Sequential Watermarking of Language Models

D. Juyal, E. Cantu-Cervini, Z. Jin, S. L. Adru

1 Introduction

Traditional watermarking methods, which relied on clear distinctions between machine-generated and human-written content, are becoming less effective as LLMs improve[9][3]. As watermarking methods improve, more advanced detection techniques are also emerging[1][5][6]. Recent watermarking techniques have explored statistical embedding and detection strategies. Kuditipudi[8] proposed a distortion-free watermarking approach with a reliable detection method, but its reliance on batch processing limits real-time applicability. This project addresses this limitation by developing a sequential watermark detection algorithm[2]. We propose an anytime valid e-process/p-process framework that enables real-time watermark detection with early stopping, on-the-fly access to statistical values, and reduced computational complexity.

In this report, we examine the mathematical framework proposed by Kuditipudi[8], evaluate its validity in supporting their claims, and demonstrate the theoretical soundness of our proposed substitution of the permutation test with a sequential Monte Carlo test, inspired by work of Fischer[4]. Furthermore, we validate our findings through simulations, exhibiting preliminary results of reduction in computations, in this case, reduction in number of permutations and a few instances of early stopping of the test.

2 Methodology

An ideal watermark satisfies three key characteristics: it is agnostic, distortion-free, and robust. The watermarking algorithm proposed by Kuditipudi[8], is specifically designed to achieve all three of these attributes. Mathematically, we break down the process into two components: a *generate* method that deterministically maps a sequence of random keys, denoted as ξ, encoded by the watermark key to a sample from the language model, and a detect method that aligns a given watermarked text with the watermark key sequence using the shared key.

Now we discuss the test setup, we start dicussing our algorithm's working and it's theoretical analysis. Let V be the vocabulary, and let $p \in V^* \to \Delta(V)$ be an autoregressive language model which maps a string of arbitrary length to a distribution over the vocabulary, with $p(\cdot \mid x)$ denoting the distribution of the next token given the prefix $x \in V^*$. Let Ξ denote the space in which the elements of the watermark key sequence lie.

1. The LM provider shares a random watermark key sequence $\xi \in \Xi^*$ with the detector;
2. The user sends a prompt $x \in V^*$ to the LM provider;
3. The LM provider generates text $Y \in V^*$ by $Y = \text{generate}(x, \xi)$;
4. The user publishes text $Y_e \in V^*$, which may be either:
   (a) (an edited version of) the generated text Y, or
   (b) text independent of Y (e.g., text that they wrote themselves);
5. The detector determines if Y_e is watermarked—i.e., if Y_e depends on the watermark key sequence—by computing a p-values $p = \text{detect}(Y, \xi_e)$ with respect to the null hypothesis that Y_e is independent (exchangeable) of ξ (i.e., not watermarked).

To simplify our discussion, we break our work into four algorithms as follows:

2.1 Watermarked Text Generation

The generate method generates a watermarked text sequence by incorporating a watermark key sequence into the decoding process of a language model. Given an input watermark key sequence $\xi \in \Xi^n$, a language model p, and a decoder Γ, the algorithm constructs an output string y of length m. At each step i, the decoder Γ generates the next token y_i based on the watermark key $\xi_{i\%n}$ and the conditional probability distribution of the language model given the previously generated tokens $y_{:i-1}$. This introduces a structured perturbation into the generated text, embedding the watermark information in a way that aligns with the probabilistic nature of the language model.

2.2 Watermarked Text Detection

The detect method detects watermarked text by computing a p-value through a statistical test. Given an input string y and a watermark key sequence ξ, the algorithm leverages a test statistic Φ to measure the relationship between y and ξ. It first samples T new key sequences $\xi^{(t)}$ from the distribution ν and computes the corresponding test statistic values $_t$. The p-value p is then estimated by the sequential monte carlo test. A lower p-value suggests stronger evidence that y contains the watermark, while a higher p-value indicates that the observed statistic could have arisen by chance.

2.3 Sequential Monte Carlo Test

SMC test implements a sequential Monte Carlo test to replace the traditional permutation test. Given a sequence of test statistics Φ_t, the method dynamically tracks a wealth function W_t^{uc} based on a binomial mixture strategy, as in [4][10]. The statistic L_t counts the number of times Y_t exceeds the initial threshold Φ_0, and the wealth function is updated accordingly. The test stops at time τ when W_t^{uc} either drops below α or exceeds $1/\alpha$. The final p-value Q_t is derived from the supremum of W_s up to time t, ensuring a valid sequential hypothesis testing procedure.

Algorithm 1 Sequential Monte Carlo Test

```
1: **Input:** Significance level $\alpha \in (0,1)$, parameter $c < \alpha$, sequence of test statistics $\Phi_0, \Phi_1, \Phi_2, \dots$

2: **Output:** Stopping time $\tau$, wealth sequence $\{W_t^{uc}\}_{t=1}^{\tau}$
3: Initialize $W_0^{uc} \leftarrow 1$, $L_0 \leftarrow 0$
4: for $t = 1, 2, \dots$ do
5:   if $\Phi_t \geq \Phi_0$ then
6:     $L_t \leftarrow L_{t-1} + 1$
7:   else
8:     $L_t \leftarrow L_{t-1}$
9:   end if
10:  $W_t^{uc} \leftarrow \frac{1-\mathrm{Bin}(L_t; t+1, c)}{c}$
11:  if $W_t^{uc} < \alpha$ or $W_t^{uc} \geq \frac{1}{\alpha}$ then
12:    $\tau \leftarrow t$
13:    return $\tau, \{W_t^{uc}\}_{t=1}^{\tau}$
14:  end if
15: end for
```

2.4 Test Statistic

The algorithm calculates a test statistic $\Phi(y, \xi)$ that measures the alignment between the input string y and the watermark key sequence ξ. It works by sliding blocks of size k over both the string y and the watermark sequence ξ. For each position i in the string y, the algorithm extracts a block of size k from both y and ξ. The alignment cost $d(y_i, \xi_j)$ is then computed for each block pair, and the algorithm returns the minimum alignment cost as the test statistic. A smaller value of $\Phi(y, \xi)$ indicates a better match between y and ξ, suggesting a stronger presence of the watermark in the text.

3 Results and Discussion

To evaluate whether the Sequential Monte Carlo (SMC) test outperforms the traditional permutation test and the fast permutation test, we conduct experiments across various generation lengths m. For each value of m, we sample $T = 5000$ prompts and generate outputs using different sampling methods[7]. The results are summarized in **Figure 1** . For all sampling methods, as m increases, the median p-value of the watermarked text decreases, indicating that longer generation lengths significantly enhance the detection power of the test. Compared to the result of the same experiment but using fast permutation test, which does not decrease monotonically in some sampling method, the SMC test outperforms previous method.

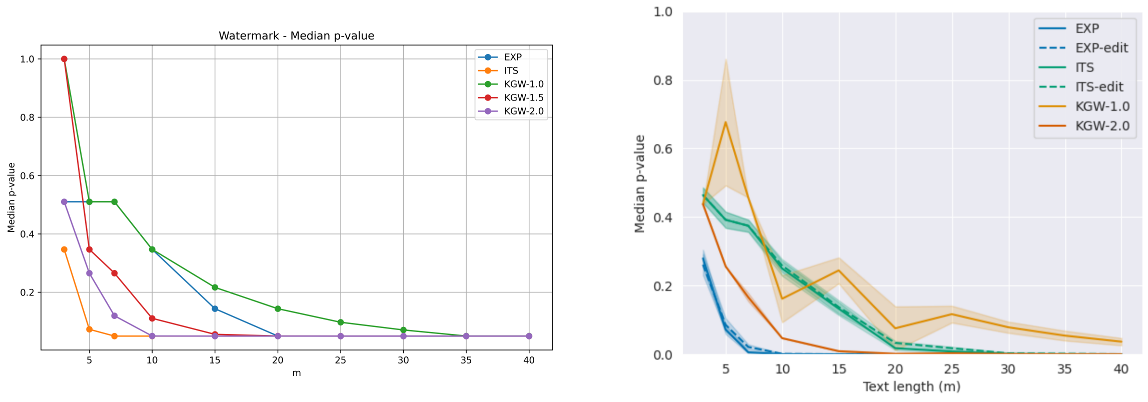

Figure 1: Median p-value of watermarked text relative to varying the text length m; Left: SMC test; Right: Fast permutation test[8]

We also aim to evaluate whether the SMC test can effectively detect watermarks. Taking the exponential minimum sampling method[8] as an example, the results shown in **Figure 2** demonstrate that the performance of the SMC test improves with increasing m when detecting watermarked outputs. For null outputs, the SMC test mislabels fewer than 5% of the samples, indicating low mistake rate. Overall, these results suggest that the SMC test is both sensitive and reliable in distinguishing watermarked from non-watermarked content.

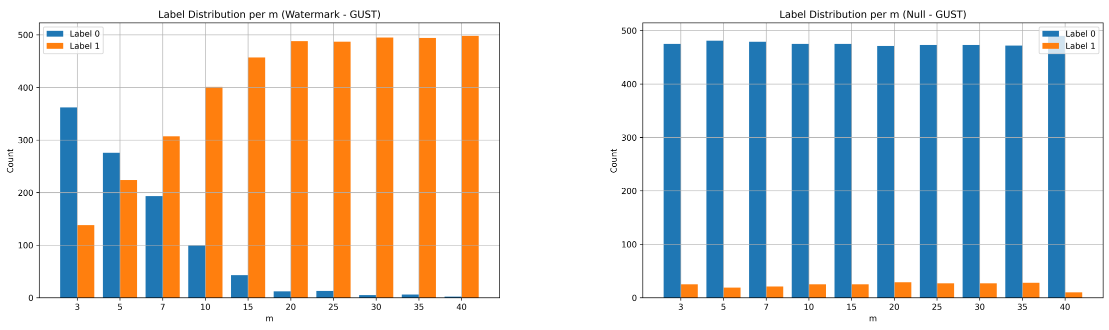

Figure 2: Result of detection. Label 0: Null; Label 1: Watermarked

4 Future Works

Moving forward, we will conduct a thorough comparison between our sequential approach and the traditional permutation approach using the oracle experiment as a benchmark. This experiment serves a critical purpose: determining the ideal number of permutation resamples needed for the standard permutation test to reach full statistical power (power = 1) when detecting watermarked text. To implement this, we will simulate numerous trials of watermarked text under the alternative hypothesis. For each trial, we will run the permutation test with various fixed values of T (number of resamples), recording whether the test correctly rejects the null hypothesis (p-value $< \alpha$) for each T. This process will allow us to identify the smallest T (the "oracle number") at which the empirical power (fraction of rejections) effectively reaches 1.

Once we establish this oracle number, we will evaluate our sequential permutation test to determine if it achieves similar detection power while using a comparable or smaller number of permutations. The sequential test stops accumulating permutation resamples as soon as the p-value drops below the significance level. By comparing the distribution of these stopping times against the oracle number, we can quantify the efficiency gains of our approach while ensuring we maintain detection accuracy. This oracle experiment is crucial because it provides a scientific basis for claiming our method is not only faster but also maintains the statistical guarantees of the traditional approach. It will help us demonstrate that our sequential test can gain "adaptivity" without sacrificing reliability.

We will also explore the resilience of our sequential test against diverse attacks, including translation-based attacks (translating watermarked text to other languages and back to English) and other adversarial transformations. Understanding how our watermarking method performs under these conditions is essential for practical deployment in real-world scenarios where text may undergo various transformations.

These research directions should provide a much clearer picture of how our algorithm performs in real-world applications, where both efficiency and robustness are non-negotiable requirements.

5 Conclusion

In this project, we have successfully developed a sequential watermarking approach for language models that addresses the limitations of traditional batch-based methods. By implementing a Sequential Monte Carlo test inspired by Fischer[4], we have created an anytime-valid statistical framework that enables real-time watermark detection with early stopping capabilities. Our experimental results demonstrate that as text length increases, the median p-value of watermarked text decreases consistently and exponentially, illustrating the effectiveness of our approach. This provides evidence that our sequential method offers comparable detection power to existing techniques while significantly reducing computational overhead.

Our future work will focus on a comprehensive comparison between our sequential approach and the oracle experiment, which utilizes fixed-number permutation tests as a benchmark. This oracle experiment will serve as a ground truth for evaluating the performance trade-offs in our method, particularly quantifying the computational efficiency gains achieved through sequential testing against any potential compromises in detection accuracy. Additionally, we plan to investigate the resilience of our watermarking technique against various adversarial scenarios, including translation-based attacks, to ensure its practical applicability in real-world settings where both efficiency and robustness are essential.

References

[1] Anton Bakhtin, Sam Gross, Myle Ott, Yuntian Deng, Marc'Aurelio Ranzato, and Arthur Szlam. Real or fake? learning to discriminate machine from human generated text. *arXiv preprint arXiv:1906.03351*, 2019.

[2] Can Chen and Jun-Kun Wang. Online detecting llm-generated texts via sequential hypothesis testing by betting. *arXiv preprint arXiv:2410.22318*, 2024.

[3] Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 1125–1139. PMLR, 2024.

[4] Lasse Fischer and Aaditya Ramdas. Sequential monte-carlo testing by betting. *arXiv preprint arXiv:2401.07365*, 2024.

[5] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[6] Steffen Herbold, Annette Hautli-Janisz, Ute Heuer, Zlata Kikteva, and Alexander Trautsch. A large-scale comparison of human-written versus chatgpt-generated essays. *Scientific reports*, 13(1):18617, 2023.

[7] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *International Conference on Machine Learning*, pages 17061–17084. PMLR, 2023.

[8] Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*, 2023.

[9] Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip Yu. A survey of text watermarking in the era of large language models. *ACM Computing Surveys*, 57(2):1–36, 2024.

[10] Shubhanshu Shekhar and Aaditya Ramdas. Nonparametric two-sample testing by betting. *IEEE Transactions on Information Theory*, 70(2):1178–1203, 2023.