024b) and Dathathri et al. (2024b) demonstrated that this attack is particularly effective for low token count settings like ours (around 100-200 tokens in our datasets).

Watermarking schemes to evaluate We consider in our Kirchenbauer's et al. Soft Red List algorithm (Kirchenbauer et al., 2024a), SynthID (Dathathri et al., 2024a) and SWEET (Lee et al., 2024). This small selection of watermarking schemes allows us to compare them in different tasks and with different parameters. We would like to expand our benchmarks to more schemes in the future. The watermark implementations are based on this repo for the Soft Red List and SWEET algorithm: `https://github.com/THU-BPM/MarkLLM/tree/main`. We used HuggingFace's implementation for SynthID as of Transformers version 4.46.2 for the "old version" and as of version 4.47.0 for the patched "new version".

4. Results and Discussion

4.1. Detectability benchmark and PPL benchmark

We created datasets of human-written news/LLM-written news using different watermarking schemes (see 3.4 for the details of the dataset creation). For each watermarking scheme, we created two targets representing text quality and detection. The text quality target is met when the perplexity (PPL) for the generated CNN articles is at most 5% higher than the non-watermarked text PPL (high-quality version). The detectability target is met when the detection TPR@1%FPR is at least 0.95. When both targets could not be met at the same by a watermarking scheme, we created two versions of that scheme where each version meets one of the targets through a different set of parameters. To detect the watermarking, we used the corresponding watermark detector for each scheme, except for the non-watermarked text, where we used Fast-DetectGPT as the detector.

The results are shown in Tab. 1 in the CNN articles columns for our benchmark.

We notice that only our scheme achieves both targets. SynthID almost achieves both as well, but cannot achieve a higher than 92% TPR@1%FPR value under the non-distortionary variant (the one publicly available.) SWEET watermarking and Kirchenbauer's green/red token algorithm almost achieve both targets as well, but still with a considerable increase in PPL for the high detectability version to 7.75 and 7.86 respectively (compared to 6.59 for the non-watermarked text). The old version of SynthID falls completely short of achieving both targets, achieving a PPL of 9.48 for the high detectability version, highlighting the issue with the previous Huggingface's implementation we mentioned earlier in section 2.2.

Overall, our scheme achieves the best PPL and detectability tradeoff for our CNN news detection benchmark.

4.2. Reasoning and coding benchmarks

We ran the HumanEval+ benchmark using a modified version of BigCode eval and similarly for GSM8K using lm-evaluation-harness (see Appendix D) to support watermarking. We used the same parameters to obtain the results in all tasks in Tab. 1. Reusing the parameters obtained by testing in our benchmark allows us to test the transferability of the watermark configurations to different tasks. Human-eval+ and GSM8k task results are available in Tab. 1.

Our results suggest that our parameter tuning does not fully transfer to the coding task. Indeed, our scheme achieves a much lower Pass@1 score on HumanEval+ than on the non-

Watermarking scheme	Dataset + Metric CNN articles TPR@5%FPR ↑	CNN articles TPR@1%FPR ↑	CNN articles PPL ↓	Human-eval+ Pass@1 ↑	GSM8k 8-shots acc ↑
Non-watermarked	0.96 ± 0.01	0.88 ± 0.01	6.59 ± 0.13	22.2	82.2
High quality target					
Kirchenbauer	**0.69** ± 0.02	0.47 ± 0.02	6.83 ± 0.12	↓2.4 19.8	↓2.6 79.6
SWEET	0.64 ± 0.02	**0.48** ± 0.02	6.88 ± 0.13	↓2.7 19.5	↑0.7 82.9
SynthID (old version)	0.46 ± 0.02	0.13 ± 0.01	6.71 ± 0.13	↓4.2 18.0	↓0.6 81.6
High detectability target					
Kirchenbauer	0.998 ± 0.01	0.98 ± 0.01	8.38 ± 0.17	↓14.9 7.3	↓7.9 74.3
SWEET	0.98 ± 0.006	0.96 ± 0.01	**7.75** ± 0.18	↓12.3 **9.9**	↓2.2 **80.06**
SynthID (old version)	0.996 ± 0.003	0.96 ± 0.01	9.48 ± 0.30	↓14.3 7.9	↓17.3 64.9
Both targets					
SynthID (new version)*	**0.99** ± 0.005	0.92 ± 0.01	**6.58** ± 0.12	↑1.1 **23.3**	↓17.3 64.9
Ours	0.98 ± 0.006	**0.96** ± 0.01	6.67 ± 0.11	↓12.0 10.2	↓4.4 **77.8**

Table 1. Detectability-Quality tradeoff comparison on our benchmark. High detectability target is TPR@1%FPR over 95%. High quality - a PPL at most 5% over the non-watermarked. * indicates that SynthID could not be tuned to the high detectability but was selected for direct comparison as the closest in PPL/detectability tradeoff. Fast-DetectGPT was used for non-watermarked text detection.

Watermarking scheme	Human-eval+ Pass@1 ↑ Llama-3.1-8B	Qwen2.5-Coder-7B
Non-watermarked	22.2	55.07
Kirchenbauer(high quality)	19.8	54.18
SWEET (high quality)	19.5	52.68
Kirchenbauer(high detectability)	7.29	32.66
SWEET (high detectability)	9.9	44.12
SynthID (new version)	**23.3**	**57.9**
Ours	10.2	46.87

Table 2. Watermarking effect on the Human-eval+ Pass@1 metric. High detectability and quality targets are the same as before.

watermarked test, while maintaining a high performance score on the GSM8k reasoning task. While this might suggest that our scheme's parameters need retuning to work with code generation, it might also reflect an intrinsic limitation of our method. While SynthID's performance drops off faster on GSM8k than ours does, it is the only one to match the non-watermarked model in Human-eval+, and the only one not to bias the token probabilities. Outside SynthID, even SWEET - developed specifically for code watermarking - performs comparably to Soft Red.

In Tab. 2, we evaluate the watermarking schemes using the same coding benchmark but with a specialized coding LLM with a higher base performance. Using a different model, we obtain a smaller decrease in Pass@1 score compared to when using Llama-3.1 (15% vs 55% respectively). This suggests a level of variability in the results when testing with various models, although supports the hypothesis of an intrinsic limitation of bias-based watermarks for code watermarking.

4.3. Paraphrasing resistance

We generated text in the same way as Tab.1 with the high detection watermark parameter and then paraphrased the texts with Llama-3.1-Instruct using the paraphrasing prompt in appendix C.5. The TPR@5% FPR results are shown in Tab. 3.

Watermarking scheme	CNN articles TPR@5%FPR ↑ No attack	Paraphrasing attack
Non-watermarked	0.96 ± 0.01	0.56 ± 0.01
Kirchenbauer(high detectability)	**0.998** ± 0.01	0.12 ± 0.008
SWEET (high detectability)	0.98 ± 0.006	0.15 ± 0.02
SynthID	0.99 ± 0.005	0.09 ± 0.01
Ours	0.98 ± 0.006	**0.21** ± 0.02

Table 3. Watermarking detection vulnerability against paraphrasing attack. High detectability target is the same as before. Fast-DetectGPT was used for non-watermarked text detection.

We observe that none of the watermarking detectors achieve a detection above 20% TPR under the 5% FPR target for the paraphrased watermarked text, far below the 56% TPR achieved by a baseline LLM detector. While Softer Red List resits the detectability loss the best, it is not sufficient and confirms once again the vulnerability of current watermarking schemes to reformulation (Kirchenbauer et al., 2024b).

4.4. Parametric sweep

We use the experimental pipeline presented in part 4.1 in order to perform a parametric sweep, focusing on one parameter at a time, demonstrating the controllability of the Sweeter Red List. We used a vertical red line to indicate the PPL target of at most 5% more than the non-watermarked text PPL and a blue horizontal line for the detection target. The detection target here is for TPR@5% rather than TPR@1%, which is more permissive than the one on Tab. 1. Additionally, we circled the parameters that achieve both targets for the different scenarios in yellow.

p-ratio cutoff Fig. 1, diamond markers, shows the impact of the varying p-ratio cutoff, covering the range between 10^{-5} and 0.5. The former corresponds to the extreme case where almost all tokens are eligible for logit boosting (working as the original "Soft Red List"), while the latter - to the case where only the tokens that are 50% or more as likely as the top token. Values of p-ratio close to 0 lead to higher detectability but higher PPL, as expected, given the observed behavior of "Soft Red List". A higher p-ratio makes it impossible for the algorithm to amplify highly unlikely tokens, improving the PPL but making the watermark less detectable.

Top-p cutoff Fig. 1, plus markers, shows the impact of the varying top-p cutoff, covering the range between 0.5 and 0.95, covering most common values and stopping at the base model sampling top-p. While the overall trend is unsurprisingly similar to the tradeoff with the p-ratio cutoff, the p-ratio cutoff achieves a significantly better tradeoff. Overall, a p-ratio cutoff between 0.1 and 0.25 seems optimal, where both the 95% TPR target and the target PPL range are reached.

We hypothesize that the p-ratio cutoff performs better due to its ability to gracefully handle both watermarking-friendly token positions, where many tokens are possible with similar likelihoods, hence being eligible for amplification without a noticeable impact on PPL, and watermarking-incompatible token positions, where a single token is overwhelmingly likely, and selection of any other token would strongly impact the PPL. We believe this analysis explains the watermark detectability-quality tradeoff highlighted by Pang et al. (2024).

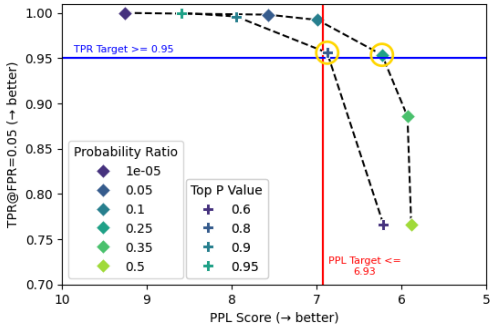

Figure 1. Tradeoff between PPL and detectability (TPR @ 5% FPR) with top-p value vs prob ratio cutoff

Influence of entropy threshold Introduced by Lee et al. (2024), entropy thresholding is an extension to the original green/red token algorithm, where the entropy of the next token distribution is calculated before a decision is made to apply the watermark. The goal of this addition is to prevent adding a bias in a watermarking-incompatible position, where only a few tokens are likely and any bias would strongly impact the overall text quality. Fig. 2 shows the influence of the entropy threshold for our watermarking scheme. Given that in our setting, the watermarking-incompatible token positions are well-managed by p-ratio, the entropy threshold offers a minimal advantage. Considering the computational overhead of using it, we did not include it in the Softer Red List implementation and excluded it from any further experiments.

Variable bias strength Fig. 3 shows how varying the bias strength (watermark delta parameter) influences the PPL/detectability tradeoff.

While a low watermark delta value leads to poor detectability, we observe that increasing this value offers no benefit past some threshold. A possible explanation is that when the watermark delta is high enough, the biasing is enough to make the sampling almost deterministic toward generating green tokens. Similarly, the impact of the bias value is limited for PPL. We suggest that the value for the bias is not critical in our watermarking scheme. Our parametric sweep suggests that probability thresholding plays the most important role in the PPL/detectability tradeoff.

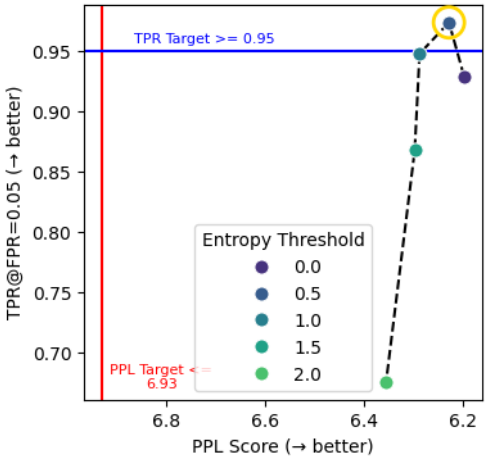

Figure 2. Tradeoff between PPL and detectability (TPR @ 5% FPR) as function of the entropy threshold

5. Conclusion

In our paper, we presented a new SotA LLM watermarking algorithm, achieving both a high detectability and generation quality, Softer Red List. The probability cutoff trick that allowed us to derive it from the Soft Red List is not restricted to it and can also be applied to other watermarking schemes, allowing for tight control over the detectability-quality tradeoff. Moreover, we introduced a new dynamic benchmark for watermarking scheme testing, more directly

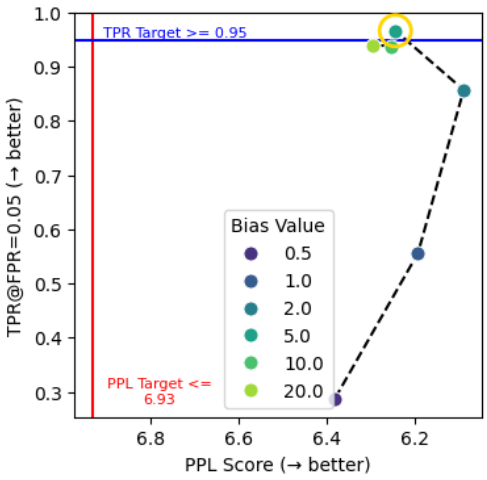

Figure 3. Tradeoff between PPL and detectability (TPR @ 5% FPR) as function of the watermark delta (bias value)

relevant to the real-world setting of LLM watermarking usage. We showed that the detectability-quality tradeoff is harder to characterize than prior work suggested, especially when more recent metrics, such as coding and reasoning task success, are considered. While our method did not match SynthID in the code-generation quality-detectability tradeoff, it achieved it across all natural language tasks, allowing for tighter control over the FPR-TPR tradeoff. We hope that our work will encourage further improvement of LLM watermarking testing practices and drive the development of better watermarks.

Acknowledgments

The authors are thankful to Dr. Ljiljana Dolamic and Prof. Antoine Bosselut for their valuable input on the paper, and to the anonymous reviewers of Swiss AI Days'26 and COLM'25 for their constructive comments. We additionally thank Prof. Bosselut for hosting Henrique Da Silva Gameiro (HDSG) in his lab during the course of this work. Andrei Kucharavy (AK) is supported by the CYD Campus, armasuisse W+T, ARAMIS AR-CYD-C-025 grant.

Limitations

While we believe our work has a general scope in terms of impact through a simple and motivated addition to Kirchenbauer et Al.'s already comprehensive work, we noted several assumptions that we made to limit the resources in time and compute needed for our experiments.

Firstly, our CNN news article benchmark only comprises text of 500 characters (around 200 tokens). We believe that varying the text length has been studied enough in previous work, such as by Kirchenbauer et al. (2024b), and that the relationship between text length and detectability is, therefore, well understood.

Secondly, we only considered a limited number of models and datasets in our experiments. While our results show that both elements have a critical impact, they also show that statically generated benchmarks are ineffective. We, therefore, believe that it is more useful to provide adaptable benchmarks for given attacks and environments than general-purpose static benchmarks.

Additionally, we left out several watermarking schemes. Some schemes, such as the one proposed by Liu et al. (2024), could achieve better robustness results through the "semantic watermarking" idea of linking the watermarking with the semantics of the context through the embedding. In our current work, we decided to study each scheme in detail rather than test more schemes, but we would like to expand our experiments to include more schemes in the future.

Finally, although we studied the impacts of the different parameters of our scheme through a parametric sweep, some aspects would still benefit from more study. In particular, our central proposal - the probability cutoff method. Different alternatives could be interesting to consider. Another interesting study we omitted is how each function compares in terms of different factors, such as smoothness and number of tokens considered by the cutoff. Future work could also focus on other understudied aspects of our work, such as a better understanding of the bias (delta) parameter in our scheme, as well as the gamma and entropy threshold parameters.

Ethics Statement

To generate the datasets, we used 1 A100 GPU on a local cluster. Each dataset takes around 3 minutes to be generated in that manner, and the code/reasoning benchmark takes around 3 hours each to be completed under our setup. In total, we used the local cluster for about 100 hours of GPU-days on the A100 (numbers including hyperparameter search and testing correctness), leading to total emissions of 10.8kg of CO2. No crowdsourced labor was used in this work. LLM assistants were used to make minor

stylistic and grammatical corrections to the final manuscript. GitHub Copilot has been used to assist in coding with autocompletion, but no script or algorithm has been fully generated with it.

References

Adelani, D. I., Mai, H. T., Fang, F., Nguyen, H. H., Yamagishi, J., and Echizen, I. Generating sentiment-preserving fake online reviews using neural language models and their human- and machine-based detection. In *AINA*, 2020.

Bakhtin, A., Gross, S., Ott, M., Deng, Y., Ranzato, M., and Szlam, A. Real or fake? learning to discriminate machine from human generated text. *CoRR*, abs/1906.03351, 2019. URL http://arxiv.org/abs/1906.03351.

Bao, G., Zhao, Y., Teng, Z., Yang, L., and Zhang, Y. Fast-DetectGPT: Efficient Zero-Shot Detection of Machine-Generated Text via Conditional Probability Curvature, February 2024. URL http://arxiv.org/abs/2310.05130. arXiv:2310.05130 [cs].

Brooks, C., Eggert, S., and Peskoff, D. The Rise of AI-Generated Content in Wikipedia, October 2024. URL http://arxiv.org/abs/2410.08044. arXiv:2410.08044.

Christ, M., Gunn, S., and Zamir, O. Undetectable Watermarks for Language Models, May 2023. URL http://arxiv.org/abs/2306.09194. arXiv:2306.09194.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

Dathathri, S., See, A., Ghaisas, S., Huang, P.-S., McAdam, R., Welbl, J., Bachani, V., Kaskasoli, A., Stanforth, R., Matejovicova, T., Hayes, J., Vyas, N., Merey, M. A., Brown-Cohen, J., Bunel, R., Balle, B., Cemgil, T., Ahmed, Z., Stacpoole, K., Shumailov, I., Baetu, C., Gowal, S., Hassabis, D., and Kohli, P. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823, October 2024a. ISSN 1476-4687. doi: 10.1038/s41586-024-08025-4. URL https://www.nature.com/articles/s41586-024-08025-4. Publisher: Nature Publishing Group.

Dathathri, S., See, A., Ghaisas, S., Huang, P.-S., McAdam, R., Welbl, J., Bachani, V., Kaskasoli, A., Stanforth, R., Matejovicova, T., Hayes, J., Vyas, N., Merey, M. A., Brown-Cohen, J., Bunel, R., Balle, B., Cemgil, T., Ahmed, Z., Stacpoole, K., Shumailov, I., Baetu, C., Gowal, S., Hassabis, D., and Kohli, P. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823, October 2024b. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-024-08025-4. URL https://www.nature.com/articles/s41586-024-08025-4.

Dugan, L., Hwang, A., Trhlik, F., Ludan, J. M., Zhu, A., Xu, H., Ippolito, D., and Callison-Burch, C. Raid: A shared benchmark for robust evaluation of machine-generated text detectors. *ArXiv*, abs/2405.07940, 2024. URL https://api.semanticscholar.org/CorpusID:269757699.

Gagiano, R., Kim, M., Zhang, X., and Biggs, J. Robustness analysis of grover for machine-generated news detection. In *ALTA*, 2021.

Gameiro, H. D. S., Kucharavy, A., and Dolamic, L. LLM Detectors Still Fall Short of Real World: Case of LLM-Generated Short News-Like Posts, September 2024. URL http://arxiv.org/abs/2409.03291. arXiv:2409.03291 [cs].

Goldstein, J. A., Sastry, G., Musser, M., DiResta, R., Gentzel, M., and Sedova, K. Generative language models and automated influence operations: Emerging threats and potential mitigations. *ArXiv*, abs/2301.04246, 2023. URL https://api.semanticscholar.org/CorpusID:255595557.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

Hu, Z., Chen, L., Wu, X., Wu, Y., Zhang, H., and Huang, H. Unbiased Watermark for Large Language Models, October 2023. URL http://arxiv.org/abs/2310.10669. arXiv:2310.10669.

Ippolito, D., Duckworth, D., Callison-Burch, C., and Eck, D. Automatic detection of generated text is easiest when humans are fooled. In *ACL*, 2020.

Kirchenbauer, J., Geiping, J., Wen, Y., Katz, J., Miers, I., and Goldstein, T. A Watermark for Large Language Models, May 2024a. URL http://arxiv.org/abs/2301.10226. arXiv:2301.10226.

Kirchenbauer, J., Geiping, J., Wen, Y., Shu, M., Saifullah, K., Kong, K., Fernando, K., Saha, A., Goldblum, M., and Goldstein, T. On the Reliability of Watermarks for Large Language Models, May 2024b. URL http://arxiv.org/abs/2306.04634. arXiv:2306.04634.

Krishna, K., Song, Y., Karpinska, M., Wieting, J., and Iyyer, M. Paraphrasing evades detectors of AI-generated text, but retrieval is an effective defense, October 2023. URL http://arxiv.org/abs/2303.13408. arXiv:2303.13408 [cs].

Kuditipudi, R., Thickstun, J., Hashimoto, T., and Liang, P. Robust Distortion-free Watermarks for Language Models, June 2024. URL http://arxiv.org/abs/2307.15593. arXiv:2307.15593.

Latona, G. R., Ribeiro, M. H., Davidson, T. R., Veselovsky, V., and West, R. The AI Review Lottery: Widespread AI-Assisted Peer Reviews Boost Paper Scores and Acceptance Rates, May 2024. URL http://arxiv.org/abs/2405.02150. arXiv:2405.02150.

Lee, T., Hong, S., Ahn, J., Hong, I., Lee, H., Yun, S., Shin, J., and Kim, G. Who Wrote this Code? Watermarking for Code Generation, July 2024. URL http://arxiv.org/abs/2305.15060. arXiv:2305.15060 [cs].

Liang, W., Yuksekgonul, M., Mao, Y., Wu, E., and Zou, J. GPT detectors are biased against non-native English writers, July 2023. URL http://arxiv.org/abs/2304.02819. arXiv:2304.02819 [cs].

Liu, A., Pan, L., Hu, X., Meng, S., and Wen, L. A Semantic Invariant Robust Watermark for Large Language Models, May 2024. URL http://arxiv.org/abs/2310.06356. arXiv:2310.06356.

Liu, J., Xia, C. S., Wang, Y., and Zhang, L. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/43e9d647ccd3e4b7b5baab53f0368686-Abstract-Conference.html.

Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D., and Finn, C. DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature, July 2023. URL http://arxiv.org/abs/2301.11305. arXiv:2301.11305 [cs].

Omidi, R., Dong, Y., and Wang, B. On google's synthid-text llm watermarking system: Theoretical analysis and empirical validation, 2026. URL https://arxiv.org/abs/2603.03410.

Pan, L., Liu, A., He, Z., Gao, Z., Zhao, X., Lu, Y., Zhou, B., Liu, S., Hu, X., Wen, L., King, I., and Yu, P. S. MarkLLM: An Open-Source Toolkit for LLM Watermarking, October 2024. URL http://arxiv.org/abs/2405.10051. arXiv:2405.10051.

Pang, Q., Hu, S., Zheng, W., and Smith, V. No Free Lunch in LLM Watermarking: Trade-offs in Watermarking Design Choices, May 2024. URL http://arxiv.org/abs/2402.16187. arXiv:2402.16187.

Sadasivan, V. S., Kumar, A., Balasubramanian, S., Wang, W., and Feizi, S. Can AI-Generated Text be Reliably Detected?, February 2024. URL http://arxiv.org/abs/2303.11156. arXiv:2303.11156 [cs].

Solaiman, I., Brundage, M., Clark, J., Askell, A., Herbert-Voss, A., Wu, J., Radford, A., and Wang, J. Release strategies and the social impacts of language models. *ArXiv*, abs/1908.09203, 2019.

Stiff, H. and Johansson, F. Detecting computer-generated disinformation. *International Journal of Data Science and Analytics*, 13:363–383, 2022.

Tian, E. and Cui, A. Gptzero: Towards detection of ai-generated text using zero-shot and supervised methods, 2024. URL https://gptzero.me.

Wang, Y., Mansurov, J., Ivanov, P., Su, J., Shelmanov, A., Tsvigun, A., Whitehouse, C., Mohammed Afzal, O., Mahmoud, T., Sasaki, T., Arnold, T., Aji, A. F., Habash, N., Gurevych, I., and Nakov, P. M4: Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection. pp. 1369–1407, March 2024. URL https://aclanthology.org/2024.eacl-long.83/.

Zellers, R., Holtzman, A., Rashkin, H., Bisk, Y., Farhadi, A., Roesner, F., and Choi, Y. Defending against neural fake news. In *NeurIPS*, 2019.

A. Algorithms

Algorithm 2 Text Generation with Soft Red List (Kirchenbauer 2023)

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

0: **for** $t = 0, 1, \ldots$ **do**
0:

1. Apply the language model to prior tokens $s^{(-N_p)}, \ldots, s^{(t-1)}$ to get a logit vector $l^{(t)}$ over the vocabulary.
2. Compute a hash of token $s^{(t-1)}$, and use it to seed a random number generator.
3. Using this random number generator, randomly partition the vocabulary into a "green list" G of size $\gamma|V|$, and a "red list" R of size $(1-\gamma)|V|$.
4. Add δ to each green list logit. Apply the softmax operator to these modified logits to get a probability distribution over the vocabulary:
$$\hat{p}_k^{(t)} = \begin{cases} \frac{\exp(l_k^{(t)}+\delta)}{\sum_{i\in R}\exp(l_i^{(t)})+\sum_{i\in G}\exp(l_i^{(t)}+\delta)}, & \text{if } k \in G, \\ \frac{\exp(l_k^{(t)})}{\sum_{i\in R}\exp(l_i^{(t)})+\sum_{i\in G}\exp(l_i^{(t)}+\delta)}, & \text{if } k \in R. \end{cases}$$
5. Sample the next token $s^{(t)}$ using the watermarked distribution $\hat{p}^{(t)}$.

0: **end for**=0

B. Watermark detection and security

For both aspects, we don't change anything from the original proposal in the green/red token algorithm (Kirchenbauer et al., 2024a).

B.1. Watermark detection

We use the same hypothesis framework as Kirchenbauer et al. (2024a) with the null hypothesis being the hypothesis that the text is not watermarked and the following hypothesis test using the z score:

$$z = (C_G - \gamma T)/\sqrt{T\gamma(1-\gamma)}$$

We tune the threshold on the test on an evaluation set to obtain the desired false-positive rate target.

B.2. Watermark security

We refer to the "Private watermarking" section in Kirchenbauer et al. (2024a)'s work. While they outline the importance of having a large context window for the watermark seeding, we consider here the naive approach by only selecting the previous token for the seeding since we don't focus on the aspect of security and choosing the right "prefix length" to achieve that. We however strongly recommend following the advice from the original paper and using a large "prefix length" parameter to make it harder for the attacker to find the watermark secret key through brute-force.

C. Datasets and Data Generation Details

C.1. Generation Parameters

We use a temperature of 0.8, top_p of 0.95, top_k of 50, and repetition penalty of 1 for the sampling.

C.2. Watermark parameters

We highlight the parameters used for the different watermarking schemes variants in Tab. 1, Tab.3, and Tab. 2. The parameters were not found by rigorous hyperparameter search but rather by taking the range of the parameters suggested by

the different papers and slightly tuning the parameters until we fit the target detection/quality threshold for Tab. 1. For that tuning, high quality target means a PPL of at most 5% more than the unwatermarked text's PPL. High detectability target is a TPR@1% FPR of at least 95%. The different parameters used for each watermarking scheme are as follows:

Soft Red List (Kirchenbauer et al.)

- High quality version: gamma = 0.5, delta = 1.0
- High detectability version: gamma = 0.5, delta = 2.0

SWEET (Lee et al.)

- High quality version: gamma = 0.5, delta = 1.0, entropy threshold 1.2
- High detectability version: gamma = 0.5, delta = 2.5, entropy threshold 1.2

SynthID old version (Dathathri et al.)

- High quality version: N-gram size = 1, key = 1 (i.e. list of size 1). Weighted mean detector.
- High detectability version: N-gram size = 5, key = 30 (i.e. list of size 30). Weighted mean detector.

SynthID new version (Dathathri et al.)

- N-gram size = 5, key = 30 (i.e. list of size 30). Weighted mean detector.

Softer Red List (Ours)

- gamma = 0.5, delta = 5.0, cut_off_method = "ratio", prob_ratio = 0.15, no entropy threshold

C.3. Prompts for Generating Data

We use the prompt in table 4. Also, we force the first tokens of the output to be the prefix for that particular sample. This prevents the chat model from generating typical assistant-starting messages that would be too obvious to spot for the detectors (in practice, an attacker could also remove it easily).

System Prompt	User Prompt
You are a helpful assistant.	Continue to write this news article: {prefix}

Table 4. Prompt used to generate the fake news articles with the chat models.

C.4. Dataset Example

We provide below an example of a true and a LLM-generated news article. As explained in the methodology, samples are regrouped by pairs of true and fake samples (ordered randomly within the pair). The true and fake samples in the pair start with the same 10-word prefix. **True sample:**
"Shopping in the Chinese city of Shenyang is very similar to shopping anywhere in the world... very similar indeed. Just pop down to Wanda Square in the heart of the city and you'll see a cornucopia of well-known brands - all with incredible knock-off prices. Start your spree at HERMES PARIS before heading on to CHANEL for perfume, PRADA for clothes and Cairter or Tifeany & Co for that jewellery you always wanted. If wobbly shopping legs begin to take . hold you can always stop for some ice-cream"

source: CNN news dataset (https://huggingface.co/datasets/abisee/cnn_dailymail)

Fake sample:
"Shopping in the Chinese city of Shenyang is very similar to shopping in any other major Chinese city, with a mix of traditional and modern markets and malls. The city has a wide range of shopping options, from high-end department stores and luxury brands to traditional street markets and night markets. The city also has a number of popular shopping districts, such as the Shenyang Financial Street and the Wanghua Shopping Center, where visitors can find everything from clothing and electronics to"

source: Llama-3.1-Instruct with Softer red list watermarking

C.5. Attack Prompts

Attack prompt - Paraphrasing prompt

System Prompt: You are a paraphraser. You are given an input passage 'INPUT'. You should paraphrase 'INPUT' to print 'OUTPUT'." "'OUTPUT' shoud be diverse and different as much as possible from 'INPUT' and should not copy any part verbatim from 'INPUT'." "'OUTPUT' should preserve the meaning and content of 'INPUT' while maintaining text quality and grammar." "'OUTPUT' should not be much longer than 'INPUT'. You should print 'OUTPUT' and nothing else so that its easy for me to parse.

User Prompt: INPUT: {fake_text}"

Figure 4. Prompt used to paraphrase an LLM-generated news article. The prompt is taken from Sadasivan et al. (2024).

D. Models, Datasets and Third-Party Code

Name	Retrieved From
LLama-3.1-8B-Instruct	Hugging Face link: https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct
Qwen2-7B-Instruct	Hugging Face link: https://huggingface.co/Qwen/Qwen2-7B-Instruct
CNN Dailymail	Hugging Face link: https://huggingface.co/datasets/cnn_dailymail
Qwen2.5-Coder-7B-Instruct	Hugging Face link: https://huggingface.co/Qwen/Qwen2.5-Coder-7B-Instruct
BigCode evaluation harness (BigCode project)	GitHub link: https://github.com/bigcode-project/bigcode-evaluation-harness
LM evaluation harness (EleutherAI)	GitHub link: https://github.com/EleutherAI/lm-evaluation-harness
MarkLLM (Open source watermarking toolkit)	GitHub link: https://github.com/THU-BPM/MarkLLM

Table 5. Urls from which models and code were retrieved

Attack	Detector	Datasetl		
		Gemma Chat	Zephyr	Avg.
High temperature	Roberta_Phi-2	0.9912*	0.9268	
	Roberta_Phi-2	0.9890	**0.9721**	
	(Diff)	*-0.0021*	*0.0452*	
	Fast-DetectGPT (No attack)	0.7365	0.8713	
	Fast-DetectGPT (With attack)	0.8886		
	(Diff)	*0.1521*	*0.0495*	

Generator	Detector	Attack	
		High temperature	Repetition penalty
Zephyr	Electra_Phi-2 (No attack)	0.99	0.99
	Electra_Phi-2 (With attack)	↓0.20 0.76	↓0.12 0.82
	(Diff)	*-0.20*	*-0.14*
	OpenAI(No attack)	-	-
	OpenAI(With attack)	-	**-**
	(Diff)	*-0.04*	
	Fast-DetectGPT (No attack)	0.92	0.92
	Fast-DetectGPT (With attack)	-	**-**
	(Diff)	0.07	*-0.22*

Generator	Detector	Attack		
		No Attack	High temperature	Repetition penalty
Zephyr	Electra_Phi	0.99	0.20 0.76	↓0.12 0.82
	OpenAI	0.99	-	-
	Fast-DetectGPT	0.92	-	**-**

Generator	Detector	Attack			
		News prompt	Tweet prompt	Paraphrasing	Example prompt
Zephyr	Electra_Phi-2 (Best trained) (No attack)	0.99	0.99	0.99	0.99
	Electra_Phi-2 (Best trained) (With attack)	-	**0.56**	0.92	0.86
	(Diff)	-	-	-	-
	OpenAI(No attack)	-	-	-	**-** -
	OpenAI(With attack)	-	**-**	-	- -
	(Diff)	-	-	-	-
	Fast-DetectGPT (No attack)	0.92	0.92	0.92	**0.92**
	Fast-DetectGPT (With attack)	-	**-**	-	-
	(Diff)	*-0.07*	*-0.22*	*-0.04*	*-0.11*

Generator	Detector	Attack				
Avg.		High temperature	Repetition penalty	News prompt	Tweet prompt	Paraphrasing
Gemma chat	Roberta_Phi-2 (No attack)	0.97	0.97	0.97	0.97	0.97
	Roberta_Phi-2 (With attack)	0.91/0.0.93	0.89/0.93	0.94/0.94	**0.63/0.93**	0.90/0.94
	(Diff)	*-0.06*	*-0.08*	*-0.03*	*-0.34*	*-0.07*
	Roberta_gemma (No attack)	0.94	0.94	0.94	0.94	0.94
	Roberta_gemma (With attack)	0.89	0.88	0.90	**0.83**	0.84
	(Diff)	*-0.05*	*-0.06*	*-0.04*	*-0.11*	*-0.10*
	Roberta_mistral (No attack)	0.93	0.93	0.93	0.93	0.93
	Roberta_mistral (With attack)	0.88	0.79	0.88	**0.67**	0.86
	(Diff)	*-0.05*	*-0.14*	*-0.05*	*-0.26*	*-0.07*
	Roberta_RR (No attack)	0.90	0.90	0.90	0.90	0.90
	Roberta_RR(With attack)	0.95	0.98	0.92	**0.98**	0.96
	(Diff)	*+0.05*	*+0.08*	*+0.02*	*+0.08*	*+0.06*
	OpenAI(No attack)	0.82	0.82	0.82	0.82	0.82
	OpenAI(With attack)	0.76	**0.62**	0.82	0.63	0.87
	(Diff)	*-0.06*	*-0.20*	*-0.00*	*-0.19*	*+0.05*
	Fast-DetectGPT (No attack)	0.95	0.95	0.95	0.95	0.95
	Fast-DetectGPT (With attack)	0.87	0.79	0.93	0.84	**0.76**
	(Diff)	*-0.08*	*-0.16*	*-0.02*	*-0.11*	*-0.19*
Zephyr	Roberta_Phi-2 (No attack)	0.96	0.96	0.96	0.96	0.96
	Roberta_Phi-2 (With attack)	**0.84**	0.88	0.90	**0.84**	0.90
	(Diff)	*-0.12*	*-0.08*	*-0.06*	*-0.12*	*-0.06*
	Roberta_gemma (No attack)	0.94	0.94	0.94	0.94	0.94
	Roberta_gemma (With attack)	0.89	**0.77**	0.90	0.79	0.84
	(Diff)	*-0.05*	*-0.17*	*-0.04*	*-0.15*	*-0.10*
	Roberta_mistral (No attack)	0.93	0.93	0.93	**0.93**	0.93
	Roberta_mistral (With attack)	0.82	0.74	0.85	**0.73**	0.86
	(Diff)	*-0.11*	*-0.19*	*-0.08*	*-0.20*	*-0.07*
	Roberta_RR (No attack)	0.88	0.88	0.8	**0.88**	0.88
	Roberta_RR(With attack)	0.85	0.90	0.87	**0.93**	0.96
	(Diff)	*-0.03*	*+0.02*	*+0.07*	*+0.05*	*+0.06*
	OpenAI(No attack)	0.70	0.70	0.70	**0.70**	0.70
	OpenAI(With attack)	0.66	**0.54**	0.73	0.60	0.87
	(Diff)	*-0.04*	*-0.16*	*+0.03*	*-0.10*	*+0.05*
	Fast-DetectGPT (No attack)	0.94	0.94	0.94	**0.94**	0.94
	Fast-DetectGPT (With attack)	0.87	**0.72**	0.90	0.83	0.76
	(Diff)	*-0.07*	*-0.22*	*-0.04*	*-0.11*	*-0.19*

Generator	Detector	Attack				
Avg.		High temperature	Repetition penalty	News prompt	Tweet prompt	Paraphrasing
Gemma chat	Roberta_Phi-2 (No attack)	0.97	0.97	0.97	0.97	0.97
	Roberta_Phi-2 (With attack)	0.88	0.84	0.94	**0.31**	0.86
	(Diff)	*-0.09*	*-0.13*	*-0.03*	*-0.66*	*-0.11*
	Roberta_gemma (No attack)	0.93	0.93	0.93	0.93	0.93
	Roberta_gemma (With attack)	0.84	0.82	0.88	**0.72**	0.74
	(Diff)	*-0.09*	*-0.11*	*-0.05*	*-0.21*	*-0.19*
	Roberta_mistral (No attack)	0.91	0.91	0.91	0.91	0.91
	Roberta_mistral (With attack)	0.84	0.66	0.84	**0.42**	0.80
	(Diff)	*-0.07*	*-0.25*	*-0.07*	*-0.49*	*-0.11*
	Roberta_RR (No attack)	-	-	-	-	-
	Roberta_RR (With attack)	-	-	-	**-**	-
	(Diff)	*+0.05*	*+0.08*	*+0.02*	*+0.08*	*+0.06*
	OpenAI(No attack)	-	-	-	-	-
	OpenAI(With attack)	-	**-**	-	-	-
	(Diff)	*-0.06*	*-0.20*	*-0.00*	*-0.19*	*+0.05*
	Fast-DetectGPT (No attack)	0.98	0.98	0.98	0.98	0.98
	Fast-DetectGPT (With attack)	-	-	-	-	**-**
	(Diff)	*-0.08*	*-0.16*	*-0.02*	*-0.11*	*-0.19*
Zephyr	Roberta_Phi-2 (No attack)	0.96	0.96	0.96	0.96	0.96
	Roberta_Phi-2 (With attack)	0.76	0.82	0.86	**0.74**	0.86
	(Diff)	*-0.20*	*-0.14*	*-0.10*	*-0.22*	*-0.10*
	Roberta_gemma (No attack)	0.94	0.94	0.94	0.94	0.94
	Roberta_gemma (With attack)	0.84	**0.60**	0.86	0.66	0.74
	(Diff)	*-0.10*	*-0.34*	*-0.08*	*-0.28*	*-0.20*
	Roberta_mistral (No attack)	0.92	0.92	0.92	**0.92**	0.92
	Roberta_mistral (With attack)	0.72	0.59	0.78	**0.54**	0.80
	(Diff)	*-0.20*	*-0.33*	*-0.14*	*-0.38*	*-0.12*
	Roberta_RR (No attack)	-	-	-	**-**	-
	Roberta_RR (With attack)	-	-	-	**-**	-
	(Diff)	*-0.03*	*+0.02*	*+0.07*	*+0.05*	*+0.06*
	OpenAI(No attack)	-	-	-	**-**	-
	OpenAI(With attack)	-	**-**	-	-	-
	(Diff)	*-0.04*	*-0.16*	*+0.03*	*-0.10*	*+0.05*
	Fast-DetectGPT (No attack)	0.92	0.92	0.92	**0.92**	0.92
	Fast-DetectGPT (With attack)	-	**-**	-	-	-
	(Diff)	*-0.07*	*-0.22*	*-0.04*	*-0.11*	*-0.19*

Generator	**Detector**	**Attack**				
Avg.		High temperature	Repetition penalty	News prompt	Tweet prompt	Paraphrasing
Gemma chat	Roberta_Phi-2 (No attack)	0.97±0.00	0.97±0.00	0.97±0.00	0.97±0.00	0.97±0.00
	Roberta_Phi-2 (With attack)	0.91±0.03	0.89±0.03	0.94±0.02	**0.63±0.05**	0.90±0.03
	(Diff)	*-0.06*	*-0.08*	*-0.03*	*-0.34*	*-0.07*
	Roberta_gemma (No attack)	0.94±0.00	0.94±0.00	0.94±0.00	0.94±0.00	0.94±0.00
	Roberta_gemma (With attack)	0.89±0.03	0.88±0.03	0.90±0.03	**0.83±0.04**	0.84±0.04
	(Diff)	*-0.05*	*-0.06*	*-0.04*	*-0.11*	*-0.10*
	Roberta_mistral (No attack)	0.93±0.00	0.93±0.00	0.93±0.00	0.93±0.00	0.93±0.00
	Roberta_mistral (With attack)	0.88±0.03	0.79±0.04	0.88±0.03	**0.67±0.05**	0.86±0.04
	(Diff)	*-0.05*	*-0.14*	*-0.05*	*-0.26*	*-0.07*
	Roberta_RR (No attack)	0.90±0.00	0.90±0.00	0.90±0.00	0.90±0.00	0.90±0.00
	Roberta_RR(With attack)	0.95±0.02	0.98±0.01	0.92±0.03	**0.98±0.01**	0.96±0.02
	(Diff)	*+0.05*	*+0.08*	*+0.02*	*+0.08*	*+0.06*
	OpenAI(No attack)	0.82±0.01	0.82±0.01	0.82±0.01	0.82±0.01	0.82±0.01
	OpenAI(With attack)	0.76±0.04	0.62±0.05	0.82±0.04	**0.63±0.05**	0.87±0.03
	(Diff)	*-0.06*	*-0.20*	*-0.00*	*-0.19*	*+0.05*
	Fast-DetectGPT (No attack)	0.95±0.00	0.95±0.00	0.95±0.00	0.95±0.00	0.95±0.00
	Fast-DetectGPT (With attack)	0.87±0.03	0.79±0.04	0.93±0.03	0.84±0.04	**0.76±0.04**
	(Diff)	*-0.08*	*-0.16*	*-0.02*	*-0.11*	*-0.19*
Zephyr	Roberta_Phi-2 (No attack)	0.96	0.96	0.96	0.96	0.96
	Roberta_Phi-2 (With attack)	0.84	0.88	0.90	**0.84**	0.90
	(Diff)	*-0.12*	*-0.08*	*-0.06*	*-0.12*	*-0.06*
	Roberta_gemma (No attack)	0.94	0.94	0.94	0.94	0.94
	Roberta_gemma (With attack)	0.89	0.77	0.90	**0.79**	0.84
	(Diff)	*-0.05*	*-0.17*	*-0.04*	*-0.15*	*-0.10*
	Roberta_mistral (No attack)	0.93	0.93	0.93	**0.93**	0.93
	Roberta_mistral (With attack)	0.82	0.74	0.85	**0.73**	0.86
	(Diff)	*-0.11*	*-0.19*	*-0.08*	*-0.20*	*-0.07*
	Roberta_RR (No attack)	0.88	0.88	0.8	**0.88**	0.88
	Roberta_RR(With attack)	0.85	0.90	0.87	**0.93**	0.96
	(Diff)	*-0.03*	*+0.02*	*+0.07*	*+0.05*	*+0.06*
	OpenAI(No attack)	0.70	0.70	0.70	**0.70**	0.70
	OpenAI(With attack)	0.66	0.54	0.73	**0.60**	0.87
	(Diff)	*-0.04*	*-0.16*	*+0.03*	*-0.10*	*+0.05*
	Fast-DetectGPT (No attack)	0.94	0.94	0.94	**0.94**	0.94
	Fast-DetectGPT (With attack)	0.87	0.72	0.90	**0.83**	0.76
	(Diff)	*-0.07*	*-0.22*	*-0.04*	*-0.11*	*-0.19*