TripPattern: A Pattern-based Text Watermarking Method for Large Language Models

Sangjun Moon[1], **Dasom Choi**[1], *__Jingun Kwon__*[1],
Hidetaka Kamigaito[2], **Taro Watanabe**[2], **and Manabu Okumura**[3]
[1]Chungnam National University, [2]Nara Institute of Science and Technology (NAIST)
[3]Institute of Science Tokyo
{sangjunmoon, dasomchoi}@o.cnu.ac.kr
jingun.kwon@cnu.ac.kr
{kamigaito.h, taro}@is.naist.jp
oku@pi.titech.ac.jp

Abstract

Text watermarking techniques have gained significant attention for identifying machine-generated text and mitigating risks from large language models (LLMs). Existing methods typically divide an LLM's vocabulary into green and red tokens, but encouraging generation toward green tokens can reduce text quality and naturalness. To address this, we propose **TripPattern**, a watermarking framework that formulates text watermarking as a pattern-based matching task using three vocabulary partitions. TripPattern divides the vocabulary into one neutral group and two pattern groups. During generation, the model alternates token selection between the two pattern groups to embed detectable patterns, while neutral tokens are selected independently to improve flexibility and preserve naturalness. For detection, TripPattern uses pattern-based statistical tests that provide interpretable p-values by measuring how often adjacent tokens alternate between the pattern groups. Theoretical analysis and empirical evaluations on four multilingual datasets show that TripPattern maintains LLM generation quality while achieving robust watermark detectability. We release our code at: https://anonymized_for_review.

1 Introduction

Large language models (LLMs) have demonstrated strong generative capabilities across diverse domains (Touvron et al., 2023; OpenAI, 2024; Grattafiori et al., 2024; Qwen Team, 2025), but their widespread use has raised concerns about misinformation (Brown et al., 2020; Chen et al., 2023), academic dishonesty (Stokel-Walker, 2022; Vasilatos et al., 2023), and the reliability of digital content (Tang et al., 2024; Li et al., 2024). These concerns highlight the need for robust watermarking methods that can reliably identify machine-generated text, especially as LLM-generated content may affect future models trained on artificial data (Radford et al., 2022; Shumailov et al., 2024).

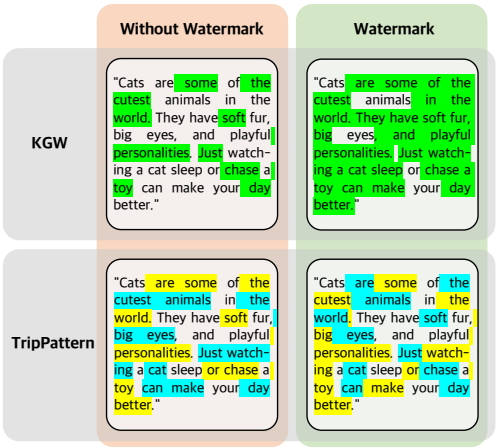


Figure 1: Unlike KGW's two-group partition, which encourages green token generation, TripPattern alternates between two pattern groups while allowing neutral-token sampling to preserve text quality.

Decoding-based watermarking methods embed identifiable signals directly into generated text and have received significant attention for detecting LLM-generated content (Sadasivan et al., 2025). Most existing approaches rely on the KGW-style two-group token partitioning framework, which divides the vocabulary into green and red tokens (Kirchenbauer et al., 2023; Guo et al., 2024; Ren et al., 2024; Chen et al., 2024; Zhao et al., 2024; Wu and Chandrasekaran, 2024). By biasing generation toward green tokens, these methods encode detectable fingerprints; however, this restriction reduces vocabulary flexibility and often degrades text quality and naturalness. This limitation is particularly problematic in multilingual and practical settings, where linguistic diversity and fluency are important.

Although several studies have attempted to reduce quality degradation through unbiased distribution adjustments (Kuditipudi et al., 2024; Hu et al., 2024), entropy-based gating (Lu et al., 2024; Liu and Bu, 2024; Lee et al., 2024), dynamic

Findings of the Association for Computational Linguistics: IJCNLP-AACL 2026, pages 3523–3538
November 6-10, 2026 ©2026 Association for Computational Linguistics

watermark-strength scaling (Wang et al., 2025), or auxiliary models (Liu et al., 2024; He et al., 2024; Huo et al., 2024), these methods often introduce trade-offs among text quality, detection robustness, computational cost, and deployment flexibility (Chen et al., 2024; Mao et al., 2025).

To address these limitations, we propose **TripPattern**, a watermarking approach that formulates watermark embedding as a pattern-based matching task within a three-group vocabulary partitioning framework. As shown in Figure 1, TripPattern divides the vocabulary into one neutral group and two distinct pattern groups. During generation, the model alternates token selection between the two pattern groups to embed statistically detectable patterns, while tokens from the neutral group can be sampled independently to preserve fluency and naturalness.

To construct the neutral group, we further consider subword token length and assign tokens shorter than a predefined threshold to this group. Since shorter tokens often occur more frequently due to linguistic economy (Kanwal et al., 2017), and tokenization algorithms tend to encode frequent morphemes or words as shorter subword units (Chung et al., 2020), this design helps maintain grammatical coherence and can be naturally applied to multilingual scenarios.

For detection, TripPattern performs a pattern-based statistical test using only tokens from the two pattern groups, ignoring neutral tokens that do not carry watermark signals. The detector measures how often adjacent pattern-group tokens alternate and uses this statistic to determine whether the text contains the embedded watermark.

Through theoretical analysis and experiments on four multilingual datasets, we show that TripPattern preserves the generative quality of LLMs while achieving robust watermark detectability.

2 Method

In this section, we first review the widely used two-group watermarking framework. We then present **TripPattern**, which employs a three-group vocabulary partition and constructs a neutral group based on subword token length.

2.1 Preliminary

Text watermarking for LLMs generally consists of two procedures: watermark encoding and detection. Given an input prompt $\mathbf{x}_{1:n} = \{x_1, x_2, ..., x_n\}$, an LLM $\mathcal{M}$ generates an output sequence $\{x_{n+1}, x_{n+2}, ..., x_m\}$, into which the encoding procedure embeds an identifiable watermark.

A representative approach is the KGW-based two-group partitioning framework (Kirchenbauer et al., 2023). At each generation step t, a keyed hash function computes

$$h_{n+t} = H_k(\mathbf{x}_{n+t-1}),$$

where H_k is a cryptographic hash or PRF keyed by a secret key k. The resulting hash value seeds a pseudorandom number generator (PRNG), which deterministically partitions the vocabulary $\mathcal{V}$ into two disjoint subsets: green tokens $\mathcal{V}_g$ and red tokens $\mathcal{V}_r$.

Let $l^{n+t} \in \mathbb{R}^{|\mathcal{V}|}$ denote the logits predicted by $\mathcal{M}$ at step t. The KGW framework adds a positive bias δ to green-token logits:

$$l_i^{n+t} = \begin{cases} l_i^{n+t} + \delta, & \text{if } v_i \in \mathcal{V}_g, \\ l_i^{n+t}, & \text{if } v_i \in \mathcal{V}_r, \end{cases} \tag{1}$$

where i denotes the token index in the vocabulary. As a result, watermarked text statistically contains more green tokens than ordinary human-written text.

For detection, the verifier recomputes the same partitioning process using the shared hash function and PRNG, and calculates the following z-score:

$$z = (G - \gamma L)/\sqrt{L\gamma(1-\gamma)}, \tag{2}$$

where G is the number of green tokens in the generated output, L is the total number of generated tokens, and γ is the green-token ratio. The watermark is identified by testing the null hypothesis H_0*: The generated text sequence does not follow the green-list constraint*. If the computed z-score exceeds a predefined threshold, H_0 is rejected.

Despite its effectiveness, the two-group framework inherently reduces token-selection flexibility by encouraging the model to select green tokens, which can degrade text quality and naturalness. Several studies have attempted to mitigate this issue through unbiased output-distribution adjustments, such as Gumbel softmax and output reweighting (Kuditipudi et al., 2024; Hu et al., 2024). However, preserving an unbiased distribution does not necessarily guarantee both text quality and detection performance, especially for aligned LLMs (Chen et al., 2024).

2.2 TripPattern: Three-group Partitioning

We propose **TripPattern**, a watermarking method that formulates watermark embedding as a pattern-based matching task. Unlike the conventional green/red partition, TripPattern divides the vocabulary into three disjoint subsets: one neutral group $\mathcal{V}_n$ and two pattern groups $\mathcal{V}_{p1}$ and $\mathcal{V}_{p2}$. The neutral group contains freely selectable tokens, while the two pattern groups are used to embed detectable alternation patterns.

Following prior watermarking methods (Kuditipudi et al., 2024), TripPattern uses a keyed hash and PRNG to deterministically partition the vocabulary at each generation step. The same procedure is used during both generation and detection, ensuring that the partition can be reconstructed by an authorized detector.

The encoding process encourages an alternating pattern between the two pattern groups. Specifically, if the last non-neutral token belongs to $\mathcal{V}_{p1}$, the next allowable set is biased toward $\mathcal{V}_{p2} \cup \mathcal{V}_n$; conversely, if the last non-neutral token belongs to $\mathcal{V}_{p2}$, the next allowable set is biased toward $\mathcal{V}_{p1} \cup \mathcal{V}_n$. Neutral tokens do not change the current alternation state, allowing the model to preserve fluency while maintaining the watermark pattern.

At each generation step, TripPattern applies a positive soft bias $\delta > 0$ to the logits of tokens in the allowable set $\mathcal{A}_t$ and then normalizes over the full vocabulary $\mathcal{V}$. This design statistically encourages the intended alternation pattern while retaining broader token-selection flexibility than two-group partitioning.

2.3 Subword Length-based Neutral Group Construction

We further construct the neutral group using subword token length at each decoding step. Given the step-specific hash value, the PRNG deterministically repartitions the vocabulary into $\mathcal{V}_n$, $\mathcal{V}_{p1}$, and $\mathcal{V}_{p2}$. During this per-step partitioning process, tokens shorter than a predefined threshold are prioritized for assignment to $\mathcal{V}_n$, and the remaining neutral slots are filled according to the PRNG until the target neutral-group ratio γ is reached. The remaining tokens are then divided into the two pattern groups.

Thus, the neutral group is not a fixed global vocabulary subset; rather, it is reconstructed at each generation and detection step using the same keyed hash/PRNG procedure, with subword length serving as a deterministic priority rule. This ensures that the detector can reproduce the same step-specific partitions while allowing frequent short subword tokens to more often function as neutral tokens.

2.4 Detecting the Watermark

TripPattern detects watermarks by testing whether a given text follows the alternation pattern induced during generation. Given a text sequence, the detector reconstructs the three-way partition using the same keyed hash and PRNG. It then removes tokens belonging to the neutral group and maps the remaining tokens to their corresponding pattern labels, such as "1" and "2".

The detector counts the number of adjacent pattern-token pairs and the number of alternations between $\mathcal{V}_{p1}$ and $\mathcal{V}_{p2}$. Let C denote the number of observed alternations and r the corresponding alternation ratio. The null hypothesis is defined as H_0*: The generated text sequence does not follow the alternation pattern*. The observed statistic is tested under the permutation null conditional on (n_1, n_2) using the runs moments (Wald and Wolfowitz, 1940). If the resulting z-score exceeds a predefined threshold, H_0 is rejected and the watermark is identified. The detailed generation and detection algorithms are described in Appendix A.

2.5 Theoretical Analysis

We provide a theoretical analysis showing that TripPattern offers greater token-selection flexibility than the KGW-based two-group framework.

Definition (Vocabulary Partition). *Let $\mathcal{V}$ denote the vocabulary of the language model. In the KGW-based framework, the vocabulary is partitioned into two disjoint subsets:*

$$\mathcal{V} = \mathcal{V}_g \cup \mathcal{V}_r, \qquad \mathcal{V}_g \cap \mathcal{V}_r = \emptyset,$$

where $\mathcal{V}_g$ and $\mathcal{V}_r$ denote the green and red token sets, respectively. In TripPattern, the vocabulary is partitioned into three disjoint subsets:

$$\mathcal{V} = \mathcal{V}_{p1} \cup \mathcal{V}_{p2} \cup \mathcal{V}_n,$$

where $\mathcal{V}_{p1}$ and $\mathcal{V}_{p2}$ are the two pattern groups and $\mathcal{V}_n$ is the neutral group.

Theorem 1. (Greater selection flexibility of TripPattern) *Let $\mathcal{A}_t^{KGW}$ and $\mathcal{A}_t^{Trip}$ denote the biased token sets at decoding step t under KGW and TripPattern, respectively. Let γ_g be the green-token*

ratio in KGW and γ_n be the neutral-group ratio in TripPattern. Then,

$$|\mathcal{A}_t^{KGW}| = \gamma_g|\mathcal{V}|, \qquad |\mathcal{A}_t^{Trip}| = \frac{1+\gamma_n}{2}|\mathcal{V}|.$$

Therefore,

$$|\mathcal{A}_t^{Trip}| \geq |\mathcal{A}_t^{KGW}| \quad \textit{if} \quad \frac{1+\gamma_n}{2} \geq \gamma_g,$$

with strict inequality when $\frac{1+\gamma_n}{2} > \gamma_g$. See Appendix B for the proof.

Thus, TripPattern provides a larger biased token set than KGW under this condition, suggesting greater token-selection flexibility. We empirically examine whether this flexibility translates into improved generation quality.

3 Experiments

3.1 Experimental Settings

Datasets. We evaluate TripPattern across four languages: English, Korean, German, and Spanish. For English fluency evaluation, we use the C4 dataset (Dodge et al., 2021), which is widely used in text watermarking studies. For multilingual fluency evaluation, we use mC4, a multilingual web-crawled corpus from Common Crawl (Xue et al., 2021). Following prior work, we randomly sample 500 instances from each test dataset (Kirchenbauer et al., 2023; Guo et al., 2024; Kuditipudi et al., 2024).

For mathematical reasoning evaluation, we use GSM8K (Cobbe et al., 2021), which contains 1,319 test instances of grade-school math problems. We follow the CoT Hub prompting protocol (Fu et al., 2023) and use few-shot prompts that require models to generate reasoning steps before producing final answers. For multilingual reasoning, we use MGSM (Shi et al., 2022), which contains 250 test instances per language. Following the MGSM setup, we use eight training instances as few-shot examples. Since Korean is not included in the original MGSM benchmark, we use a translated version of GSM8K.[1]

Evaluation Metrics. Following prior work (Ren et al., 2024; Kirchenbauer et al., 2023; Guo et al., 2024; Chen et al., 2024; Lu et al., 2024), we evaluate both generation quality and watermark detectability. For fluency evaluation, we report perplexity (PPL) on C4. For mathematical reasoning, we measure accuracy based on the correctness of generated answers. For watermark detection, we report the Area Under the Receiver Operating Characteristic curve (AUC) and true positive rate at a false positive rate of at most 5% (TPR@5).

Baselines. We compare **TripPattern** with several state-of-the-art watermarking baselines. **TripPattern w/o Len** is an ablated variant that removes subword length-based neutral group construction. **KGW** is a standard watermarking method that partitions the vocabulary into green and red token groups (Kirchenbauer et al., 2023). **Unigram** uses a fixed green list determined by a hash key to improve robustness (Zhao et al., 2024). **UPV** extends KGW with a cryptographically secure public verification scheme using an auxiliary detector (Liu et al., 2024). **SWEET** applies watermarking only in high-entropy contexts (Lee et al., 2024). We also evaluate **Morphmark**, which adaptively scales the watermark bias according to token-level green probability mass (Wang et al., 2025); its results are reported in Appendix C due to limited detection performance.

Models and Implementation Details. We conduct experiments using instruction-tuned models from the Llama-3 family (Grattafiori et al., 2024) with 3B and 8B parameters, and the Qwen-2.5 family (Qwen Team, 2025) with 3B and 7B parameters. For PPL evaluation, we use Qwen-2.5-14B-Instruct as the evaluator. Following prior work (Kirchenbauer et al., 2023; Chen et al., 2024), we use greedy decoding in all experiments to remove sampling randomness. We set the subword length threshold for the neutral group to three across all languages. For KGW-based methods, we set $\delta = 3.0$ and $\gamma = 0.3$; for pattern-based methods, we set $\delta = 5.0$ and $\gamma = 0.3$. The minimum and maximum generation lengths are set to 50 and 200 tokens, respectively. For KGW-based methods, γ denotes the green-token ratio, whereas for TripPattern, γ denotes the neutral-group ratio.

3.2 Main Results

TripPattern Preserves Text Quality Across Languages. Table 1 reports the results on C4 for English and mC4 for Korean, German, and Spanish. TripPattern generally maintains low perplexity across languages and models, showing limited degradation compared to non-watermarked generation. On average across the four languages, TripPattern achieves competitive or better text quality than the baseline watermarking methods.

[1] https://huggingface.co/datasets/ChuGyouk/GSM8k-Ko

Model	English			Korean			German			Spanish			Average		
	PPL ↓	AUC ↑	TPR@5 ↑	PPL ↓	AUC ↑	TPR@5 ↑	PPL ↓	AUC ↑	TPR@5 ↑	PPL ↓	AUC ↑	TPR@5 ↑	PPL ↓	AUC ↑	TPR@5 ↑
Llama-3.2-3B	3.49	-	-	3.86	-	-	3.53	-	-	3.14	-	-	3.51	-	-
+KGW	5.83	0.996	0.990	7.48	0.987	0.966	6.41	0.990	0.972	5.48	0.993	0.984	6.30	0.991	0.978
+Unigram	6.55	0.998	0.996	**4.86**	0.988	0.962	6.07	0.993	**0.984**	5.11	0.992	0.976	5.65	0.993	0.980
+UPV	5.36	**1.000**	0.998	5.70	0.960	0.858	**5.65**	0.993	0.978	4.95	0.986	0.972	**5.42**	0.985	0.952
+SWEET	5.25	0.996	0.990	7.07	0.989	0.966	5.68	0.990	0.970	5.07	0.993	0.982	5.77	0.992	0.977
+TripPattern	**5.24**	**1.000**	**1.000**	6.33	**0.995**	0.968	**5.65**	**0.996**	0.982	**4.53**	0.988	0.968	5.44	**0.995**	0.980
+TripPattern w/o Len	5.26	0.998	0.990	7.88	0.986	**0.974**	6.00	0.990	0.972	5.18	**0.996**	**0.988**	6.08	0.993	**0.981**
Llama-3.1-8B	3.08	-	-	3.06	-	-	3.07	-	-	2.66	-	-	2.97	-	-
+KGW	5.16	**1.000**	**1.000**	6.03	0.998	**0.996**	5.24	0.998	0.996	4.29	0.998	0.998	5.18	0.999	**0.998**
+Unigram	5.58	**1.000**	**1.000**	5.16	0.996	0.988	5.00	0.996	0.992	4.29	0.997	0.996	5.01	0.997	0.994
+UPV	4.66	0.999	0.994	**4.55**	0.962	0.926	4.72	0.994	0.984	3.95	0.997	0.984	4.47	0.988	0.972
+SWEET	**4.61**	0.999	0.998	5.69	0.998	0.988	4.78	0.998	0.996	3.94	**1.000**	**1.000**	4.76	0.999	0.996
+TripPattern	**4.61**	**1.000**	**1.000**	4.63	0.990	0.970	**4.45**	0.993	0.986	**3.63**	0.995	0.992	**4.33**	0.995	0.987
+TripPattern w/o Len	**4.61**	0.999	0.996	5.23	**0.999**	**0.996**	4.80	**1.000**	**0.999**	3.88	**1.000**	0.999	4.63	**1.000**	**0.998**
Qwen-2.5-3B	3.93	-	-	3.30	-	-	3.68	-	-	3.19	-	-	3.53	-	-
+KGW	8.96	0.992	0.982	7.22	0.968	0.930	8.08	0.966	0.932	6.13	0.963	0.924	7.60	0.972	0.942
+Unigram	8.03	0.990	0.964	6.55	0.955	0.892	6.45	0.951	0.878	5.32	0.927	0.808	6.59	0.956	0.886
+UPV	7.82	0.991	0.978	5.56	0.943	0.874	6.40	0.952	0.898	5.22	0.945	0.856	6.25	0.958	0.902
+SWEET	8.28	0.988	0.974	6.27	0.949	0.904	7.33	0.954	0.910	5.52	0.940	0.874	6.85	0.958	0.916
+TripPattern	7.05	**0.997**	**0.994**	**4.74**	**0.985**	0.956	**6.30**	0.986	0.964	**4.91**	**0.969**	0.924	**5.75**	**0.984**	0.960
+TripPattern w/o Len	**6.80**	0.994	0.984	6.17	0.984	**0.962**	7.04	**0.989**	**0.966**	5.37	**0.969**	**0.940**	6.35	**0.984**	**0.963**
Qwen-2.5-7B	3.24	-	-	3.17	-	-	3.49	-	-	2.90	-	-	3.20	-	-
+KGW	5.73	**1.000**	**1.000**	5.86	0.992	0.986	6.49	0.993	**0.992**	4.70	0.992	0.986	5.70	0.994	0.991
+Unigram	5.50	0.997	0.994	5.55	0.991	**0.990**	6.39	0.991	0.984	4.86	0.989	0.984	5.58	0.992	0.988
+UPV	5.25	0.999	0.994	4.66	0.980	0.948	5.39	0.982	0.970	4.33	0.987	0.982	4.91	0.987	0.974
+SWEET	5.27	0.996	0.994	5.32	0.995	0.988	5.90	0.992	0.982	**4.26**	0.991	0.978	5.19	0.993	0.986
+TripPattern	5.33	0.999	0.996	5.02	0.992	0.986	**5.35**	0.992	0.986	**4.26**	0.995	0.988	**4.80**	0.995	0.989
+TripPattern w/o Len	**5.07**	**1.000**	**1.000**	**4.24**	**0.997**	0.986	5.54	**0.997**	**0.992**	**4.26**	**0.996**	**0.996**	4.97	**0.998**	**0.994**

Table 1: Experimental results comparing TripPattern with baseline watermarking methods. TripPattern w/o Len is a method that does not consider subword length when defining the neutral group. Lower PPL scores indicate better text quality, whereas higher AUC and TPR@5 scores indicate better detection performance.

Model	English			Korean			German			Spanish			Average		
	ACC ↑	AUC ↑	TPR@5 ↑	ACC ↑	AUC ↑	TPR@5 ↑	ACC ↑	AUC ↑	TPR@5 ↑	ACC ↑	AUC ↑	TPR@5 ↑	ACC ↑	AUC ↑	TPR@5 ↑
Llama-3.2-3B	72.4	-	-	7.35	-	-	11.6	-	-	11.45	-	-	25.70	-	-
+KGW	64.06	0.963	0.814	6.29	0.982	0.900	9.78	0.942	0.700	11.30	0.926	0.692	22.86	0.953	0.777
+Unigram	61.64	0.938	0.724	6.44	0.960	0.828	9.70	0.949	0.780	10.46	0.906	0.648	22.06	0.938	0.745
+UPV	61.41	0.927	0.702	5.61	0.932	0.748	8.64	0.943	0.760	10.39	0.894	0.620	21.51	0.924	0.708
+SWEET	**67.75**	0.920	0.697	6.67	**0.989**	**0.952**	10.31	0.933	0.748	**11.52**	0.903	0.676	24.06	0.936	0.768
+TripPattern	67.10	0.950	0.776	**7.96**	0.965	0.848	**10.54**	**0.966**	**0.816**	11.45	0.932	0.648	**24.26**	0.953	0.772
+TripPattern w/o Len	64.22	**0.964**	**0.817**	5.46	0.977	0.892	9.78	0.946	0.784	9.78	**0.938**	**0.760**	22.31	**0.956**	**0.813**
Llama-3.1-8B	81.05	-	-	10.31	-	-	13.65	-	-	13.95	-	-	29.74	-	-
+KGW	68.76	0.960	0.826	8.79	**0.984**	**0.924**	12.13	0.972	0.796	11.98	0.948	0.748	25.42	**0.966**	0.824
+Unigram	68.99	0.940	0.734	7.81	0.956	0.820	11.14	0.972	0.844	11.14	0.915	0.636	24.77	0.946	0.759
+UPV	71.19	0.918	0.676	8.57	0.932	0.692	10.92	0.938	0.756	12.13	0.907	0.656	25.70	0.923	0.695
+SWEET	**76.72**	0.916	0.693	9.33	0.982	0.916	12.21	**0.978**	**0.900**	**12.81**	0.948	**0.796**	27.77	0.956	**0.826**
+TripPattern	75.74	0.944	0.740	**10.46**	0.951	0.776	**12.59**	0.973	0.892	12.51	0.945	0.760	**27.83**	0.953	0.792
+TripPattern w/o Len	69.90	**0.966**	**0.835**	8.11	0.970	0.840	10.92	0.959	0.848	11.83	**0.953**	0.744	25.19	0.962	0.817
Qwen-2.5-3B	33.06	-	-	9.25	-	-	6.52	-	-	7.28	-	-	14.03	-	-
+KGW	32.37	0.909	0.649	6.29	0.977	0.864	6.14	0.965	0.852	7.58	0.951	0.832	13.10	0.951	0.799
+Unigram	29.11	0.908	0.611	6.90	0.956	0.720	6.14	0.982	0.916	6.97	0.964	0.848	12.28	0.953	0.774
+UPV	29.11	0.856	0.477	5.91	0.958	0.756	5.16	0.959	0.836	6.67	0.923	0.588	11.71	0.924	0.664
+SWEET	**34.57**	0.898	0.590	6.82	**0.994**	**0.972**	**6.44**	0.979	0.896	**8.19**	**0.985**	**0.932**	**14.01**	0.964	0.848
+TripPattern	31.54	0.948	**0.768**	**7.81**	0.980	0.904	5.46	**0.992**	**0.956**	7.43	0.982	**0.932**	13.06	**0.976**	**0.890**
+TripPattern w/o Len	30.93	**0.950**	0.755	6.67	0.988	0.928	6.29	0.971	0.880	6.82	0.977	0.868	12.68	0.972	0.858
Qwen-2.5-7B	55.27	-	-	7.51	-	-	6.29	-	-	6.37	-	-	18.86	-	-
+KGW	47.08	0.848	0.434	7.13	0.932	0.748	5.46	0.906	0.576	6.37	**0.893**	**0.612**	16.51	**0.895**	0.593
+Unigram	41.32	0.808	0.345	6.52	0.873	0.520	**7.20**	0.869	0.576	6.22	0.814	0.380	15.32	0.841	0.455
+UPV	47.23	0.822	0.368	6.52	0.828	0.388	6.29	0.822	0.468	6.07	0.803	0.316	16.53	0.818	0.385
+SWEET	50.95	0.754	0.321	**7.51**	**0.938**	**0.784**	5.53	0.897	**0.656**	**6.82**	0.876	0.568	**17.70**	0.866	0.582
+TripPattern	**52.24**	**0.870**	**0.503**	6.44	0.924	0.716	5.99	**0.911**	0.624	5.99	0.864	0.536	17.67	0.892	**0.595**
+TripPattern w/o Len	41.39	0.849	0.461	6.44	0.935	0.712	6.52	0.890	0.560	5.99	0.869	0.464	15.09	0.886	0.549

Table 2: Experimental results comparing TripPattern and baseline watermarking methods. Higher ACC scores indicate a greater proportion of correct solutions.

The improvement is particularly clear when using the subword length-based neutral group. Compared with TripPattern w/o Len, TripPattern generally improves average fluency by assigning frequent short subword tokens, such as functional words and morphemes, to the neutral group, although the effect varies across models and languages. These results suggest that introducing a neutral group enables pattern-based watermarking while reducing the quality loss typically caused by conventional two-group green/red partitioning.

TripPattern Maintains Performance on CoT-Prompted Reasoning Tasks. Table 2 presents the results on GSM8K and MGSM. TripPattern preserves mathematical reasoning performance across languages, with limited accuracy degradation compared to baseline watermarking methods. This indicates that TripPattern can embed detectable watermarks while maintaining the model's ability to generate coherent reasoning steps and final answers.

The benefit of subword length-based neutral group construction is also observed in reasoning tasks. Across open-ended generation and reasoning tasks, TripPattern generally benefits from subword

length-based neutral group construction, improving average fluency on mC4 and maintaining stronger reasoning accuracy on MGSM.

TripPattern Achieves Competitive Detection Performance. Although watermarking generally involves a trade-off between text quality and detectability, TripPattern achieves AUC scores comparable to KGW while often improving text quality, as shown in Table 1. KGW can still yield higher TPR@5 in some settings. Moreover, on GSM8K and MGSM, TripPattern achieves competitive detection performance while better preserving reasoning accuracy in several settings, as shown in Table 2.

TripPattern w/o Len slightly outperforms TripPattern in detection because the length-based neutral group assigns frequent functional tokens to the neutral group, which can dilute the watermark signal. Nevertheless, TripPattern provides a stronger overall balance between text quality and watermark detectability, demonstrating the effectiveness of pattern-based detection with a three-group vocabulary partition.

4 Analysis

Flexibility of TripPattern. To further analyze the token-selection flexibility of TripPattern, we evaluate a "hard" partitioning setting, which is known to reduce generation quality, particularly in low-entropy contexts (Kirchenbauer et al., 2023). Unlike the soft-bias setting in Algorithm 1, this setting restricts token selection to the designated subset. For KGW, generation is restricted to the green group, while for TripPattern, generation alternates between the target pattern group and the neutral group.

Figure 2 reports the results under full γ sweeps. Since γ has different meanings in the two frameworks, we vary it separately according to each definition: in KGW, γ denotes the green-token ratio, whereas in TripPattern, it denotes the neutral-group ratio. All other settings follow the main experiments.

TripPattern consistently outperforms both TripPattern w/o Len and KGW in text quality, demonstrating its greater token-selection flexibility and the benefit of subword length-based neutral group construction. While all methods achieve similar AUC scores, TripPattern shows a sharp performance drop at large γ values, where the neutral group becomes excessively large and weakens the watermark signal. We further compare TripPattern and KGW by varying the watermark strength δ on English C4. As shown in Figure 3, TripPattern preserves text quality more effectively than KGW across the δ sweep while maintaining reliable watermark detection.

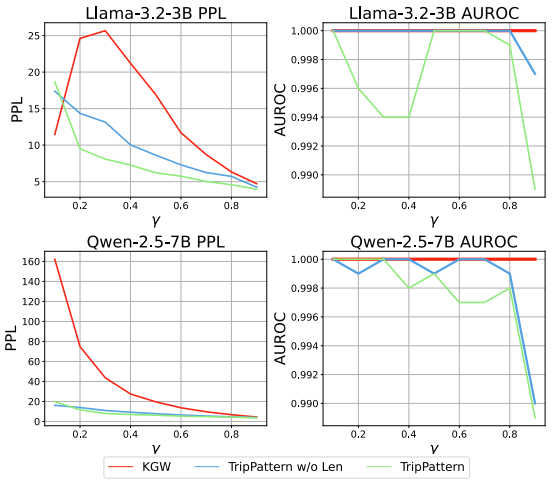


Figure 2: Comparison of TripPattern, TripPattern w/o Len, and KGW using hard group partitioning settings with different γ values on the English C4 dataset.

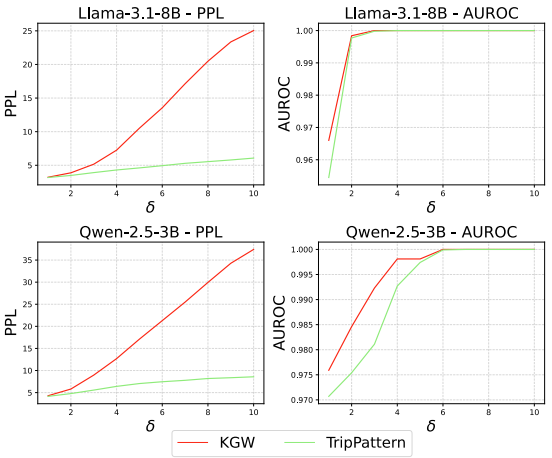


Figure 3: Comparison of TripPattern and KGW with different δ on the English C4 dataset.

Analysis of Subword Length. We examine whether the subword-length threshold used to construct the neutral group is consistent across models and languages. Figure 4 (a) shows that a threshold of three generally provides the best balance between text quality and watermark detectability. In particular, this setting consistently performs well for English, Korean, and German across models. Although increasing the neutral group can improve fluency by allowing more frequent short tokens to remain unconstrained, it may also weaken the watermark signal, revealing a trade-off between text

Method	CNN/DM				GovReport			
	R-L ↑	BS ↑	AUC ↑	TPR@5 ↑	R-L ↑	BS ↑	AUC ↑	TPR@5 ↑
Llama-3.1-8B	34.07	67.36	-	-	27.39	56.56	-	-
+KGW	**32.68**	66.55	0.997	0.980	12.37	46.23	0.999	0.992
+TripPattern	32.03	**66.56**	**0.999**	**0.996**	**30.54**	**61.08**	**1.000**	**1.000**
Qwen-2.5-3B	30.03	65.09	-	-	25.88	59.57	-	-
+KGW	26.77	63.50	**1.000**	**1.000**	20.28	56.78	**1.000**	**1.000**
+TripPattern	**26.92**	**63.55**	**1.000**	**1.000**	**23.68**	**58.94**	**1.000**	**1.000**

Table 3: Performance on the CNN/DM and GovReport datasets.

Model	Method	Info.	Conc.	Faith.
Qwen-2.5 -7B-Instruct	KGW	3.44	3.50	3.08
	TripPattern	**4.00***	**3.88***	**3.58***

Table 4: Human evaluation results.

Method	Original	WordDel			WordSub			Dipper
		0.1	0.2	0.3	0.1	0.2	0.3	
KGW	0.992	0.992	0.993	**0.991**	0.992	0.992	0.988	0.893
Unigram	0.990	0.987	0.986	0.985	0.986	0.984	0.981	**0.958**
UPV	0.991	0.987	0.983	0.980	0.991	0.991	**0.991**	0.873
SWEET	0.988	0.988	0.987	0.989	0.989	0.986	0.988	0.911
TripPattern	**0.997**	**0.997**	**0.996**	0.976	**0.997**	**0.995**	0.951	0.675
TripPattern w/o Len	0.994	0.992	0.988	0.966	0.992	0.983	0.934	0.641

Table 5: AUC under attacks on C4.

Model	English	Korean	German	Spanish
Llama-3.2-3B	0.67	0.75	0.73	0.72
Qwen-2.5-3B	0.54	0.66	0.65	0.67

Table 6: Average softmax probabilities on the mC4 datasets.

quality and detection performance.

Performance Trade-offs at Different δ. We further analyze the effect of the watermark strength δ, which controls the bias applied during token selection (Chen et al., 2024). As shown in Figure 4 (b), larger δ values improve watermark detectability but can degrade text quality by imposing stronger constraints on generation. This confirms the expected trade-off between fluency and detection performance.

Effect of Decoding Strategy. We also evaluate whether different decoding strategies affect watermark embedding and detection. Figure 4 (c) shows that detection performance remains largely stable across decoding strategies, indicating that TripPattern is robust to decoding choices.

Performance on Text Quality. To evaluate text quality beyond PPL, we conducted experiments on CNN/DM (Nallapati et al., 2016) and GovReport (Huang et al., 2021) for long-document summarization. We used ROUGE-L (R-L) (Lin, 2004) and BERTScore (BS) (Zhang et al., 2020) as automatic metrics. We also conducted human evaluation. We sampled 50 instances from CNN/DM (Zhou et al., 2020; Wang et al., 2023), and using MTurk, 50 evaluators with U.S. high school and bachelor's degrees rated outputs from 1 to 5 (5 is the best) for informativeness (Info.), conciseness (Conc.), and factual consistency (Faith.). Table 3 and 4 show the results. * denotes statistically significant (p<0.01) improvements, compared to the underlined scores. We used paired bootstrap resampling with 100,000 random samples (Koehn, 2004) for the significance test. Our TripPattern generates informative, concise, and faithful summaries while enabling watermark detection.

Robustness against Attacks. We evaluate robustness against three attack types: WordDel, WordSub, and Dipper. WordDel randomly removes a given percentage of words, while WordSub replaces a given percentage of words with WordNet synonyms. Dipper performs context-aware paraphrasing by controlling lexical diversity and content reordering (Krishna et al., 2023).

Table 5 reports the results using texts generated by Qwen-2.5-3B. TripPattern maintains reasonable detection performance under word-level perturbations, although it slightly underperforms KGW under 30% WordDel and WordSub attacks. In contrast, TripPattern is more vulnerable to Dipper, likely because context-aware paraphrasing disrupts the local alternation patterns used for detection.

Investigating Softmax Probability. To better understand the AUC degradation observed for both KGW and TripPattern on multilingual datasets in Table 1, we measure the average top-1 next-token softmax probability without watermarking. As shown in Table 6, multilingual datasets exhibit substantially higher next-token certainty than English. This suggests that the model distribution is more concentrated in multilingual settings, making generation less responsive to the watermarking bias δ. Consequently, watermark insertion becomes weaker, leading to reduced detection performance.

We provide additional case studies of TripPattern outputs in Appendix D.

5 Related Work

Due to recent advances in LLMs, which can generate fluent human-level text, early post-hoc AI-generated text detection methods formulated as binary classification have become less effective (Jawahar et al., 2020; Mitchell et al., 2023). Another line of work injects watermarks by modifying textual content or structure through rule-based

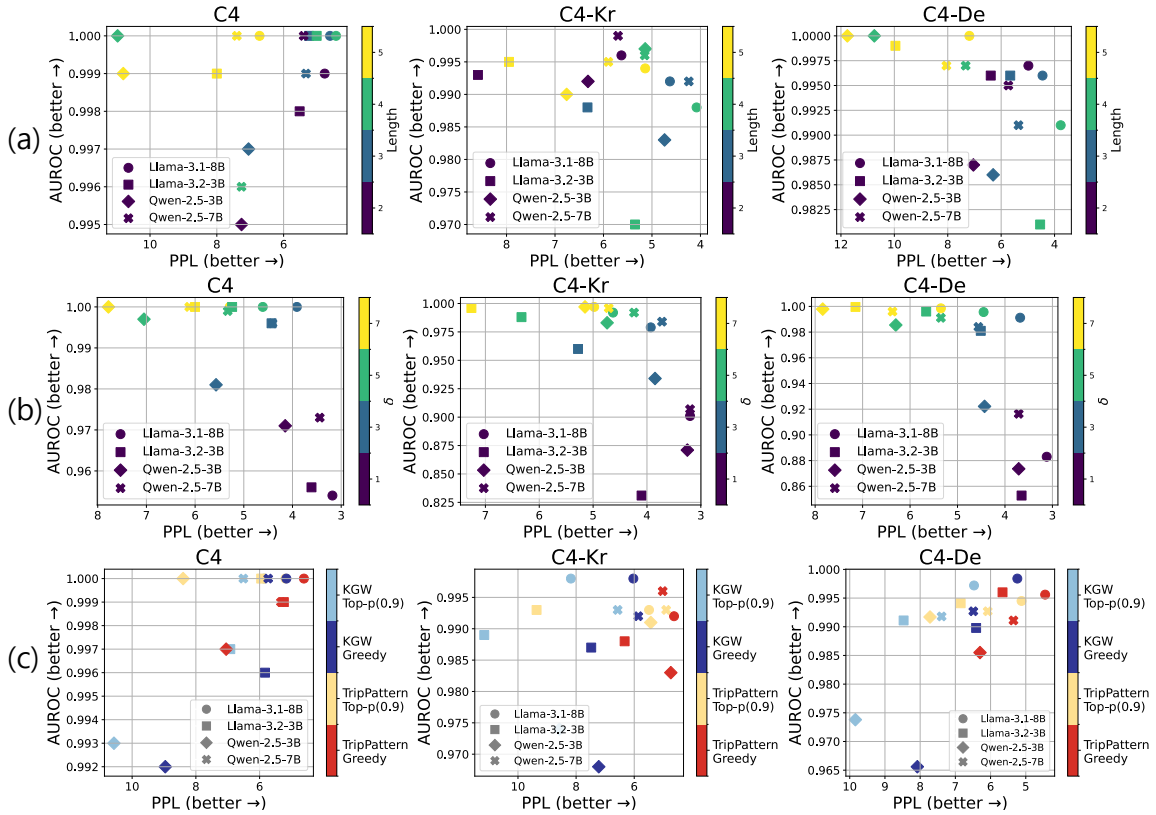


Figure 4: Analysis of key TripPattern design choices. (a) Effect of varying the subword-length threshold for neutral group construction. (b) Effect of varying the watermark strength δ on text quality (PPL) and detection performance (AUC). (c) Effect of different decoding strategies on text quality and detection performance.

methods (Katzenbeisser and Petitcolas, 2000) or neural encoders (Yang et al., 2021; Ueoka et al., 2021). These methods embed messages that can later be extracted using corresponding rules or neural decoders, but substantial modifications often degrade text quality due to the discrete nature of natural language.

Recent watermarking methods mainly operate at the decoding stage by adjusting token logits during generation without modifying the underlying LLM. The KGW framework introduced green/red vocabulary partitioning and green-token biasing (Kirchenbauer et al., 2023), which has since been extended through context-aware partitioning, robust detection rules, mutual exclusivity constraints, fixed green lists, and unbiased output-distribution adjustments (Guo et al., 2024; Ren et al., 2024; Wu and Chandrasekaran, 2024; Chen et al., 2024; Zhao et al., 2024; Kuditipudi et al., 2024; Hu et al., 2024).

Despite these advances, existing work has paid limited attention to vocabulary-partitioning watermarking in multilingual settings. To address this gap, we propose a three-group vocabulary partitioning method with two pattern groups and one neutral group. We further incorporate subword token length into neutral group construction, improving text quality and making the watermarking scheme more suitable for multilingual scenarios.

6 Conclusion

This study introduced TripPattern, a novel watermarking approach for LLMs that leverages a three-group vocabulary partitioning framework. By explicitly alternating token selection between two distinct pattern groups and incorporating a neutral group, TripPattern significantly enhances the token selection flexibility, preserving text quality and naturalness across multilingual datasets. We further propose considering subword token length as a criterion to define the neutral group. Because little attention has been paid to embedding watermarks in multilingual settings, we evaluate four languages. Our analysis and experimental results show that TripPattern provides a favorable balance between text quality and watermark detectability compared with conventional two-group partitioning methods in multilingual settings.

Limitations

Although TripPattern improves the balance between text quality and watermark detectability, it has several limitations. First, TripPattern relies on local alternation patterns between two pattern groups. As shown in our robustness analysis, this makes the method vulnerable to strong context-aware paraphrasing attacks such as Dipper, which can disrupt local token-level patterns while preserving the original meaning. Improving robustness against semantic paraphrasing remains an important direction for future work.

Second, TripPattern still involves a trade-off between text quality and detection performance. The watermark strength δ and the neutral-group ratio γ directly affect this balance. Larger δ values improve detectability but may reduce generation quality, while larger neutral groups provide greater token-selection flexibility but can dilute the watermark signal. Although we use fixed hyperparameters across languages, optimal settings may vary depending on the model, language, domain, and decoding strategy.

Third, detection performance can degrade in multilingual settings where the model distribution is highly concentrated. Our softmax probability analysis suggests that higher next-token certainty makes generation less responsive to watermarking bias, weakening watermark insertion. This issue may be more pronounced for low-resource languages, domain-specific text, or models with different tokenization behavior.

Finally, our experiments focus on instruction-tuned Llama and Qwen models and evaluate a limited set of languages and tasks. While TripPattern shows strong results across multilingual generation, reasoning, and summarization settings, broader evaluation on more languages, model families, domains, and real-world deployment scenarios is needed to fully assess its generalizability.

Acknowledgments

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (RS-2026-25569031, Development of an AIBOM-Based Platform for Strengthening the Reliability and Security of the AI Supply Chain). This study was financially supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (RS-2026-25580044). This work was supported by the Korea Basic Science Institute (KBSI) Young Researcher Infrastructure Support Program funded by the Ministry of Science and ICT (MSIT) (Grant No. RS-2026-25473607). Following are results of a study on the "Convergence and Open Sharing System" Project, supported by the Ministry of Education and National Research Foundation of Korea.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS)*.

Liang Chen, Yatao Bian, Yang Deng, Deng Cai, Shuaiyi Li, Peilin Zhao, and Kam-Fai Wong. 2024. WatME: Towards lossless watermarking through lexical redundancy. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9166–9180, Bangkok, Thailand. Association for Computational Linguistics.

Liang Chen, Yang Deng, Yatao Bian, Zeyu Qin, Bingzhe Wu, Tat-Seng Chua, and Kam-Fai Wong. 2023. Beyond factuality: A comprehensive evaluation of large language models as knowledge generators. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6325–6341, Singapore. Association for Computational Linguistics.

Hyung Won Chung, Dan Garrette, Kiat Chuan Tan, and Jason Riesa. 2020. Improving multilingual models with language-clustered vocabularies. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4536–4546, Online. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1286–1305, Online and

Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yao Fu, Litu Ou, Mingyu Chen, Yuhao Wan, Hao Peng, and Tushar Khot. 2023. Chain-of-thought hub: A continuous effort to measure large language models' reasoning performance. *Preprint*, arXiv:2305.17306.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Yuxuan Guo, Zhiliang Tian, Yiping Song, Tianlun Liu, Liang Ding, and Dongsheng Li. 2024. Context-aware watermark with semantic balanced green-red lists for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22633–22646, Miami, Florida, USA. Association for Computational Linguistics.

Zhiwei He, Binglin Zhou, Hongkun Hao, Aiwei Liu, Xing Wang, Zhaopeng Tu, Zhuosheng Zhang, and Rui Wang. 2024. Can watermarks survive translation? on the cross-lingual consistency of text watermark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4115–4129, Bangkok, Thailand. Association for Computational Linguistics.

Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. 2024. Unbiased watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1419–1436, Online. Association for Computational Linguistics.

Mingjia Huo, Sai Ashish Somayajula, Youwei Liang, Ruisi Zhang, Farinaz Koushanfar, and Pengtao Xie. 2024. Token-specific watermarking with enhanced detectability and semantic coherence for large language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Ganesh Jawahar, Muhammad Abdul-Mageed, and Laks Lakshmanan, V.S. 2020. Automatic detection of machine generated text: A critical survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2296–2309, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Jasmeen Kanwal, Kenny Smith, Jennifer Culbertson, and Simon Kirby. 2017. Zipf's law of abbreviation and the principle of least effort: Language users optimise a miniature lexicon for efficient communication. *Cognition*, 165:45–52.

Stefan Katzenbeisser and Fabien A. Petitcolas. 2000. *Information Hiding Techniques for Steganography and Digital Watermarking*, 1st edition. Artech House, Inc., USA.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Preprint*, arXiv:2303.13408.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2024. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*.

Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. 2024. Who wrote this code? watermarking for code generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4890–4911, Bangkok, Thailand. Association for Computational Linguistics.

Junyi Li, Jie Chen, Ruiyang Ren, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2024. The dawn after the dark: An empirical study on factuality hallucination in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10879–10899, Bangkok, Thailand. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. 2024. An unforgeable publicly verifiable watermark for large language models. In *International Conference on Learning Representations (ICLR)*.

Yepeng Liu and Yuheng Bu. 2024. Adaptive text watermark for large language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Yijian Lu, Aiwei Liu, Dianzhi Yu, Jingjing Li, and Irwin King. 2024. An entropy-based text watermarking detection method. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11724–11735, Bangkok, Thailand. Association for Computational Linguistics.

Minjia Mao, Dongjun Wei, Zeyu Chen, Xiao Fang, and Michael Chau. 2025. Watermarking large language models: An unbiased and low-risk method. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7939–7960, Vienna, Austria. Association for Computational Linguistics.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. DetectGPT: Zero-shot machine-generated text detection using probability curvature. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, pages 280–290.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Qwen Team. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision. *Preprint*, arXiv:2212.04356.

Jie Ren, Han Xu, Yiding Liu, Yingqian Cui, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. 2024. A robust semantics-based watermark for large language model against paraphrasing. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 613–625, Mexico City, Mexico. Association for Computational Linguistics.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2025. Can ai-generated text be reliably detected? *Preprint*, arXiv:2303.11156.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2022. Language models are multilingual chain-of-thought reasoners. *Preprint*, arXiv:2210.03057.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. 2024. The curse of recursion: Training on generated data makes models forget. *Preprint*, arXiv:2305.17493.

Chris Stokel-Walker. 2022. AI bot ChatGPT writes smart essays — should professors worry? *Nature*.

Liyan Tang, Igor Shalyminov, Amy Wong, Jon Burnsky, Jake Vincent, Yu'an Yang, Siffi Singh, Song Feng, Hwanjun Song, Hang Su, Lijia Sun, Yi Zhang, Saab Mansour, and Kathleen McKeown. 2024. TofuEval: Evaluating hallucinations of LLMs on topic-focused dialogue summarization. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4455–4480, Mexico City, Mexico. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and efficient foundation language models. In *arXiv preprint arXiv:2302.13971*.

Honai Ueoka, Yugo Murawaki, and Sadao Kurohashi. 2021. Frustratingly easy edit-based linguistic steganography with a masked language model. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5486–5492, Online. Association for Computational Linguistics.

Christoforos Vasilatos, Manaar Alam, Talal Rahwan, Yasir Zaki, and Michail Maniatakos. 2023. Howkgpt: Investigating the detection of chatgpt-generated university student homework through context-aware perplexity analysis. *Preprint*, arXiv:2305.18226.

Abraham Wald and Jacob Wolfowitz. 1940. On a test whether two samples are from the same population. *Annals of Mathematical Statistics*, 11(2):147–162.

Yiming Wang, Zhuosheng Zhang, and Rui Wang. 2023. Element-aware summarization with large language models: Expert-aligned evaluation and chain-of-thought method. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8640–8665, Toronto, Canada. Association for Computational Linguistics.

Zongqi Wang, Tianle Gu, Baoyuan Wu, and Yujiu Yang. 2025. MorphMark: Flexible adaptive watermarking for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Qilong Wu and Varun Chandrasekaran. 2024. Bypassing LLM watermarks with color-aware substitutions. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8549–8581, Bangkok, Thailand. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and

Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Xi Yang, Jie Zhang, Kejiang Chen, Weiming Zhang, Zehua Ma, Feng Wang, and Nenghai Yu. 2021. Tracing text provenance via context-aware lexical substitution. *Preprint*, arXiv:2112.07873.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang. 2024. Provable robust watermarking for AI-generated text. In *The Twelfth International Conference on Learning Representations*.

Qingyu Zhou, Furu Wei, and Ming Zhou. 2020. At which level should we extract? an empirical analysis on extractive document summarization. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5617–5628, Barcelona, Spain (Online). International Committee on Computational Linguistics.

A TripPattern Algorithm

Generation Algorithm. Algorithm 1 presents the generation procedure of TripPattern. Line 1 initializes the alternation state variable s_{prev}, which stores the label of the most recent non-neutral pattern group. Unlike storing the pattern group set itself, storing only the group label avoids ambiguity because the vocabulary is repartitioned at every generation step.

At each generation step, the vocabulary $\mathcal{V}$ is deterministically repartitioned into a neutral group $\mathcal{V}_n$ and two pattern groups, $\mathcal{V}_{p1}$ and $\mathcal{V}_{p2}$ using the step-specific hash value and PRNG. During this repartitioning process, short subword tokens are prioritized for assignment to the neutral group until the target neutral-group proportion γ is reached, and the remaining tokens are evenly divided between the two pattern groups. The same deterministic partitioning procedure is used during detection, allowing the detector to reconstruct the corresponding step-specific partitions.

Lines 3–4 compute the step-specific hash value and reconstruct the current vocabulary partition. Lines 5–11 determine the current target pattern group according to the alternation state s_{prev}. If the most recent non-neutral pattern token was assigned to group 1, the current target group is set to $\mathcal{V}_{p2}$. If it was assigned to group 2, the current target group is set to $\mathcal{V}_{p1}$. If no previous non-neutral pattern token has been observed, $\mathcal{V}_{p1}$ is used as the default initial target group.

Line 12 defines the allowable token set $\mathcal{A}_t$ as the union of the selected target pattern group and the neutral group. Lines 13–16 obtain the next-token logits from the model, add the watermark strength δ to the logits of tokens in $\mathcal{A}_t$, normalize the adjusted logits over the full vocabulary, and sample the next token. This soft-bias procedure statistically encourages the model to select tokens from the current target pattern group or the neutral group while preserving the possibility of sampling from the full vocabulary.

Lines 17–23 update the alternation state after sampling. If the sampled token belongs to $\mathcal{V}_{p1}$, the state is updated to $s_{prev} = 1$. If the sampled token belongs to $\mathcal{V}_{p2}$, the state is updated to $s_{prev} = 2$. If the sampled token belongs to the neutral group, the state remains unchanged, since neutral tokens do not carry watermark signals. By repeating this process, TripPattern embeds a detectable alternation pattern between the two pattern groups while allowing neutral tokens to preserve generation flexibility and fluency.

Detection Algorithm. Algorithm 2 describes the detection procedure of TripPattern. The detector receives a token sequence $\mathbf{x}_{0:n}$, where x_0 is either the preceding prompt token or a predefined initial seed token. This makes the hash input $H(x_{i-1})$ well-defined for the first detected token. Line 1 initializes an empty label list $\mathbf{y}$.

For each token position, the detector reconstructs the position-specific partition of the vocabulary into a neutral group $\mathcal{V}_n$ and two pattern groups, $\mathcal{V}_{p1}$ and $\mathcal{V}_{p2}$ using the same hash/PRNG procedure, neutral-group proportion γ, and subword-length priority rule as in generation. Lines 4–10 map each non-neutral token to its pattern label: tokens in $\mathcal{V}_{p1}$ are assigned "1", and tokens in $\mathcal{V}_{p2}$ are assigned "2". Neutral tokens are excluded because they do not carry watermark signals.

After obtaining the pattern-label sequence, the detector computes the number of adjacent pattern-token pairs and the number of alternations C between adjacent labels. It then calculates the observed alternation ratio $r = C/m$ and compares it against the permutation null conditional on (n_1, n_2), where n_1 and n_2 are the numbers of labels "1" and "2", respectively. The z-score is computed using the null moments of the runs statis-

Algorithm 1 Text Generation with TripPattern

Require: Prompt tokens $\mathbf{x}_{1:n} = \{x_1, x_2, \ldots, x_n\}$, neutral group proportion $\gamma \in (0,1)$, watermark strength $\delta > 0$, model $\mathcal{M}$, vocabulary $\mathcal{V}$

Ensure: Watermarked text $x_{n+1}, x_{n+2}, \ldots, x_{n+T}$

Initialize the alternation state $s_{prev} \leftarrow \varnothing$, where $s_{prev} \in \{1, 2, \varnothing\}$ stores the label of the most recent non-neutral pattern group.
for $t = 1, 2, \ldots, T$ **do**
  $h_{n+t} \leftarrow H(x_{n+t-1})$
  Reconstruct the step-specific partition $(\mathcal{V}_n, \mathcal{V}_{p1}, \mathcal{V}_{p2})$ using h_{n+t}, the PRNG, neutral-group proportion γ, and the subword-length priority rule.
  if $s_{prev} = 1$ **then**
    Set current target pattern group $\mathcal{V}_p \leftarrow \mathcal{V}_{p2}$.
  else if $s_{prev} = 2$ **then**
    Set current target pattern group $\mathcal{V}_p \leftarrow \mathcal{V}_{p1}$.
  else
    Set current target pattern group $\mathcal{V}_p \leftarrow \mathcal{V}_{p1}$.
  end if
  Allowable token set: $\mathcal{A}_t = \mathcal{V}_p \cup \mathcal{V}_n$.
  Obtain logits $l^{n+t} \in \mathbb{R}^{|\mathcal{V}|}$ from model $\mathcal{M}$.
  Adjust logits by adding watermark strength δ to tokens in the allowable token set $\mathcal{A}_t$:

$$l_i^{n+t} = \begin{cases} l_i^{n+t} + \delta, & \text{if } v_i \in \mathcal{A}_t, \\ l_i^{n+t}, & \text{otherwise.} \end{cases}$$

  Compute probability distribution for token selection:

$$\hat{p}_{n+t}[i] = \frac{\exp(l_i^{n+t})}{\sum_{j \in \mathcal{V}} \exp(l_j^{n+t})}, \quad \forall v_i \in \mathcal{V}.$$

  Sample the next token x_{n+t} from distribution $\hat{p}_{n+t}$.
  if $x_{n+t} \in \mathcal{V}_{p1}$ **then**
    Update alternation state $s_{prev} \leftarrow 1$.
  else if $x_{n+t} \in \mathcal{V}_{p2}$ **then**
    Update alternation state $s_{prev} \leftarrow 2$.
  else
    Keep s_{prev} unchanged.
  end if
end for
return Generated tokens: $x_{n+1}, x_{n+2}, \ldots, x_{n+T}$.

Algorithm 2 Text Detection with TripPattern

Require: Text tokens $\mathbf{x}_{0:n} = \{x_0, x_1, \ldots, x_n\}$, where x_0 is a prompt token or a predefined initial seed token; neutral group proportion $\gamma \in (0,1)$; vocabulary $\mathcal{V}$

Ensure: z-score (z)

Initialize an empty list $\mathbf{y} \leftarrow []$
for $i = 1, 2, \ldots, n$ **do**
  $h_i \leftarrow H(x_{i-1})$
  Reconstruct the position-specific partition $(\mathcal{V}_n, \mathcal{V}_{p1}, \mathcal{V}_{p2})$ using h_i, the PRNG, neutral-group proportion γ, and the subword-length priority rule.
  if $x_i \in \mathcal{V}_{p1}$ **then**
    Append "1" to $\mathbf{y}$
  else if $x_i \in \mathcal{V}_{p2}$ **then**
    Append "2" to $\mathbf{y}$
  end if
end for
Let $k = |\mathbf{y}|$ be the number of non-neutral pattern tokens.
if $k < 2$ **then**
  return $z = 0$
end if
Let $m = k - 1$ be the number of adjacent pattern-token pairs.
Compute the number of alternations between adjacent pattern labels:

$$C = \sum_{i=1}^{m} \mathbb{I}[y_i \neq y_{i+1}].$$

Calculate the observed alternation ratio:

$$r = \frac{C}{m}.$$

Let $n_1 = \sum_{i=1}^{k} \mathbb{I}[y_i = 1]$ and $n_2 = k - n_1$.
Under the permutation null conditional on (n_1, n_2),

$$\mu_R = 1 + \frac{2n_1n_2}{k}, \qquad \sigma_R^2 = \frac{2n_1n_2(2n_1n_2 - k)}{k^2(k-1)}.$$

if $\sigma_R^2 = 0$ **then**
  return $z = 0$
end if
Since $R = C + 1$ and $r = C/(k-1)$,

$$\mathbb{E}[r] = \frac{\mu_R - 1}{k-1} = \frac{2n_1n_2}{k(k-1)}, \qquad \mathrm{Var}(r) = \frac{\sigma_R^2}{(k-1)^2}.$$

Compute the z-score:

$$z = \frac{r - \mathbb{E}[r]}{\sqrt{\mathrm{Var}(r)}} = \frac{R - \mu_R}{\sigma_R}.$$

return z-score z.

tic. If the z-score exceeds a predefined threshold, the null hypothesis that the text does not follow the TripPattern alternation rule is rejected, and the watermark is identified.

B Proof

We provide a mathematical analysis demonstrating how TripPattern outperforms the KGW-based framework, the commonly used two-group partitioning framework of "green" and "red" tokens, focusing on the flexibility of token selection at each decoding step.

Definition (Vocabulary Partition). *Let $\boldsymbol{V}$ be the vocabulary of the language model, and let $|\boldsymbol{V}|$ represent its size. Under the KGW-based framework, the vocabulary is partitioned into two groups, $\boldsymbol{V} = \boldsymbol{V}_g \cup \boldsymbol{V}_r, \boldsymbol{V}_g \cap \boldsymbol{V}_r = \emptyset$. Here, $\boldsymbol{V}_g$ and $\boldsymbol{V}_r$ denote the set of "green" and "red" tokens. Under the TripPattern framework, the vocabulary is partitioned into three groups, $\boldsymbol{V} = \boldsymbol{V}_{p1} \cup \boldsymbol{V}_{p2} \cup \boldsymbol{V}_n$, $\boldsymbol{V}_{p1} \cap \boldsymbol{V}_{p2} = \boldsymbol{V}_{p1} \cap \boldsymbol{V}_n = \boldsymbol{V}_{p2} \cap \boldsymbol{V}_n = \emptyset$. Here, $\boldsymbol{V}_{p1}$ and $\boldsymbol{V}_{p2}$ denote the two pattern groups and $\boldsymbol{V}_n$ is the neutral group.*

Proof. Consider an arbitrary decoding step t during text generation. Let A_t denote the set of allowable tokens for selection at each decoding step t. At each decoding step, KGW methods consistently bias selection towards tokens in the green group, V_g, resulting in a fixed allowable selection set:

$$|\mathbf{A}_t^{KGW}| = |\mathbf{V}_g|, \tag{3}$$

In contrast, at each decoding step in TripPattern, the allowable selection set $\mathbf{A}_t$, dynamically depends on the pattern group of the previously generated token, $\mathbf{V}_{prev}$, as defined below:

$$\mathbf{V}_p = \begin{cases} \mathbf{V}_{p2}, & \text{if } \mathbf{V_{prev}} = \mathbf{V}_{p1} \\ \mathbf{V}_{p1}, & \text{if } \mathbf{V_{prev}} = \mathbf{V}_{p2} \\ (\mathbf{V}_{p1} \cup \mathbf{V}_{p2}) - \mathbf{V}_{prev}, & \text{otherwise} \end{cases} \tag{4}$$

where the initial $\mathbf{V_{prev}}$ is set to *None*, and the allowable selection set for TripPattern is given by:

$$\mathbf{A}_t^{TripPattern} = \mathbf{V}_p \cup \mathbf{V}_n. \tag{5}$$

Given that the neutral group V_n always increases the allowable selection set compared to the two-group method, thus it directly follows:

$$|\mathbf{A}_t^{TripPattern}| \geq |\mathbf{A}_t^{KGW}| \tag{6}$$

when the neutral group $\mathbf{V}_n$ is non-empty and sufficiently large.

Thus, TripPattern mathematically ensures greater flexibility in token selection than KGW-based methods (when $\gamma_{\text{KGW}} \leq \frac{1+\gamma_N}{2}$), directly enhancing the naturalness and quality of generated texts. Here, γ_{KGW} indicates the proportion of green tokens for KGW-based methods and γ_N indicates the proportion of neutral tokens for TripPattern.

C Additional Experimental Results

We additionally compared our TripPattern to MorphMark (Wang et al., 2025), which adaptively scales the bias according to the token-level green probability mass on KGW. Table 7 and Table 8 show the results including MorphMark. TripPattern consistently achieves substantially higher detection performance than MorphMark across models and languages, while maintaining practical generation quality.

D Case Study

Table 9 shows an example of outputs generated by Llama-3.2-3B-Instruct based on TripPattern.

E Bin-Wise Analysis

Table 10 shows the results for the bin-wise analysis on the C4 dataset. We confirm that TripPattern consistently detects even when fewer tokens are generated.

Table 7: Experimental results for Morphmark and TripPattern on the C4 and mC4 datasets

Model	English			Korean			German			Spanish			Average		
	PPL ↓	AUC ↑	TPR@5 ↑	PPL ↓	AUC ↑	TPR@5 ↑	PPL ↓	AUC ↑	TPR@5 ↑	PPL ↓	AUC ↑	TPR@5 ↑	PPL ↓	AUC ↑	TPR@5 ↑
Llama-3.2-3B	3.49	-	-	3.86	-	-	3.53	-	-	3.14	-	-	3.51	-	-
+Morphmark	**3.62**	0.974	0.852	**4.65**	0.898	0.672	**3.86**	0.924	0.660	**3.35**	0.934	0.744	**3.87**	0.933	0.732
+TripPattern	5.24	**1.000**	**1.000**	6.33	**0.995**	0.968	5.65	**0.996**	**0.982**	4.53	0.988	0.968	5.44	**0.995**	0.980
+TripPattern w/o Len	5.26	0.998	0.990	7.88	0.986	**0.974**	6.00	0.990	0.972	5.18	**0.996**	**0.988**	6.08	0.993	**0.981**
Llama-3.1-8B	3.08	-	-	3.06	-	-	3.07	-	-	2.66	-	-	2.97	-	-
+Morphmark	**3.23**	0.971	0.892	**3.35**	0.928	0.742	**3.25**	0.937	0.676	**2.81**	0.937	0.730	**3.16**	0.943	0.760
+TripPattern	4.61	**1.000**	**1.000**	4.63	0.990	0.970	4.45	0.993	0.986	3.63	0.995	0.992	4.33	0.995	0.987
+TripPattern w/o Len	4.61	0.999	0.996	5.23	**0.999**	**0.996**	4.80	**1.000**	**0.999**	3.88	**1.000**	0.999	4.63	**1.000**	**0.998**
Qwen-2.5-3B	3.93	-	-	3.30	-	-	3.68	-	-	3.19	-	-	3.53	-	-
+Morphmark	**4.29**	0.968	0.952	**3.41**	0.887	0.724	**3.75**	0.893	0.748	**2.96**	0.847	0.672	**3.60**	0.899	0.774
+TripPattern	7.05	**0.997**	**0.994**	4.74	**0.985**	0.956	6.30	0.986	0.964	4.91	**0.969**	0.924	5.75	**0.984**	0.960
+TripPattern w/o Len	6.80	0.994	0.984	6.17	0.984	**0.962**	7.04	**0.989**	**0.966**	5.37	**0.969**	**0.940**	6.35	**0.984**	**0.963**
Qwen-2.5-7B	3.24	-	-	3.17	-	-	3.49	-	-	2.90	-	-	3.20	-	-
+Morphmark	**3.40**	0.983	0.914	**3.39**	0.957	0.838	**3.75**	0.954	0.810	**3.01**	0.954	0.856	**3.39**	0.962	0.855
+TripPattern	5.33	0.999	0.996	5.02	0.992	**0.986**	5.35	0.992	0.986	4.26	0.995	0.988	4.80	0.995	0.989
+TripPattern w/o Len	5.07	**1.000**	**1.000**	4.24	**0.997**	**0.986**	5.54	**0.997**	**0.992**	4.26	**0.996**	**0.996**	4.97	**0.998**	**0.994**

Table 8: Experimental results for Morphmark and TripPattern on the GSM8K and MGSM datasets.

Model	English			Korean			German			Spanish			Average		
	ACC ↑	AUC ↑	TPR@5 ↑	ACC ↑	AUC ↑	TPR@5 ↑	ACC ↑	AUC ↑	TPR@5 ↑	ACC ↑	AUC ↑	TPR@5 ↑	ACC ↑	AUC ↑	TPR@5 ↑
Llama-3.2-3B	72.4	-	-	7.35	-	-	11.6	-	-	11.45	-	-	25.70	-	-
+Morphmark	**72.25**	0.750	0.253	7.43	0.745	0.272	**11.37**	0.715	0.156	**11.45**	0.718	0.252	**25.63**	0.732	0.233
+TripPattern	67.10	0.950	0.776	**7.96**	0.965	0.848	10.54	**0.966**	**0.816**	**11.45**	0.932	0.648	24.26	0.953	0.772
+TripPattern w/o Len	64.22	**0.964**	**0.817**	5.46	**0.977**	**0.892**	9.78	0.946	0.784	9.78	**0.938**	**0.760**	22.31	**0.956**	**0.813**
Llama-3.1-8B	81.05	-	-	10.31	-	-	13.65	-	-	13.95	-	-	29.74	-	-
+Morphmark	**78.92**	0.752	0.287	**11.22**	0.751	0.356	**13.34**	0.712	0.148	**13.50**	0.728	0.216	**29.25**	0.736	0.252
+TripPattern	75.74	0.944	0.740	10.46	0.951	0.776	12.59	**0.973**	**0.892**	12.51	0.945	**0.760**	27.83	0.953	0.792
+TripPattern w/o Len	69.90	**0.966**	**0.835**	8.11	**0.970**	**0.840**	10.92	0.959	0.848	11.83	**0.953**	0.744	25.19	**0.962**	**0.817**
Qwen-2.5-3B	33.06	-	-	9.25	-	-	6.52	-	-	7.28	-	-	14.03	-	-
+Morphmark	**35.94**	0.722	0.227	**7.88**	0.840	0.504	**7.13**	0.774	0.280	7.20	0.748	0.248	**14.54**	0.771	0.315
+TripPattern	31.54	0.948	**0.768**	7.81	0.980	0.904	5.46	**0.992**	**0.956**	**7.43**	**0.982**	**0.932**	13.06	**0.976**	**0.890**
+TripPattern w/o Len	30.93	**0.950**	0.755	6.67	**0.988**	**0.928**	6.29	0.971	0.880	6.82	0.977	0.868	12.68	0.972	0.858
Qwen-2.5-7B	55.27	-	-	7.51	-	-	6.29	-	-	6.37	-	-	18.86	-	-
+Morphmark	**54.36**	0.637	0.129	**6.82**	0.718	0.260	**7.05**	0.706	0.296	**7.28**	0.651	0.180	**18.88**	0.678	0.216
+TripPattern	52.24	**0.870**	**0.503**	6.44	0.924	**0.716**	5.99	**0.911**	**0.624**	5.99	0.864	**0.536**	17.67	**0.892**	**0.595**
+TripPattern w/o Len	41.39	0.849	0.461	6.44	**0.935**	0.712	6.52	0.890	0.560	5.99	**0.869**	0.464	15.09	0.886	0.549

Table 9: Example of generated texts with TripPattern on the English C4 and GSM datasets. Blue and yellow represent the two different pattern groups, respectively.

Data	Watermarked-Text	Z-score
C4-English	and that the industry is facing a perfect storm of declining advertising revenue , increasing production costs, and the rise of digital media . They emphasized the need for a comprehensive approach to address these challenges , rather than simply looking for short-term cuts. The meeting also touched on the importance of divers ification and exploring new business models to sustain the company 's long-term viability. While no specific solutions were proposed , the discussions were seen as a crucial step in laying the groundwork for more substantive talks in the future. The Star Tribune 's financial struggles are well -d ocumented , with the paper's cash flows consistently lagging behind its debt payments . The company 's debt -to-equity ratio has become a significant concern , and the possibility of a debt restructuring or even bankruptcy looms large . The Strib's decline has also had a knock -on effect on the broader newspaper industry , as investors and lenders become increasingly cautious about the financial prospects of other media outlets . The fact that the Star Tribune has turned to Restr uct	6.81
GSM8K-English	Let's think backwards : The final batch of sauce is 32 oz. The problem states that the tomatoes lose 50% of their volume when cooked , so we need to find out what the original amount of sauce was before it was cooked. Since the volume is halved, we can multiply the amount of sauce by 2 to get the original amount: 32 * 2 = 64 oz. Now we know the original amount of sauce . Each 16 oz can of sauce contains 3 tomatoes . To find the number of tomatoes , we can divide the amount of sauce by the amount of tomatoes per can: 64 / 16 = 4 So there were 4 cans of tomatoes . Each can has 3 tomatoes , so the number of tomatoes is: 4 * 3 = 12 The answer is 12. However, the answer was not asked, but we were asked how may tomatoes were in the cans used	5.43

Table 10: Comparison of AUC and TPR@5 across token-length bins.

Model	Method	25–50		50–75		75–100		100–125		125–150		150–175		175∼	
		AUC	TPR@5	AUC	TPR@5	AUC	TPR@5	AUC	TPR@5	AUC	TPR@5	AUC	TPR@5	AUC	TPR@5
Llama-3.2-3B	KGW	0.997	0.984	0.997	0.992	0.996	0.990	0.996	0.990	0.996	0.990	0.996	0.990	0.997	0.990
	TripPattern	**0.998**	**0.994**	**0.999**	**0.996**	**0.999**	**0.998**	**1.000**	**0.998**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Llama-3.1-8B	KGW	**0.998**	**0.992**	**0.999**	**0.998**	**0.999**	**0.998**	0.999	**0.998**	0.999	0.998	0.999	0.998	**1.000**	**1.000**
	TripPattern	0.997	0.988	**0.999**	0.996	**0.999**	0.996	**1.000**	**0.998**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Qwen-2.5-3B	KGW	**0.987**	**0.974**	**0.990**	**0.976**	**0.992**	**0.982**	**0.993**	0.980	0.991	0.979	0.992	0.979	0.992	0.980
	TripPattern	0.985	0.972	0.988	**0.976**	0.990	0.980	**0.993**	**0.986**	**0.995**	**0.992**	**0.997**	**0.991**	**0.997**	**0.993**
Qwen-2.5-7B	KGW	0.992	0.976	0.995	**0.992**	0.997	0.990	**1.000**	**0.996**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	TripPattern	**0.995**	**0.984**	**0.997**	0.990	**0.998**	**0.996**	0.998	**0.996**	0.998	0.996	0.998	0.996	0.998	0.996