AdaptiveWatermark: Dynamic Watermarking for Large Language Models with Agent

1st Qingze Peng
School of Information Management
Sun Yat-sen University
Guangzhou, China
Email: pengqz@mail2.sysu.edu.cn

2nd Daifeng Li*
School of Information Management
Sun Yat-sen University
Guangzhou, China
Email: lidaifeng@mail.sysu.edu.cn

***Abstract*—Text watermarking embeds invisible yet detectable markers in generated content to prevent misuse of Large Language Models (LLMs). Existing methods either require fine-tuning or randomly select tokens from the vocabulary to form a green list, which increases generation likelihood. The former is computationally expensive, while the latter may degrade content quality. To address this, we propose Reinforcement Learning-based Adaptive WaterMarking (RLAWM), a novel framework using Reinforcement Learning (RL) to dynamically generate watermarks without fine-tuning. The watermark agent selects green-list tokens based on content semantics, while the detection agent identifies watermarks by analyzing green word frequency and provides feedback to improve concealment. To preserve quality, Maximum Mean Discrepancy (MMD) measures semantic differences between original and watermarked texts. Direct Preference Optimization (DPO) enables end-to-end training and system stability. On LLaMA2-7B, RLAWM improved TPR@1% by 2.48%, best F1 by 1.25%, and reduced perplexity by 3.45%.**

***Index Terms*—Large Language Models, Reinforcement Learning, Watermarking, Maximum Mean Discrepancy.**

I. Introduction

Large Language Models (LLMs) have demonstrated powerful text generation capabilities, often producing content that is indistinguishable from human-written text. However, this content can be misused to spread misinformation and violate copyright laws, posing risks to society [1]. Text watermarking technology embeds invisible markers in generated content that can be detected by models, allowing the tracing of its origin. This technique is widely used for copyright protection, document authentication, and anti-plagiarism [2].

Current watermarking methods for LLMs can be categorized into four classes: training-based, token sampling-based, logits-based, and reinforcement learning (RL)-based techniques [1]. Training-based watermarking modifies model parameters during training or fine-tuning to embed a watermark [3], [4]. However, training LLMs is computationally expensive, requiring significant resources and time, and parameter modifications may negatively affect model performance on original tasks.

Token sampling-based methods embed a watermark by altering the sampling strategy. For instance, WatME by Chen et al. [5] leverages vocabulary redundancy to achieve nearly lossless watermark embedding. Similarly, the cross-attention watermarking approach by Baldassini et al. [6] modifies the attention mechanism to influence token selection. However, these methods are often limited to specific generation modes, restricting their applicability and generalization across tasks.

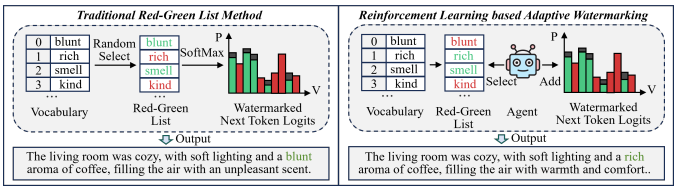

Fig. 1. The figure shows an example of watermark generation using the traditional red-green list and our proposed RLAWM.

The logits-based watermarking method, red-green word lists, uses a hash index to randomly select words from the LLM vocabulary for the green list and increases the probability of generating these words. The watermark is determined by the frequency of green words in the output text [7]. Zhang et al.'s REMARK-LLM [8] improves this method with a robust encoding mechanism to enhance resistance to attacks. Guo et al. [9] proposed a semantically balanced green-red word list technique to reduce the impact on text quality while maintaining watermark strength. However, random selection of green-list words can still lead to poor-quality content if semantically inappropriate words are chosen (e.g., Fig.1).

Currently, few studies focus on using RL for watermark generation. Xu et al. [10] used RL to adjust the weights of the LLM during training to embed and detect the watermark through a paired detector, while Guo et al. [11] utilized LoRA-based agents for efficient fine-tuning. All existing RL-based watermarking methods require fine-tuning model parameters, demand substantial computational resources, and may degrade the model's output quality.

This paper's core idea is to use RL for adaptive red-green list watermark generation. RL enables the agent to autonomously learn how to select and adjust the green list and its generation probability by sensing context and exploring. This allows LLMs to generate more appropriate green words and improve response quality. Crucially, watermark generation occurs only during the final token sampling process, requiring no parameter fine-tuning, thus reducing computational costs. However, this method faces two main challenges. First, embedding a watermark can alter the model's output probability

distribution, potentially degrading content quality [12], [13]. Second, watermark detection based on the red-green list often relies on a fixed threshold, which may become invalid if the generation environment changes.

To address these challenges, we propose the Reinforcement Learning-based Adaptive WaterMarking (RLAWM) framework, an RL-driven dynamic watermarking approach. Specifically, we design an adversarial training setup in which the watermark generation agent adaptively constructs the green word list and adjusts the probability of green words to embed a watermark. The detection agent is responsible for identifying the watermark and providing feedback to the generation agent to improve the watermark's stealth. Additionally, we employ Maximum Mean Discrepancy (MMD) to measure the semantic shift in the LLM-generated text before and after watermark embedding, ensuring output quality. Finally, we use the Direct Preference Optimization (DPO) algorithm for end-to-end training to stabilize performance.

The main contributions of this paper are as follows:

- We introduce RL into the red-green list-based watermark generation process for the first time, demonstrating that an agent can adaptively learn to generate covert watermarks without any model parameter tuning.
- We propose RLAWM, an end-to-end watermark generation algorithm. The watermark generation agent constructs the green word list and assigns generation probabilities, while the detection agent identifies the watermark and provides feedback to guide the generation process. We also use MMD to minimize the impact of watermarking on the output quality of the LLM.
- We conducted experiments on the C4 dataset. On the LLaMA2-7B model, RLAWM improved TPR@1% by 2.48%, Best F1 by 1.25%.

II. Preliminaries

Given a pre-trained LLM M and an input context $c_{1:t-1} = [w_1, w_2, \ldots, w_{t-1}]$, the model generates an output sequence $x = [w_1, w_2, \ldots, w_T]$ containing detectable watermark features. To verify the watermark, we define a detection function $\mathcal{D} : x \to 0, 1$, outputting 1 if a watermark is detected and 0 otherwise. The Red-Green List method embeds watermarks by increasing the probability of generating specific tokens, known as the green list. For each generation step t, the green list G_t is constructed using a deterministic hash function as follows:

$$G_t = v \in V \mid \text{hash}(c_{1:t-1} \oplus v \oplus k) \bmod 2^r < \gamma \cdot 2^r \quad (1)$$

Here, $c_{1:t-1}$ is the context sequence, $v \in V$ represents a token from the vocabulary V, and hash$(\cdot)$ is a hash function. The parameter k is a secret key, and r is a hash precision parameter. The green list ratio is $\gamma \in [0, 1]$. To embed the watermark, the adjusted probability distribution for selecting green-list tokens, $p'(w_t \mid c_{1:t-1})$, at time step t is:

$$p'(w_t \mid c_{1:t-1}) = \begin{cases} \sigma\left(\log p(w_t \mid c_{1:t-1}) + \delta\right), & \text{if } w_t \in G_t \\ p(w_t \mid c_{1:t-1}), & \text{otherwise} \end{cases} \quad (2)$$

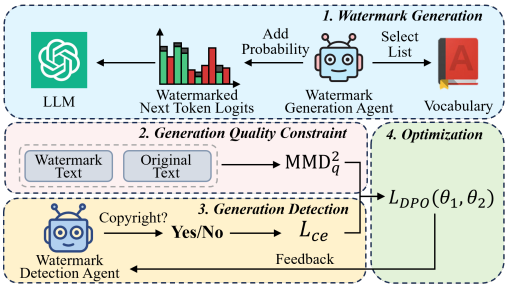

Fig. 2. RL-based Adaptive Watermarking Algorithm Computational Process.

Here, w_t is the current token, σ is the softmax function, and $\delta > 0$ is the probability enhancement parameter. To detect the watermark, the statistic z is computed as:

$$z = \frac{s - n\gamma}{\sqrt{n\gamma(1-\gamma)}} \quad (3)$$

where n is the length of the generated sequence and s is the number of green-list tokens. If z exceeds a threshold, the sequence is classified as containing a watermark.

III. Method

A. Overview

This paper proposes the RLAWM framework (e.g., Fig. 2). The generation agents enable the LLM to produce watermarked text, while the detection agents identify watermark features and use MMD to ensure generation quality. The system is trained end-to-end with DPO, allowing adaptive generation of high-quality watermarked text.

B. Design of Watermark Generation Agent

The watermark generation process is modeled as a Markov Decision Process (MDP) [14]. At each time step t, the agent observes the current state s_t and selects an action a_t to modify the watermark embedding. The state space is derived from the sample probabilities of the token w_t:

$$s_{\text{vocab}}^{(t)} = \left(p(w \mid c_{1:t-1}), V_t\right) \quad (4)$$

where $p(w \mid c_{1:t-1})$ denotes the probability distribution over the vocabulary given the previous context $c_{1:t-1}$, and V_t is the vocabulary subset relevant to the current context. To improve flexibility, the action space introduces a controllable hash seed, $a_{\text{seed}}^{(t)}$, sampled from the interval $[0, 1]$. The green list at time step t is constructed as follows:

$$G_t = \{v \in V \mid \text{hash}\left(c_{1:t-1} \oplus v \oplus k \oplus \lfloor a_{\text{seed}}^{(t)} \cdot 2^{16} \rfloor\right) \bmod 2^r < \gamma \cdot 2^r\} \quad (5)$$

where hash$(\cdot)$ is a hash function, v is a token from the vocabulary V, k is a predefined secret key, $\lfloor \cdot \rfloor$ denotes the floor function, r is the hash precision parameter, and $\gamma = 0.5$ is a predefined ratio that determines the expected size of the green list. Additionally, the action space allows the agent to control the probability enhancement strength applied to the green list

tokens. This enhancement is represented by $a_\delta^{(t)}$, within the interval $[\delta_{\min}, \delta_{\max}]$. The adjusted probability distribution is:

$$D_t = \sum_{v \in G_t} p(v \mid c_{1:t-1}) \cdot \exp(a_\delta^{(t)}) + \sum_{v \notin G_t} p(v \mid c_{1:t-1}) \quad (6)$$

Here, $\mathbf{I}G_t(v)$ denotes the indicator function. Additionally, $\exp(a_\delta^{(t)})$ applies an enhancement factor to adjust the likelihood of green list tokens, while D_t is the normalization factor. The policy network uses an encoder-decoder architecture, with input features processed through $N = 6$ Transformer blocks [15]. After encoding, the network branches into two pathways: one controls the seed for green list construction, and the other adjusts probability enhancement. The Seed Control Branch determines the hash seed value based on the current state s_t. The shared feature representation h_t is passed through three fully connected (FC) layers with ReLU activations to reduce dimensionality. The output is modeled using a Beta distribution, constraining the seed parameter within $[0, 1]$:

$$\pi_{\text{seed}}(a_{\text{seed}}^{(t)} \mid s_t) = \text{Beta}(\alpha_{\text{seed}}, \beta_{\text{seed}}) \quad (7)$$

where α_{seed} and β_{seed} are hyperparameters. The Beta distribution controls the seed within $[0, 1]$. The Intensity Control Branch determines the enhancement strength $a_\delta^{(t)}$. The shared feature representation h_t is processed through three FC layers with ReLU activations, mapped to a 2-dimensional vector, and modeled using a truncated normal distribution to ensure the enhancement strength remains within a safe range:

$$\pi_\delta(a_\delta^{(t)} \mid s_t) = \text{TruncatedNormal}(\mu_\delta, \sigma_\delta^2) \quad (8)$$

where μ_δ and σ_δ^2 are the mean and variance parameters predicted by the network.

C. Design of Watermark Detection Agent

The Watermark Detection Agent autonomously determines if a text contains a watermark. Given an input token sequence $x = [w_1, w_2, \ldots, w_T]$, where w_i is the i-th token, the agent also receives a green list marking sequence $g = [g_1, g_2, \ldots, g_T]$, where each $g_i \in 0, 1$ indicates if w_i is part of the green list. The agent uses a six-layer Transformer architecture [15], identical to the policy network. The model uses both the token and green list marking sequences for binary classification, trained with cross-entropy loss:

$$\mathcal{L}_{\text{ce}} = -\sum_i y_i \log(\text{softmax}(\text{logits}_i)), \quad (9)$$

where p_t is the predicted probability for the true class y_i of token w_i at position t, and logits_i is the token probability.

D. Design of Generation Quality Constraint Mechanism

We minimize the impact on text quality by reducing the semantic similarity between the model output before and after embedding the watermark. We use MMD [16] to quantify this similarity. Given two text distributions, P and Q, MMD is defined as:

$$\text{MMD}^2_{\mathcal{H}}(P, Q) = |\mathbb{E}x \sim P[\phi(x)] - \mathbb{E}y \sim Q[\phi(y)]|^2_{\mathcal{H}}, \quad (10)$$

where ϕ is the feature mapping function, and $\mathcal{H}$ is the Reproducing Kernel Hilbert Space (RKHS). The notation $\mathbb{E}_{x \sim P}[\phi(x)]$ represents the expected value of $\phi(x)$ under distribution P, and similarly for Q. In our approach, we use SentenceBERT to extract the semantic representations of input texts, x_i, from both the original and watermarked text sets. We use the Gaussian Radial Basis Function (RBF) kernel [17]:

$$k(e_i, e_j) = \exp\left(-\frac{|e_i - e_j|^2}{2\sigma^2}\right), \quad (11)$$

where e_i and e_j are the semantic representations of two tokens, and σ is the bandwidth parameter. The value of σ is determined using the median heuristic:

$$\sigma = \text{median}(|e_i - e_j|^2), \quad (12)$$

which adapts the kernel to the data scale. To estimate the MMD between the original text set $x_i^{(0)} i = 1^m$ and the watermarked set $x_i^{(1)} i = 1^m$, we compute:

$$\text{MMD}_{\text{q}}^2 = \frac{1}{m^2} \sum_{i,j=1}^{m} k(e_i^{(0)}, e_j^{(0)}) + \frac{1}{m^2} \sum_{i,j=1}^{m} k(e_i^{(1)}, e_j^{(1)}) - \frac{2}{m^2} \sum_{i,j=1}^{m} k(e_i^{(0)}, e_j^{(1)}). \quad (13)$$

This estimation considers pairwise similarities within the original and watermarked text sets, as well as the cross-similarities between them.

E. DPO Optimization Strategy

We adopt the DPO algorithm [18] to optimize the agents' performance. For the Watermark Detection Agent, the detection success rate is defined as the reward function:

$$R_{\text{detect}}(x) = \mathbb{E}[\pi_2(y = 1|x; \theta_2^{\text{current}})], \quad (14)$$

where $\pi_2(y = 1|x; \theta_2^{\text{current}})$ represents the probability that the detection agent correctly classifies input x as containing a watermark, given the current model parameters $\theta_2^{\text{current}}$. The Watermark Generation Agent aims to create watermarks difficult for the Detection Agent to distinguish, while the Detection Agent works to improve its detection success rate, learning to identify better watermarks under adversarial conditions. We define a quality preservation reward to ensure watermark generation does not degrade output quality:

$$R_{\text{quality}}(x) = \exp(-\lambda_{\text{mmd}} \cdot \text{MMD}_{\text{q}}^2), \quad (15)$$

where MMD_{q}^2 is the estimation of MMD between the original and watermarked text distributions, and λ_{mmd} controls the penalty for quality loss. The total reward function is:

$$R_{\text{total}}(x) = \alpha R_{\text{detect}}(x) + \beta R_{\text{quality}}(x), \quad (16)$$

where α and β are weight parameters. The loss function in DPO is formulated as:

$$\mathcal{L}_{\text{DPO}}(\theta_1, \theta_2) = -\mathbb{E}_{(c, x_w, x_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_{\theta_1}(x_w|c)}{\pi_{\text{ref}}(x_w|c)} - \beta \log \frac{\pi_{\theta_1}(x_l|c)}{\pi_{\text{ref}}(x_l|c)}\right)\right]. \quad (17)$$

TABLE I
COMPARISON OF THE RESULTS OF THE MAIN EXPERIMENT.

LLM	Model	TPR@1%↑	Best F1↑	PPL↓	SR↑	TPR@1%↑(Word-s/30%)
OPT-2.7B	KGW	0.892	0.937	12.41	0.873	0.629
	REMARK	0.903	0.948	12.03	0.887	0.712
	SemStamp	0.859	0.921	11.89	0.862	0.695
	WatME	0.861	0.908	11.71	0.847	0.673
	MorphMark	0.924	0.961	11.37	0.908	0.785
	RLMark	0.881	0.931	12.63	0.874	0.698
	CRMark	0.916	0.955	11.64	0.901	0.753
	RLAWM	**0.947**	**0.973**	**11.08**	**0.931**	**0.829**
OPT-6.7B	KGW	0.901	0.942	11.03	0.885	0.651
	REMARK	0.916	0.958	10.98	0.899	0.728
	SemStamp	0.884	0.928	10.47	0.871	0.708
	WatME	0.867	0.915	10.14	0.853	0.686
	MorphMark	0.935	0.927	10.45	0.919	0.801
	RLMark	0.896	0.938	11.76	0.881	0.714
	CRMark	0.927	0.962	10.64	0.912	0.773
	RLAWM	**0.956**	**0.976**	**9.81**	**0.942**	**0.839**
LLaMA2-7B	KGW	0.895	0.938	10.87	0.879	0.647
	REMARK	0.909	0.951	10.58	0.893	0.721
	SemStamp	0.879	0.924	10.39	0.866	0.701
	WatME	0.861	0.911	10.24	0.849	0.673
	MorphMark	0.928	0.963	9.85	0.912	0.792
	RLMark	0.890	0.935	11.26	0.871	0.707
	CRMark	0.923	0.958	10.56	0.906	0.764
	RLAWM	**0.951**	**0.975**	**9.51**	**0.936**	**0.831**
Mistral-7B	KGW	0.908	0.945	10.09	0.891	0.663
	REMARK	0.927	0.953	10.18	0.904	0.739
	SemStamp	0.887	0.931	9.98	0.874	0.716
	WatME	0.873	0.918	9.24	0.857	0.694
	MorphMark	0.941	0.973	9.41	0.925	0.814
	RLMark	0.902	0.943	10.71	0.887	0.725
	CRMark	0.933	0.965	9.94	0.918	0.786
	RLAWM	**0.962**	**0.981**	**9.12**	**0.948**	**0.852**

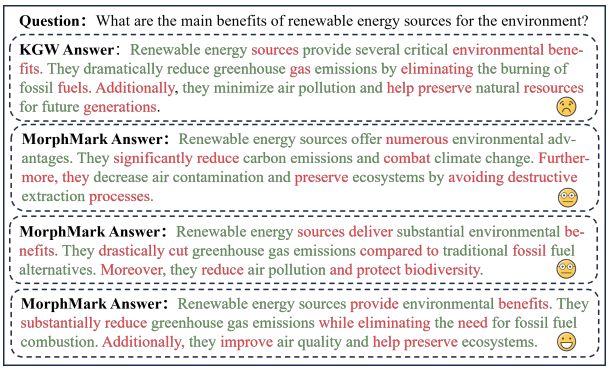

Fig. 3. Generation performance of different methods in QA tasks.

where (c, x_w, x_l) is the preference data triplet consisting of a context c, a watermark sample x_w, and a non-watermarked sample x_l. π_{ref} is the reference strategy, initialized as the Watermark Generation Agent. The parameter β is the temperature parameter, controlling preference strength in optimization.

IV. EXPERIMENT AND RESULT

A. Experimental Settings

Dataset: Following [11], we use the WikiText-103 [19] and GSM8K datasets [20] for model training. Following [3], [21], we extract 1,000 samples from the C4 dataset [22] to evaluate the performance of RLAWM.

Implementation Details: Following [3], [11], [21], we use OPT-2.7B, OPT-6.7B [23], LLaMA2-7B [24], and Mistral-7B [25] as backbone models. For semantic representation extraction, we employ the all-MiniLM-L6-v2 model [26]. The green vocabulary ratio is set to 0.5, and hash precision is 16. During training, we use the AdamW optimizer with a learning rate of 2×10^{-5}, weight decay of 0.01, and Beta parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

Baselines: Our baselines include KGW [7], REMARK [8], SemStamp [27], WatME [5], MorphMark [3], RLMark [10] and CRMark [11].

Metrics: Following [3], we evaluate the effectiveness of the watermarks in terms of detectability (measured by TPR@1% and the best F1 score) and robustness (evaluated under a Word-S/30% attack, where 30% of words are randomly replaced with synonyms from WordNet). Additionally, text quality is assessed using perplexity (PPL). We also use the successful trigger ratio (SR) as a metric for evaluating the watermark's ability to be triggered during verification.

B. Main Results

In this study, we evaluate the performance of the proposed RLAWM framework. As shown in Table I, RLAWM outperforms all baseline methods across all model sizes. For TPR@1% on the OPT-2.7B model, RLAWM shows a 4.9% improvement over MorphMark. For the F1 score, RLAWM achieves a 2.6% improvement over REMARK. RLAWM SR improves by 5.7% and 6.3% on the OPT-6.7B and Mistral-7B models, respectively. Under the Word-s/30% setting, TPR@1% shows even greater gains. RLAWM's ability to learn optimal watermarking strategies that scale with model size is a capability not consistently demonstrated by baseline methods. Fig. 4 shows the model's performance based on Total Reward, Detection Reward, and Quality Reward over training epochs from 0 to 2000. The growth phase is steep early on

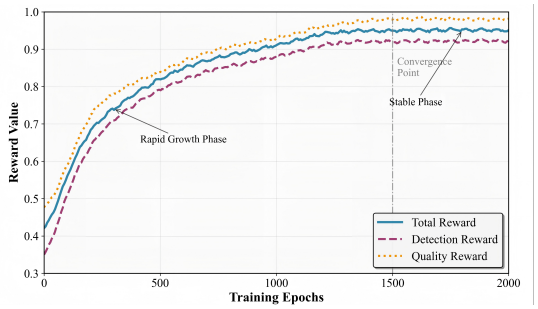

Fig. 4. RLAWM Training Convergence Curves.

TABLE II
COMPUTING RESOURCE CONSUMPTION COMPARISON

Method	Traning(h)	Memory (GB)	Inference Delay (ms)	GPU(%)
KGW	-	0.5	12.6	-
MorphMark	-	0.7	15.7	-
RLMark	48.2	24.7	89.4	87.3
CRMark	72.1	31.6	156.2	92.7
RLAWM	36.2	18.6	45.8	78.4

and stabilizes after Epoch 1500, reflecting the model's rapid learning of essential features.

Table II evaluates and compares the resource consumption of five different methods in terms of training time, memory usage, inference delay, and GPU utilization. RLAWM is 24.9% faster than RLMark, making it more efficient in terms of training time. It also uses 24.7% less memory compared to RLMark. In terms of inference delay, RLAWM performs better, achieving a delay of 45.8 ms, which is 48.8% lower than that of RLMark, indicating higher efficiency during inference. Additionally, RLAWM uses 78.4% of the GPU, which is 10.2% lower than RLMark, showing that it is less demanding on GPU resources while maintaining efficient performance.

C. Ablation Study

1) Impact of Different Module: The results in Table III show improvements as components are added to the basic Red-Green List method. The baseline Red-Green List achieves a TPR@1% of 0.895. Adding the Generation Agent increases this to 0.921, a 2.9% improvement. Incorporating the Detection Agent further raises the TPR@1% to 0.937, a 1.7% gain. The Generation Agent improves TPR@1%, F1 score, and PPL, while the Detection Agent enhances watermark detection and text quality. Though the MMD and DPO components offer smaller gains, they refine the watermarking process by adjusting the embedding space.

2) Impact of Hyperparameters α and β: The goal of this experiment is to analyze model performance by adjusting the detection weight α and the quality weight β (e.g., Fig. 5). The results show an optimal point at $\alpha = 0.5$ and $\beta = 0.5$ for both metrics. At this setting, TPR@1% peaks at 0.951, and PPL is minimized at 9.54. As α increases, TPR@1% improves, but PPL rises. Prioritizing detection identifies more true positives, but reduces text quality. Conversely, as β increases, PPL decreases, indicating better text quality, while TPR@1% drops, showing a reduced ability to detect watermark features.

TABLE III
ABLATION EXPERIMENT RESULTS ON LLAMA2-7B

Method	TPR@1%↑	Best F1↑	PPL↓	SR↑	TPR@1%↑(Word-s/30%)
Red-Green List	0.895	0.938	10.87	0.879	0.647
w/ Generation Agent	0.921	0.957	10.48	0.904	0.748
w/ Detection Agent	0.937	0.961	10.27	0.918	0.782
w/ MMD	0.943	0.971	9.78	0.927	0.816
w/ DPO	0.951	0.975	9.51	0.936	0.831

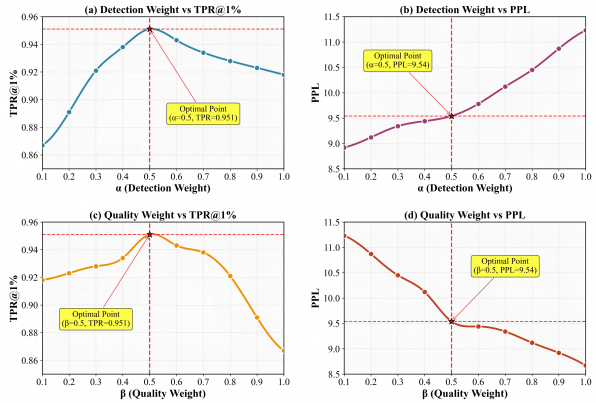

Fig. 5. Hyperparameter Sensitivity Analysis.

3) Impact of Different Text Lengths: The experiment in Fig. 6 evaluates eight watermarking methods based on text length, using TPR@1% as the metric. At 100 tokens, RLAWM achieves a TPR@1% of 0.978. At 600 tokens, RLAWM shows a 3.7% drop in performance from short to long texts. Generally, short texts have a smaller vocabulary, making watermark embedding easier, while longer texts introduce greater lexical variety, reducing embedding consistency.

D. Model Robustness

We conducted experiments under several adversarial attack scenarios, including common text manipulation techniques such as word sampling (Word-S) at varying proportions, paraphrasing, grammar correction, and style transfer. The results, shown in Table IV, indicate that under the Word-S (10%) attack, RLAWM demonstrates the highest watermark strength. As the word sampling proportion increases to 50%, detection rates decline across all methods, but RLAWM still maintains the highest performance, with a detection rate of 0.765. Against paraphrasing attacks, RLAWM again achieves the best result, with a 25.2% improvement over KGW. RLAWM shows strong resilience to these attacks, proving its watermarking strategy is robust against both word-level alterations and semantic-preserving transformations.

V. CONCLUSION

In this paper, we introduced RLAWM, a novel framework that uses RL to dynamically generate watermarks for LLMs without requiring parameter fine-tuning. The framework

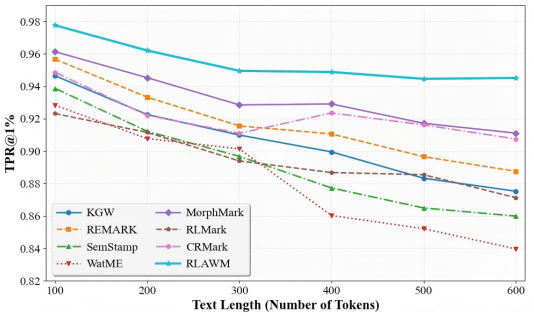

Fig. 6. Performance Comparison Across Different Text Lengths.

TABLE IV
ROBUSTNESS COMPARISON UNDER DIFFERENT ATTACK INTENSITIES

Attack	KGW	REMARK	MorphMark	CRMark	RLAWM
Word-S/10%	0.834	0.854	0.891	0.863	0.924
Word-S/30%	0.663	0.739	0.814	0.786	0.852
Word-S/50%	0.513	0.598	0.691	0.642	0.765
Paraphrase	0.523	0.617	0.738	0.681	0.776
Grammar Correction	0.724	0.781	0.836	0.798	0.869
Style Transfer	0.554	0.535	0.652	0.598	0.712

allows the watermark generation agent to adaptively select tokens for the green list based on semantic context, while a detection agent provides feedback for watermark concealment. Using MMD, we minimize the difference between the model's output before and after watermark injection, ensuring high-quality content with concealed watermarks. Through extensive experiments, we demonstrated the effectiveness and robustness of RLAWM in embedding watermarks.

ACKNOWLEDGMENT

This work was supported by the Guangdong Natural Science Foundation General Project under Grant No. 2024A1515011793.

REFERENCES

[1] A. Liu, L. Pan, Y. Lu, J. Li, X. Hu, X. Zhang, L. Wen, I. King, H. Xiong, and P. Yu, "A survey of text watermarking in the era of large language models," *ACM Computing Surveys*, vol. 57, no. 2, pp. 1–36, 2024.

[2] X. Wang, H. Jiang, Y. Yu, J. Yu, Y. Lin, P. Yi, Y. Wang, Y. Qiao, L. Li, and F.-Y. Wang, "Building intelligence identification system via large language model watermarking: a survey and beyond," *Artificial Intelligence Review*, vol. 58, no. 8, p. 249, 2025.

[3] Z. Wang, T. Gu, B. Wu, and Y. Yang, "Morphmark: Flexible adaptive watermarking for large language models," *CoRR, arXiv:2505.11541*, 2025.

[4] Z. Sun, X. Du, F. Song, and L. Li, "Codemark: Imperceptible watermarking for code datasets against neural code completion models," in *Proceedings of the ESEC/FSE*, 2023, pp. 1561–1572.

[5] L. Chen, Y. Bian, Y. Deng, D. Cai, S. Li, P. Zhao, and K.-F. Wong, "Watme: Towards lossless watermarking through lexical redundancy," in *Proceedings of the ACL*, 2024, pp. 9166–9180.

[6] F. B. Baldassini, H. H. Nguyen, C.-C. Chang, and I. Echizen, "Cross-attention watermarking of large language models," in *Proceedings of the ICASSP*, 2024, pp. 4625–4629.

[7] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein, "A watermark for large language models," in *Proceedings of the ICML*, 2023, pp. 17 061–17 084.

[8] R. Zhang, S. S. Hussain, P. Neekhara, and F. Koushanfar, "Remark-llm: A robust and efficient watermarking framework for generative large language models," in *Proceedings of the USENIX Security*, 2024, pp. 1813–1830.

[9] Y. Guo, Z. Tian, Y. Song, T. Liu, L. Ding, and D. Li, "Context-aware watermark with semantic balanced green-red lists for large language models," in *Proceedings of the EMNLP*, 2024, pp. 22 633–22 646.

[10] X. Xu, Y. Yao, and Y. Liu, "Learning to watermark llm-generated text via reinforcement learning," in *Proceedings of the ICML WMARK*, 2024.

[11] S. Guo, K. Pang, Z. Yang, Y. Li, Y. Qing, and Y. Huang, "Reinforcement learning-based copyright protection watermarking for large language model," in *Proceedings of the ACM IHMMSec*, 2025, pp. 114–120.

[12] Q. Pang, S. Hu, W. Zheng, and V. Smith, "No free lunch in llm watermarking: Trade-offs in watermarking design choices," *Proceedings of the NeurIPS*, vol. 37, pp. 138 756–138 788, 2024.

[13] Q. Wu and V. Chandrasekaran, "Bypassing llm watermarks with color-aware substitutions," in *Proceedings of the ACL*, 2024, pp. 8549–8581.

[14] J. Tang, J. Song, and A. Gupta, "Dynamic watermarking for finite markov decision processes," *IEEE Open Journal of Control Systems*, 2025.

[15] C. Tong, I. Natgunanathan, Y. Xiang, J. Li, T. Zong, X. Zheng, and L. Gao, "Enhancing robustness of speech watermarking using a transformer-based framework exploiting acoustic features," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2024.

[16] S. Zhang, Y. Song, J. Yang, Y. Li, B. Han, and M. Tan, "Detecting machine-generated texts by multi-population aware optimization for maximum mean discrepancy," in *Proceedings of the ICLR*, 2024.

[17] S. Jaiswal and M. K. Pandey, "Digital watermark extraction using rs-knn and rs-lda with lwt and statistical features," *SN Computer Science*, vol. 4, no. 5, p. 496, 2023.

[18] Y. Chen, J. Tan, A. Zhang, Z. Yang, L. Sheng, E. Zhang, X. Wang, and T.-S. Chua, "On softmax direct preference optimization for recommendation," *Proceedings of the NeurIPS*, vol. 37, pp. 27 463–27 489, 2024.

[19] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," in *Proceedings of the ICLR*, 2017.

[20] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano *et al.*, "Training verifiers to solve math word problems," *CoRR, arXiv:2110.14168*, 2021.

[21] L. Pan, A. Liu, Z. He, Z. Gao, X. Zhao, Y. Lu, B. Zhou, S. Liu, X. Hu, L. Wen *et al.*, "Markllm: An open-source toolkit for llm watermarking," in *Proceedings of the EMNLP*, 2024, pp. 61–71.

[22] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.

[23] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin *et al.*, "Opt: Open pre-trained transformer language models," *CoRR, arXiv:2205.01068*, 2022.

[24] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *CoRR, arXiv:2302.13971*, 2023.

[25] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," *CoRR, arXiv:2310.06825*, 2023.

[26] N. Reimers, E. Omar, G. Joao, and A. Tom, "Train the best sentence embedding model ever with 1b training pairs," *Community week using JAX/Flax for NLP CV. HugginFace*, vol. 6, 2021.

[27] A. Hou, J. Zhang, T. He, Y. Wang, Y.-S. Chuang, H. Wang, L. Shen, B. Van_Durme, D. Khashabi, and Y. Tsvetkov, "Semstamp: A semantic watermark with paraphrastic robustness for text generation," in *Proceedings of the NAACL*, 2024.