REMARK-LLM: A *R*obust and *E*fficient Water*mark*ing Framework for Generative Large Language Models

Ruisi Zhang, Shehzeen Samarah Hussain, Paarth Neekhara,
and Farinaz Koushanfar, *University of California, San Diego*

https://www.usenix.org/conference/usenixsecurity24/presentation/zhang-ruisi

This paper is included in the Proceedings of the
33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

Open access to the Proceedings of the
33rd USENIX Security Symposium
is sponsored by USENIX.

REMARK-LLM: A *R*obust and *E*fficient Water*mark*ing Framework for Generative Large Language Models

Ruisi Zhang, Shehzeen Samarah Hussain, Paarth Neekhara, and Farinaz Koushanfar

University of California, San Diego

Abstract

We present REMARK-LLM, a novel efficient, and robust watermarking framework designed for texts generated by large language models (LLMs). Synthesizing human-like content using LLMs necessitates vast computational resources and extensive datasets, encapsulating critical intellectual property (IP). However, the generated content is prone to malicious exploitation, including spamming and plagiarism. To address the challenges, REMARK-LLM proposes three new components: (i) a learning-based message encoding module to infuse binary signatures into LLM-generated texts; (ii) a reparameterization module to transform the dense distributions from the message encoding to the sparse distribution of the watermarked textual tokens; (iii) a decoding module dedicated for signature extraction; Besides, we introduce an optimized beam search algorithm to generate content with coherence and consistency. REMARK-LLM is rigorously trained to encourage the preservation of semantic integrity in watermarked content, while ensuring effective watermark retrieval. Extensive evaluations on multiple unseen datasets highlight REMARK-LLM's proficiency and transferability in inserting 2× more signature bits into the same texts when compared to prior art, all while maintaining semantic integrity. Furthermore, REMARK-LLM exhibits better resilience against a spectrum of watermark detection and removal attacks.

1 Introduction

Recent advancements in the development of large language models (LLMs) such as ChatGPT [38], LLaMA [43], and GPT-4 [32] indicate a paradigm shift in human-computer dialogue interactions. These AI-powered systems have the capacity to generate human-like text responses and are integrated into various aspects of our daily lives. Training LLMs [4, 29, 34] requires substantial computational resources and extensive datasets, both of which are critical components of valuable intellectual property (IP). Concurrently, the increasing capabilities of these foundational models present potential risks in the form of malicious uses, including spam and plagiarism. Thus, there is a need to devise mechanisms to claim ownership of LLM-generated text and trace the distribution of the generated content.

Watermarking offers a promising solution to tackle two persistent issues: asserting ownership of generated output and tracing the source of content. By embedding watermark signatures into the outputs of LLMs, model proprietors can effectively monitor their content utilizations and validate their ownership. As such, the system can be applied to detect plagiarism to maintain academic integrity [20], protect the copyrights of LLM owners [21], and track the distribution of the potential misinformation generated by LLMs [23]. Existing literature on text watermarking can be classified into three categories [39]: (1) rule-based watermarking [16, 28], (2) inference-time watermarking [18, 22], and (3) neural-based watermarking [1]. The rule-based watermarking replaces synonym [15] or transforms syntactic structures [5] in the paragraph to insert as watermarks. Such manually designed features make the inserted signatures statistically removable through word distribution or syntactical analysis. The inference-time watermarking [18] splits vocabulary into green/red lists and restricts the LLM decoding to predict the next tokens from the green list. While the inserted watermarks are robust against attacks, the decoding strategy drastically distorts the semantic similarity between the watermarked and original LLM outputs. The neural-based approach [1] leverages an end-to-end learning technique to integrate the binary watermarking signatures into the LLM-generated texts while maintaining semantic coherence. However, the maximum encodable signature length per token segment is limited compared with the rule-based and inference-time frameworks, thus hindering the practical usage of this approach.

Generally speaking, watermarking text data presents several challenges. First, text data exhibits a pronounced sparsity compared with other modalities, such as images and audio. For instance, a 256-pixel image offers approximately 65k feasible positions for watermark insertion [30], whereas the maximum token limit in GPT-4 [33] is 8.2k. Besides, text data is fragile in that subtle alterations may obfuscate or compromise the semiotic fidelity [51], whereas minor perturbations in images can remain imperceptible. Note that all inserted watermarks, including the ones applied to LLM-generated text, should be resilient to potential watermark removal and detection attacks from end users [17].

This paper proposes REMARK-LLM, a new robust and efficient watermarking framework to insert watermarks into LLM-generated texts without compromising the semantic integrity. REMARK-LLM composed of three key modules, namely, message encoding, reparameterization, and message decoding. The message encoding module encodes the LLM-generated texts and their corresponding signatures into latent feature space. Their feature representations are added and yield the watermarked distribution over the vocabulary. Sequentially, the reparameterization component exploits Gumbel-Softmax methodology [14] to transform the watermarked distribution to the sparse distribution of the watermarked textual tokens. Next, the message decoding module extracts watermarking signatures by leveraging a transformer to predict the inserted messages. REMARK-LLM enhances its robustness by incorporating malicious transformations during training, including text addition, deletion, and substitution over the transformed textual token distribution into the message decoding phase.

The three modules are trained end-to-end, targeting to (1) preserve the semantic fidelity by minimizing a semantic loss between the original LLM-generated and watermarked texts, (2) ensure watermark extraction by minimizing a message recovery loss between the inserted and extracted watermarking signatures from the watermarked texts, and (3) enhance robustness by extracting watermarking signatures from the malicious transformations.

With the trained REMARK-LLM, the LLM proprietor leverages the message encoding module to embed binary signatures into the LLM-generated texts and obtain a watermarked distribution. An optimized beam search algorithm subsequently translates the output of this module's distribution into watermarked texts, ensuring linguistic coherence, unwavering semantic fidelity, and the successful extraction of signatures. Next, the watermarked texts are disseminated to end-users as coherent responses. The watermark existence can be verified by extracting the inserted signatures using the message decoding module. It compares the extracted messages with the inserted signatures to determine if the LLM generates the texts.

In summary, our contributions are as follows:

- We introduce REMARK-LLM, a robust and efficient watermarking framework tailored for LLMs that maintains the semantic integrity of watermarked text while exhibiting resilience against potential watermark detection and removal attacks.
- We devise the watermarking framework with novelties lie in (i) a pre-trained sequence-to-sequence backbone module to significantly improve the transferability and capacity of the watermarking framework; (ii) an optimizing beam search module for balanced readability and extraction accuracy; (iii) incorporate potential malicious transformation into the training for improved robustness against watermark removal attacks.
- We validate the effectiveness and robustness of REMARK-LLM by conducting extensive evaluations on multiple datasets: (i) REMARK-LLM can successfully embed 2× more signatures into LLM-generated text compared to prior art AWT [1] within 1.5 seconds; (ii) REMARK-LLM maintains the LLM-generated texts' semantics with an average of 0.90 BERT score and exhibits transferability to watermark natural language from unseen sources without extra fine-tuning; (iii) REMARK-LLM is resilient under various watermark detection and removal attacks and maintains an average AUC of 0.85.

Paper Organization: The rest of the paper is organized as follows: Section 2 provides the background and related work on text watermarking. Section 3 describes the problem formulation, including the watermarking objectives, challenges, and potential threats. Section 4 introduces the proposed watermarking scheme REMARK-LLM, by detailing the watermarking architecture, as well as the signature insertion and extraction at the inference time. Section 5 details the extensive experiments on different datasets and comparisons with existing watermark schemes regarding effectiveness, efficiency, and robustness. Finally, Section 6 summarizes the work.

2 Background and Related Work

In this section, we first introduce the background and related work for LLM watermarking. Then, we compare the capabilities of those watermarking techniques.

2.1 LLM Watermarking

Adding post-hoc watermarks in LLM-generated texts can be methodologically categorized into [39]: (1) Rule-based watermarking, (2) Inference-time Watermarking, and (3) Neural-based Watermarking.

Rule-based watermarking This approach integrates watermarks into LLM-generated texts by manipulating linguistic features [12, 46], altering lexical properties [11], and substituting synonyms [28, 45]. The rule-based watermarking approach aims to insert the synonym replacement or syntactic transformations as watermarks while ensuring the overall semantics are not distorted.

Inference-time Watermarking Inference-time watermarking [17, 18] approach inserts signatures at the LLM decoding stage. This approach divides the vocabulary into red/green lists and only allows LLM to decode tokens from the green list. Some follow-up works [7, 50] proposed different red/green list splitting algorithms or sampling algorithms from the green list probabilistic distribution to enhance the explainability and robustness of the inference-time watermarking.

Neural-based Watermarking A neural based approach [1] encodes LLM-generated texts and associated message signatures through an end-to-end machine learning paradigm. It leverages a data-hiding network to infuse the watermark signatures into the LLM-generated texts and a data-revealing network to decode the signature from the watermark text. This facilitates the signatures to be integrated into the feature space of the watermarked text without compromising the semantic fidelity. However, a notable limitation of current state-of-the-art neural-based approach AWT [1] is the limited embeddable signature length capacity [39].

2.2 Comparison

An ideal text watermarking framework should adhere to the following three criteria:

Criteria 1 Effectiveness: The inserted watermark signatures can be seamlessly extracted.

Criteria 2 Fidelity: The watermarked content quality shall not be compromised. This entails that signature insertion not only preserves the original semantics but also ensures that the text coherence and consistency remain undistorted.

Criteria 3 Efficiency: The watermark insertion and extraction are efficient. This includes both minimal computation and time overheads to ensure rapid IP insertion/verification without excessive computational resources.

Criteria 4 Robustness: Resilience against potential threats is crucial to help LLM proprietors verify IP and trace data sources. Therefore, the signatures shall remain extractable under watermark detection and removal attacks.

Criteria 5 Undetectability: The watermarks are invisible upon inspection. As a result, the adversary cannot detect whether a given text is watermarked.

We systematically evaluate the capabilities of the watermarking frameworks previously against the proposed criteria in Table 1. The rule-based watermarking, like CATER [12], demonstrates effectiveness and efficiency by inserting watermarks in the linguistic attributes of the texts. However, the adversary may exploit syntactic transformations or synonym replacements to remove manually designed watermarking signatures from CATER [12].

The inference-time watermarking achieves resilience by embedding watermarks during each token decoding. However, KGW [18] introduces semantic discrepancies between the watermarked and original texts, undermining the LLM's fidelity. While EXP [19] tries to improve the semantic preservation, the efficiency is compromised compared to KGW [18].

The neural-based approach like AWT [1] leverages machine learning algorithms to embed watermarks into the LLM-generated texts without tampering with textual semantics. It achieves efficiency and robustness by embedding the watermarks through text feature space via lightweight language models.

To take the best properties of the neural-based watermarking, we devise REMARK-LLM as a pioneering LLM-generated text watermarking methodology. It can embed up to 2× longer signature sequences into the same contents compared with the best prior art [1] without compromising the textual semantics and coherence.

Method	Effectiveness	Fidelity	Efficiency	Robustness	Undetectability
Rule-based [12, 46]	✓	✓	✓	✗	✓
Inference-time [18, 19]	✓	✗	✗	✓	✓
Neural-based [1]	✓	✓	✓	✓	✓

Table 1: Comparison of post-hoc LLM-generated text watermarking schemes.

3 Problem Formulation

In this section, we first introduce the watermarking objectives for the LLM-generated texts. Next, we discuss the various challenges of incorporating watermarks into the content and define the threat models that impact watermarked text.

3.1 Watermarking Objective

Given the increasing popularity of LLMs and the heightened prominence of machine-generated media, the dangers associated with the content they produce have become more significant. Content generated by LLMs may inadvertently be employed to create counterfeit essays or inundate the internet with spam responses, thereby posing a threat to the credibility of online content. Recognizing this challenge, REMARK-LLM equips LLM proprietors with robust IP tracing toolsets as shown in Figure 1. The local users submit prompt requests to a remote cloud-hosted LLM API to obtain responses. The LLM inserts watermarks into their generated text response before sending it to local users. An LLM proprietor can trace malicious usages online and claim ownership by applying message decoding modules to extract the signatures. LLM proprietors can claim ownership and prove whether texts are machine-generated using REMARK-LLM by comparing the inserted and extracted signatures.

Applications: The watermarking system has wide applications with the emerging popularity of large language models: (i) Education [20]: Watermarking can help teachers and professors identify whether student homework submissions, like essays or research papers, are AI-generated. This maintains academic integrity and ensures students engage in original thinking and writing. (ii) Copyright Protection [21]: Watermarking can help detect if humans or AI write the given article. It protects the LLM owners' copyrights because the article publishers could make profits from the AI-generated

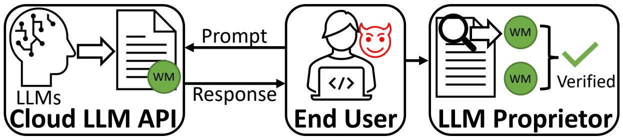


Figure 1: LLM-generated text watermarking scenario. The local user sends prompts to the remote LLM cloud API, and the API watermarks (WM) the responded texts before sending them back to users. LLM proprietor claims ownership by using the message decoding module to decode the signatures and compare them with inserted watermarks.

content without proper acknowledgment. (iii) Misinformation Monitor [23]: Watermarking can be used by social media platforms to detect and label watermarked AI-generated content automatically. It helps to combat the potential spread of misinformation or inauthentic content.

3.2 Challenges

When compared to other methods of watermarking various media [6, 8, 36, 47], the process of embedding watermarks into text data presents distinct challenges, as outlined below.
Challenges 1 Sparsity: A given image with size 256×256 provides over 65,000 potential pixel positions for embedding watermarks [30]. This granularity ensures notable flexibility in watermark accommodation. However, the maximum token length LLMs like GPT-4 can generate for text data is 8.2k [33]. This significantly constrains the potential embedding locations and demands a more sophisticated-designed watermarking approach.
Challenges 2 Sensitivity: Text data exhibits a heightened sensitivity to alterations [51]. Minor image pixel adjustments often remain imperceptible, ensuring the image's aesthetic after watermark embedding. Text data, on the other hand, minor changes can distort the intended meanings and make the text incoherent or misleading.
Challenges 3 Vulnerability: If an adversary suspects or detects watermarking, they might attempt to remove or alter the inserted signatures by rephrasing or editing attacks [17].

3.3 Threat Model

Adversary's Capacity We assume the adversary is an end-user of the LLM cloud service, where he has black access to the API. However, he does not have access to the trained watermarking models and the original LLM-generated outputs. The adversary attempts to exploit the LLM-generated content for malicious usage without being traced. Therefore, he performs attacks to detect and remove the signatures within the watermarked contents without distorting their semantics.

This threat model setting is consistent with prior work AWT [1] and EXP [19] that assume: (i) the watermarking framework is kept private, where adversaries as end-users do not have control over REMARK-LLM's weights and/or hyperparameters; (2) the adversarial attacks do not greatly compromise the generated output quality, accuracy, and readability.
Potential Attacks The adversary performs detection attacks to inspect if the LLM-generated contents have watermark insertion or not. If the adversary suspects or detects watermarks, he performs attacks to remove inserted signatures by directly manipulating the textual content or leveraging sophisticated NLP models for rephrasing.

- ***Attack 1 Watermark Detection Attack***: The adversary uses statistical analysis or machine learning models to detect whether the texts are watermarked.
- ***Attack 2 Text Edit Attack***: The adversary doesn't have prior linguistic knowledge. By randomly deleting, adding, or substituting words within the content, he attempts to destroy the watermark while preserving the overall meanings.
- ***Attack 3 Text Rephrase Attack***: The adversary can exploit open-source NLP models, such as T5, to remove watermarks. By feeding the content into these models, the adversary aims to generate a rephrased version of the original texts to remove the watermark.
- ***Attack 4 Re-watermarking Attack***: The adversary dispatches the watermarked texts into another LLM watermarking framework like REMARK-LLM and re-watermark it to remove the inserted signatures.

4 REMARK-LLM Design

In this section, we first introduce REMARK-LLM's training pipeline as illustrated in Figure 2. Then, with the trained REMARK-LLM, we introduce how coherent watermarked texts are generated from the message encoding module and how message signatures are extracted from the watermarked texts in the message decoding module.

4.1 REMARK-LLM Training

The watermarking framework is depicted in Figure 2, which is a confluence of three major components: **Message Encoding**, **Reparameterization**, and **Message Decoding**. The message encoding inserts invisible watermarks into the LLM-generated texts via a Sequence-to-sequence (Seq2Seq) model. Reparameterization converts the watermarked distribution from the Seq2Seq model towards a more sparse distribution of the watermarked textual tokens using Gumbel-Softmax. Message decoding first maps the reparameterized distribution into their respective embedding representation using a mapper network. It then employs a transformer-based decoder to extract the secret messages from the embedding.
Message Encoding: The message encoding module takes an LLM-generated token sequence, denoted as $T =$

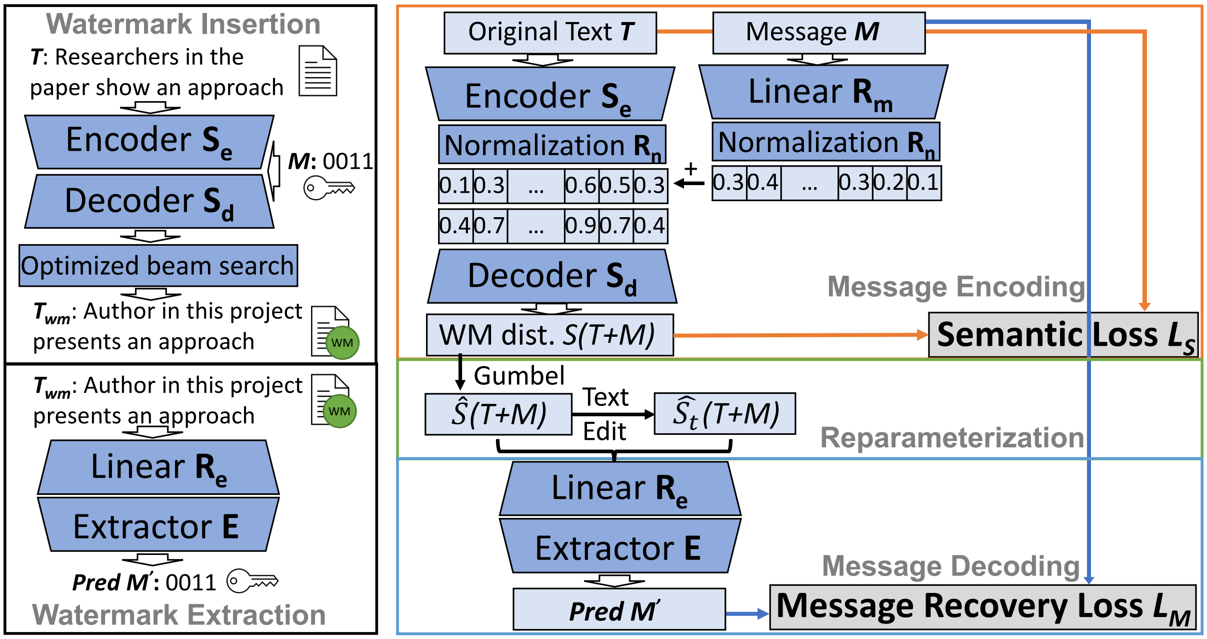


Figure 2: REMARK-LLM's Watermarking Framework. The left is an overview of REMARK-LLM: The message encoding module leverages an optimized beam search algorithm to produce coherent watermarked contents. The message decoding module is designed for efficient watermark extraction. The right is REMARK-LLM's training pipeline: The message encoding, reparametrization, and message decoding modules are trained jointly in an end-to-end fashion, aiming to minimize the semantic loss between original text T and watermarked distribution $\mathcal{S}(T+M)$, as well as minimize the message recovery loss between the inserted message M and predicted message M'.

$\{T_1, T_2, ...T_t\}$, alongside a binary signature sequence M as input. The texts T goes through the encoder $\mathbf{S}_e$ of the Seq2Seq model and acquires its corresponding latent space representation $\mathcal{S}_e(T)$ at the final normalization layer $\mathbf{R}_n$. Concurrently, the message M is encoded by a linear layer $\mathbf{R}_m$ followed by the shared normalization layer $\mathbf{R}_n$ into the same latent space representation as $R_n(M)$. At the latent space, the messages are embedded into every token in T as $\mathcal{S}_e(T+M)$. The embedded latent feature $\mathcal{S}_e(T+M)$ is directed to the Seq2Seq's decoder $\mathbf{S}_d$ to obtain the watermarked distribution over the vocabulary as $\mathcal{S}(T+M)$.

Reparameterization: The message encoding module generates a dense token distribution, whereas the message decoding module extracts messages from the watermarked textual tokens' one-hot encoding. To bridge this gap, the reparameterization module connects them and transforms the dense token distribution into a sparser form while ensuring differentiability. To achieve this transformation, Gumbel-Softmax is applied in Equation 1 to approximate the watermarked distribution $\mathcal{S}(T+M)$ to more sparse encoding, denoted as $\hat{\mathcal{S}}(T+M)$. $\mathcal{S}(T+M)$ is simplified as $\mathcal{S}$. The g_i is the noise i.i.d samples drawn from Gumbel(0,1), $|V|$ is the vocabulary size, and τ is the temperature for sampling. The lower τ is, the closer the reparameterized $\hat{\mathcal{S}}_i$ is to one-hot encoding.

$$\hat{\mathcal{S}}_i = \frac{\exp((\log(\mathcal{S}_i)+g_i)/\tau)}{\sum_{j=1}^{|V|}\exp\big((\log(\mathcal{S}_j)+g_j)/\tau\big)} \quad \text{for } i = 1,\ldots,|T| \qquad (1)$$

Message Decoding: To decode embedded M from the reparameterized distribution $\hat{\mathcal{S}}(T+M)$, REMARK-LLM first maps it into the embedding space using a linear layer $\mathbf{R}_e$, yielding $\mathcal{H}(T+M)$. The $\mathcal{H}(T+M)$ is the watermarked text representation in the embedding space. Then, the transformer-based extractor model $\mathbf{E}$ extracts messages from $\mathcal{H}(T+M)$ as $M' = \mathcal{E}(\mathcal{H}(T+M))$.

REMARK-LLM becomes robust by enforcing the extractor to learn the embeddings of the malicious transformations and decode the same messages M from those transformations as well. The transforms, including randomly dropping, adding, and replacing tokens in the watermarked distribution, are performed over $\hat{\mathcal{S}}(T+M)$ and get their corresponding distribution as $\hat{\mathcal{S}}_t(T+M)$. Similar to $\hat{\mathcal{S}}(T+M)$, the $\hat{\mathcal{S}}_t(T+M)$ is mapped to the embedding space and extracts messages as $M'_t = \mathcal{E}(\mathcal{H}_t(T+M))$.

End-to-end Training: The above modules are trained in an end-to-end manner, with objectives to (1) ensure the semantic similarity of the input text T and the watermarked distribution $\mathcal{S}(T+M)$ and (2) ensure the watermark extraction of the input message M and decoded message M' and M'_t. The first objective is reflected by the semantic loss L_s and the second is reflected by the message recovery loss L_M.

(1) Semantic Loss: REMARK-LLM formulates the semantic loss L_S by minimizing the cross entropy loss between input token T and watermarked text distribution $\mathcal{S}(T+M)$ in Equation 2. To avoid overfitting, in every epoch, the input token sequence T is randomly masked via a mask sequence T_M. T_M is of the same size as T, where 1 means the token is unmasked and 0 means the token is masked. $|V|$ is the size of vocabulary in $\mathcal{S}$ and $|T|$ is the number of tokens in the input text T.

$$L_S(T,\mathcal{S}(T\cdot T_M+M)) = -\frac{1}{|T|}\sum_{i=1}^{|T|}\sum_{j=1}^{|V|} T_{ij}\log(\mathcal{S}_{ij}(T\cdot T_M+M)) \tag{2}$$

(2) Message Recovery Loss: REMARK-LLM measures the message recovery loss L_M between the input message M and decoded message M' from the watermarked distribution using L_1 loss. Similarly, REMARK-LLM also includes the message recovery loss between the signatures decoded from malicious transformation M'_t and input message M. In Equation 3, the two losses are adjusted by the coefficients w_w and w_t.

$$L_M(M,M',M'_t) = w_w \sum_{i=1}^{|M|} |M_i - M'| + w_t \sum_{i=1}^{|M|} |M_i - M'_t| \tag{3}$$

(3) Training: When training the end-to-end framework, we include the above losses together as an objective function in Equation 4. This is reflected in REMARK-LLM, where (i) minimizing $\mathcal{L}_S$ means encouraging the watermarked texts to be semantically close to the input texts, and (ii) minimizing $\mathcal{L}_M$ to ensure the encoded messages can be successfully extracted from the watermarked texts. The w_1 and w_2 are the trade-off coefficients during training.

$$L = w_1 L_S + w_2 L_M \tag{4}$$

4.2 Watermark Insertion

With a trained REMARK-LLM, model proprietors can use our message encoding module to insert watermarks into the LLM-generated contents. The message encoding takes the LLM-generated text $\bar{T}$ and the message $\bar{M}$ as input and generates the watermarked distribution over the vocabulary as $\mathcal{S}(\bar{T}+\bar{M})$. Decoding the distribution $\mathcal{S}(\bar{T}+\bar{M})$ by simply taking its maximum index without considering the overall sentence structure will diminish the text coherence. To overcome this, an optimized beam search algorithm is introduced in Algorithm 1. It aims to ensure coherence while maximizing the watermark extraction rates.

In every decoding step, a Gumbel-Softmax noise with temperature τ is added into the token distribution $\mathcal{S}_i$. Then, the beam search algorithm with beam size B produces B candidate sentences from the perturbed token distribution. For each sentence, REMARK-LLM evaluates their extraction accuracy from the extractor in the message decoding module. Based on the empirical evidence, we find a small beam size B results in readable watermarked texts, whereas the best-accuracy sentence has good watermark extractability The beam search is repeated for K iterations with different temperatures τ_k to obtain more diverse watermarked texts.

Algorithm 1 Optimized Beam Search Algorithm

```
Require: LLM-generated text token $\bar{T}$, temperature list $\tau$,
    beam size $B$, number of iterations $K$, message $\bar{M}$
Ensure: Watermarked text $\bar{T}_{wm}$
Initialize max_accuracy = 0
Initialize $\bar{T}_{wm}$ = None
for $k$ = 1 to $K$ do
    Initialize mask $\bar{T}_M$
    Initialize watermarked dist. $\mathcal{S}(\bar{T}\cdot\bar{T}_M+\bar{M})$
    for each $\mathcal{S}_i$ in $\mathcal{S}(\bar{T}\cdot\bar{T}_M+\bar{M})$ do
        $\mathcal{S}_{\text{noisy},i} \leftarrow \mathcal{S}_i + \text{Gumbel}(\mathcal{S}_i,\tau_k)$
    end for
    $T_k \leftarrow \text{Beam\_Search}(\mathcal{S}_{\text{noisy}}, B)$
    for each $T_{ki}$ in $T_k$ do
        $a \leftarrow \text{Accuracy}(\mathbf{E}(T_{ki}),\bar{M})$
        if $a >$ max_accuracy then
            max_accuracy $\leftarrow a$
            $\bar{T}_{wm} \leftarrow T_{ki}$
        end if
    end for
end for
return $\bar{T}_{wm}$
```

4.3 Watermark Extraction

REMARK-LLM extracts the watermark via the message decoding module. Given the watermarked text $\bar{T}_{wm}$, it is first mapped into the embedding space using $\mathbf{R}_e$. Then, $\mathbf{E}$ extracts the predicted message $\bar{M}'$ from $\bar{T}_{wm}$ and compares it with LLM proprietor inserted watermark $\bar{M}$ to claim ownership.

Watermark Strength The confidence in predicting if watermark signatures reside in the watermarked texts can be evaluated using the z-score. The larger the z-score is, the more robust protection the watermark can provide. Given a message sequence with length $|M|$, $|N|$ bits out of the message can be successfully detected. The message generation is random and follows binomial distributions as in AWT [1], where the probability for generating bit 0 is $p = 0.5$ and bit 1 is $1-p = 0.5$. The mean of the message distribution can be calculated as $\mu = |M| \times p$, and the variance can be calculated as $\sigma^2 = |M| \times p \times (1-p)$. We calculate the z-score of the binominal distribution in Equation 5.

$$z = \frac{|N|-\mu}{\sigma} \tag{5}$$

5 Experiments

In this section, we first introduce the experiment setup. Then, we demonstrate REMARK-LLM's effectiveness, efficiency, and transferability compared with prior arts. Next, we present an ablation study on the effectiveness of each component in REMARK-LLM. Finally, we analyze REMARK-LLM's robustness by evaluating its performance under a spectrum of watermark removal and detection attacks.

5.1 Experiments Setup

Datasets We use four datasets to benchmark the LLM-generated content watermarking performance. The HC3 [10] is the ChatGPT-generated response to questions from QA platforms (e.g., Quora and Stack Overflow). The Human Abstract [31] and ChatGPT Abstract [31] are the research abstracts written by human researchers and their rephrased version by GPT-3.5 Turbo. The WikiText-2 [24] is a collection of paragraphs extracted from verified Good and Featured articles on Wikipedia. The detailed statistics are summarized in Table 2. We randomly split 80% of the texts as training datasets, and the remaining 20% are test datasets for HC3, ChatGPT Abstract, and Human Abstract.

Dataset	Train	Test	Data Source
HC3 [10]	19440	4860	LLM
WikiText-2 [24]	44800	4360	Human
ChatGPT Abstract [31]	8000	2000	LLM
Human Abstract [31]	8000	2000	Human

Table 2: Dataset to benchmark the watermarking performance.

Evaluation Metrics We benchmark **Effectiveness** by the fraction of the inserted watermarks successfully extracted, reflected by the Watermark Extraction Rate. **Fidelity** is measured by (1) BERT-S [49], indicating the watermarked texts should be semantically similar to the original texts, and (2) BLEU-4 [35] that the watermarked texts should be coherent and consistent w.r.t. the original texts. We compute these metrics using Huggingface Evaluate [41] and the measurement details are as follows:

- ***Watermark Extraction Rate (WER)***: the percentage of the binary watermark message successfully extracted.
- ***BERT-S*** [49]: the BERT score cosine distance between the original and watermarked text.
- ***BLEU-4*** [35]: the number of n-grams(n=4) in the watermarked text that match the reference texts.

BLEU-4 is adopted from machine translation to measure the n-gram(n=4) matches between the translated and baseline texts. While the metric has no hard threshold, baselines in prior work [44] indicate BLEU-4 of higher than 0.15 is considered a semantically coherent document transformation.

Efficiency of the watermarking frameworks are measured from two aspects: (1) the time overhead for inserting watermarks and (2) the computation resources required for inserting watermarks, as reflected by the peak GPU memory. Evaluations for **Robustness** and **Undetectability** are in Section 5.4.

Baselines We compare REMARK-LLM with four state-of-the-art LLM watermarking frameworks: CATER [12], KGW [18], EXP [19] and AWT [1]. The rule-based watermarking algorithm CATER [12] inserts conditional watermarks into the LLM-generated texts. It inserts watermarks by choosing words that minimize the distortion of overall word distributions while maximizing the change of conditional word selections. The inference-time watermarking approach KGW [18] inserts watermarks at the LLM decoding step. By dynamically splitting the vocabulary into green/red lists with the message signatures, it enforces the next token prediction to only sample from the green list. We employ the soft red list watermarking algorithm in KGW [18]. The hyperparameters and the prompt methodologies follow the default settings. EXP [19] tries to reduce the semantic distortion by proposing an exponential minimum sampling strategy at the LLM decoding stage. The neural-based watermarking approach AWT watermarks LLM-generated texts in an end-to-end manner. It trains a transformer-based encoder-decoder network that takes an input sentence and a binary message to produce a watermarked text. AWT is trained to preserve the semantics of the watermarked texts while ensuring signature extraction.

For fair comparisons, we compare REMARK-LLM with baselines at the text segment level with a fixed 80 token length following EXP [19] and AWT [1]. We use the long text sequences with a fixed 640 token length as a proof-of-concept showing REMARK-LLM can watermark longer sequences, which exceeds the maximum length studied in prior work. KGW [18] and EXP [19] use OPT-2.7B [48] as the backbone generator to insert watermarks. For CATER [12], we employ their open-sourced synonym tables in follow-up work [13]. For KGW, we follow their paper setting [17] and consider the watermarking to be successful if the z-score of the watermark extraction exceeds 4. For EXP, the z-scores are below 4 after watermarking, and we report the WER providing the same level of p-value strength as EXP in Table 4.

Hyperparameter Settings We include more information on REMARK-LLM's training hyperparameters and architecture details in Appendix B.

5.2 Results

In this subsection, we demonstrate REMARK-LLM's effectiveness and efficiency. The robustness evaluations are in Section 5.4.

5.2.1 Segment-level Watermarking

We summarize the segment-level watermarking performance in Table 3, where the unit segment length is 80 tokens following AWT [1]. For LLM-generated texts, REMARK-LLM and AWT are trained on the HC3 training set and report the performance on the test set of HC3 and ChatGPT Abstract. For human-written texts, REMARK-LLM and AWT are trained

Dataset	Methods	4 bits			8 bits			16 bits		
		WER(%) ↑	BERT-S ↑	BLEU-4 ↑	WER(%) ↑	BERT-S ↑	BLEU-4 ↑	WER(%) ↑	BERT-S ↑	BLEU-4 ↑
HC3	REMARK-LLM	97.01	0.93	0.43	95.59	0.92	0.45	73.46	**0.92**	**0.46**
	AWT [1]	96.32	**0.94**	**0.91**	74.08	**0.96**	**0.84**	50.00	-	-
	KGW [18]	**99.43**	0.62	0.01	**99.43**	0.62	0.01	**99.43**	0.62	0.01
WikiText-2	REMARK-LLM	**97.23**	0.92	0.33	89.57	0.89	0.23	76.37	**0.89**	**0.19**
	AWT [1]	97.04	**0.95**	**0.92**	78.18	**0.96**	**0.92**	50.00	-	-
	KGW [18]	97.07	0.65	0.01	**97.07**	0.65	0.01	**97.07**	0.65	0.01
ChatGPT Abstract	REMARK-LLM	96.98	0.91	0.30	93.53	0.91	0.29	73.80	**0.90**	**0.24**
	AWT [1]	83.81	**0.94**	**0.96**	62.28	**0.96**	**0.83**	50.00	-	-
	KGW [18]	**99.87**	0.63	0.01	**99.87**	0.63	0.01	**99.87**	0.63	0.01
Human Abstract	REMARK-LLM	96.85	0.89	0.26	88.85	0.88	0.20	75.81	**0.84**	**0.10**
	AWT [1]	71.39	**0.95**	**0.95**	63.78	**0.95**	**0.85**	50.00	-	-
	KGW [18]	**99.50**	0.68	0.01	**99.50**	0.68	0.01	**99.50**	0.68	0.01

Table 3: Segment-level watermarking comparison. The length of the segment is 80 tokens. Both REMARK-LLM and AWT are trained on HC3 and WikiText-2's training dataset and report the watermarking performance on the test dataset. The transferability is benchmarked by reporting the test accuracy on ChatGPT Abstract with HC3-trained frameworks and on Human Abstract with WikiText-2-trained frameworks. The best metric values are highlighted in **bold** text, and the second best metric values are underlined. Metric values that are highlighted in red suggest failure cases (low WER or high semantic distortion). A WER of 50% indicates watermark recovery failure. The WER for all unwatermarked texts generated by the original LLM stands at 50%.

on the WikiText-2 training set and report the performance on the test set of WikiText-2 and Human Abstract. For KGW, we use the first 40 tokens as prompts, and the predicted next 80 tokens are as watermarked texts. The inserted signature length is increased from 4-bit to 16-bit. All the original LLM-generated texts' WER are 50%, and we skip them in the table.

Comparison with AWT [1]: (1) REMARK-LLM can insert more signature bits into the same text length without compromising the semantics. The AWT's WER dropped to $\sim$ 70% when inserting 8 bits signatures into the token sequence and failed to insert watermarks larger than 16 bits. However, REMARK-LLM can extract more than 90% of the signature when 8 bits are inserted. (2) AWT demonstrates worse transferability compared with REMARK-LLM. When 4 bits signatures are inserted into 80 tokens, the WER of AWT trained on WikiText-2 drops 25.65% when inference on the Human Abstract dataset and drops 12.51% from HC3 to ChatGPT Abstract dataset. For REMARK-LLM, the WER only drops 0.38% and 0.03% from WikiText-2 to Human Abstract and from HC3 to ChatGPT Abstract, respectively. (3) AWT achieves higher BLEU-4 by replacing words with their synonyms but does not modify the syntactic structure of the sentences. This hinders AWT from being robust under more powerful rephrase attacks, as shown in Section 5.4. By applying masking strategies during REMARK-LLM training, it achieves similar semantic preservation as AWT, but more diverse outputs as reflected by lower BLEU-4.

Comparison with KGW [18]: (1) KGW demonstrates better WER compared with REMARK-LLM by inserting watermarking at the LLM decoding step. However, this is at the cost of compromising its semantics and coherence. (2) KGW's average semantics is dropped by 29% compared with REMARK-LLM and KGW's coherence score BLEU-4 is close to 0. This demonstrates the KGW significantly distorts the semantics of the original LLM-generated texts and adversely compromises the LLM-generated content quality.

5.2.2 Watermarking Long Sequences

The responses local users receive from the LLMs are generally long text sequences with multiple paragraphs. In this section, we investigate how effectively long sequences can be watermarked with the aforementioned frameworks. For the long sequence watermarking, we set the maximum token length to 640 as a proof-of-concept showing REMARK-LLM can watermark longer sequences, which exceeds the maximum length studied in prior work. As AWT performs watermarking at the segment level, we report the watermarking performance by dividing the long sequence into multiple segments and watermarking each segment individually. We stop watermarking later segments if the maximum length is smaller than 640 tokens. We train both AWT and REMARK-LLM on the HC3 dataset and report the inference watermarking performance on all four datasets with the trained framework. The results are reported in Table 4. The WER from unwatermarked content generated by LLMs is 50%. Therefore, we skip it in the table.

Comparison with AWT [1]: (1) AWT successfully preserves the semantics and coherence of the LLM-generated contents. However, the WER drops by an average of 19.52% when the signature length is extended to 64 bits. Thereby, AWT's ability to embed stronger watermarks into the sequence is significantly hindered. (2) Similar to segment level watermarking, AWT exhibits worse transferability compared with REMARK-LLM. When inserting 32-bit signatures, AWT's WER dropped by 14% when watermarking the ChatGPT Abstract dataset using the watermarking model trained on HC3. However, the WER of REMARK-LLM from HC3 to ChatGPT Abstract showcased no accuracy drop.

Comparison with KGW [18]: (1) The semantic distortion

Dataset	Methods	16 bits			32 bits			64 bits		
		WER(%) ↑	BERT-S ↑	BLEU-4 ↑	WER(%) ↑	BERT-S ↑	BLEU-4 ↑	WER(%) ↑	BERT-S ↑	BLEU-4 ↑
HC3	REMARK-LLM	98.93	0.94	0.45	97.84	0.92	0.41	95.61	0.91	0.41
	AWT [1]	96.12	**0.98**	**0.95**	94.51	**0.98**	**0.93**	68.42	**0.95**	**0.82**
	KGW [18]	**99.57**	0.58	0.01	**99.57**	0.58	0.01	**99.57**	0.58	0.01
	EXP [19]	79.37	0.80	0.01	70.62	0.80	0.01	64.68	0.80	0.01
	CATER [12]	75.20	0.96	0.63	75.20	0.96	0.63	75.20	0.96	0.63
WikiText-2	REMARK-LLM	99.02	0.92	0.32	98.60	0.86	0.18	94.48	0.85	0.16
	AWT [1]	89.72	**0.97**	**0.95**	85.82	**0.95**	**0.93**	65.77	**0.96**	**0.85**
	KGW [18]	**99.13**	0.61	0.02	**99.13**	0.61	0.02	**99.13**	0.61	0.02
	EXP [19]	79.37	0.82	0.01	70.62	0.82	0.01	64.68	0.82	0.01
	CATER [12]	53.10	0.94	0.75	53.10	0.94	0.75	53.10	0.94	0.75
ChatGPT Abstract	REMARK-LLM	98.24	0.92	0.33	98.55	0.90	0.27	95.04	0.89	0.27
	AWT [1]	88.26	**0.96**	**0.95**	80.62	**0.97**	**0.94**	62.39	**0.95**	**0.84**
	KGW [18]	**99.01**	0.61	0.01	**99.01**	0.61	0.01	**99.01**	0.61	0.01
	EXP [19]	79.37	0.80	0.01	70.62	0.80	0.01	64.68	0.80	0.01
	CATER [12]	75.50	0.93	0.64	75.50	0.93	0.64	75.50	0.93	0.64
Human Abstract	REMARK-LLM	98.56	0.91	0.31	98.71	0.88	0.16	95.39	0.87	0.15
	AWT [1]	86.43	**0.96**	**0.93**	77.21	**0.98**	**0.92**	63.52	**0.94**	**0.85**
	KGW [18]	**98.79**	0.69	0.01	**98.79**	0.69	0.01	**98.79**	0.69	0.01
	EXP [19]	79.37	0.81	0.01	70.62	0.81	0.01	64.68	0.81	0.01
	CATER [12]	82.00	0.95	0.54	82.00	0.95	0.54	82.00	0.95	0.54

Table 4: Long text sequence watermarking comparison. The length of the sequence is 640 tokens. The frameworks are trained on HC3's training dataset with the watermarking performance reported on the test dataset. The transferability is benchmarked by reporting the test accuracy on the rest of the datasets with the trained frameworks. The best metric values are highlighted in **bold** text, and the second best metric values are underlined. Metric values that are highlighted in red suggest failure cases (low WER or high semantic distortion). The WER for all unwatermarked texts generated by the original LLM stands at 50%.

introduced by KGW becomes worse for longer text sequence. For the LLM-generated content, KGW's BERT-S drops by 35% compared with REMARK-LLM. This suggests that KGW could not preserve the original content quality generated by LLM, resulting in an ineffective watermark insertion. (2) Besides, the BLEU-4 of KGW is close to 0, which demonstrates KGW failed to maintain the coherence between the original LLM-generated and watermarked texts.

Comparison with EXP [19] (1) While EXP maintains the semantics integrity for inference-time watermarking, the BERT-S and BLEU-4 are averagely 0.07 and 0.25 lower compared with 64-bit REMARK-LLM. This indicates neural-based REMARK-LLM preserve better semantics and coherence compared with EXP for longer text sequences. (2) EXP preserves the semantic at the cost of weaker watermark insertion. The average p-value EXP inserted into the watermarked texts is 9.9×10^{-3} and equals an average z-score of 2.36. However, the average z-score REMARK-LLM can provide at 64-bit signature is 7.12. Therefore, REMARK-LLM demonstrates better semantic preservation and stronger watermark insertion than EXP.

Comparison with CATER [12]: CATER achieves high semantic preservation by replacing words with their synonyms. However, such replacements are not generalizable toward new datasets because the candidate words can have high frequency on certain datasets but low frequency on the rest. As in Table 4, CATER has higher WER on HC3 and ChatGPT/Human Abstract datasets but low WER ($\sim 50\%$) on the WikiText-2 dataset. Note that the WERs of CATER are still 13% lower than REMARK-LLM even on the best-performing Human Abstract dataset.

5.2.3 Watermarking Strength

The watermarking strength, measured by z-score, quantitatively evaluates the statistical significance and robustness of the watermarks embedded within the content generated by the LLM. Based on Null Hypothesis [3], a z-score threshold of 1.64 corresponds to a p-value of less than 0.05, implying a significant presence of the watermarks in the content. Following KGW [18], a higher z-score of 4 ($p = 3\times10^{-5}$) indicates a greater alignment between the inserted and extracted signature bits. This alignment serves as stronger evidence that the watermarks belong to the owner of the LLM. We use the z-score from Section 4.3 to compute the watermarking strength for neural-based approach REMARK-LLM and AWT. The results are reported on the ChatGPT dataset with watermarking frameworks trained on HC3. The z-score of KGW [1] and EXP [2] is calculated using their original implementation.

From Figure 3, we find that (1) AWT, EXP, and CATER preserve the semantic integrity but fail to provide sufficient watermark strength for long text sequences. This makes these two approaches less suitable for real-world text ownership proof, where the adversary may argue the watermark insertion is coincidental. (2) While KGW effectively embeds watermarks to LLM-generated content, there is a noticeable

[1] https://github.com/jwkirchenbauer/lm-watermarking
[2] https://github.com/jthickstun/watermark

semantic distortion and greatly degrading LLM quality. (3) REMARK-LLM demonstrates the merits of the aforementioned approaches and ensures both semantic fidelity and watermarking strength. The average z-score of REMARK-LLM is 7.12, corresponding to an average one-side p-value of 5.4×10^{-13}. This provides adequate watermarking proof for a text sequence with 640 tokens. When operating LLMs like GPT-4 at their zenith, the generated contents will be a maximum of 8.2k tokens. REMARK-LLM can provide a z-score of 25.20, corresponding to a one-tail p-value of smaller than 1.59×10^{-130}, all while maintaining semantic integrity. As a result, REMARK-LLM gives ample watermark strength to LLM proprietors.

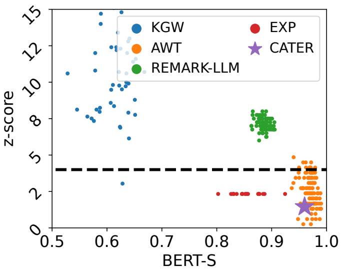


Figure 3: Watermarking strength and semantic preservation comparison of different watermarking frameworks. The threshold for a strong watermark insertion is a z-score of 4, represented as the black dotted line.

5.2.4 Watermarking Efficiency

We measure the efficiency of inserting 8-bit watermarks into 100 samples of 80-token LLM-generated texts by reporting the average wall-clock time required in seconds and peak GPU memory utilization in GB. These results are summarized in Table 5. For REMARK-LLM and AWT, we report the inference time for the watermarking framework to generate the watermarked texts. For KGW and EXP, we report the time difference between the inference time with and without watermark insertion.

The results indicate that inserting the same number of bit signatures into the LLM-generated texts, KGW requires nearly 2 × more time compared with neural-based REMARK-LLM and AWT. The additional overhead stems from the green/red list splitting over the vocabulary for every next token prediction. For EXP, which aims to improve the inference-time watermarking's semantic preservation, the time overhead becomes even larger, which takes 14× longer than REMARK-LLM to watermark the same texts. This greatly constraints such approaches from being widely applied in real-world LLM APIs.

For the computation resources, KGW and EXP require 2.1× more GPU memory compared with REMARK-LLM and 4.5× compared with AWT. The required computation resources for neural-based REMARK-LLM and AWT are agnostic to the LLM type, whereas KGW and EXP require more GPU memory as LLM size grows. Therefore, the neural-based approaches are more computationally efficient compared with inference-time frameworks. For the rule-based CATER, the textual transformations are performed on the CPU, and the GPU memory consumption is 0.

Methods	Time (s)	Memory(GB)
REMARK-LLM	1.21	5.83
AWT [1]	1.14	2.72
KGW [18]	2.36	12.21
EXP [19]	17.50	12.21
CATER [12]	1.02	0

Table 5: Efficiency (time and memory) comparison among watermarking frameworks.

5.2.5 Watermarking Integrity

We evaluate REMARK-LLM's watermarking integrity by running its message decoding module on non-watermarked texts. Then, we calculate the WERs by comparing the decoded signature with the randomly generated encoding signature. We benchmark the WERs on the four datasets' test set and report the results in Table 6. The close to 50% WER on non-watermarked texts demonstrates REMARK-LLM's integrity.

Datasets	HC3	WikiText-2	ChatGPT Abstract	Human Abstract
WER	49.9%	49.9%	50.0%	50.1%

Table 6: Watermarking integrity on different datasets.

5.2.6 Watermarking Different LLM Architectures

We have shown REMARK-LLM trained on the ChatGPT-generated HC3 dataset can successfully watermark human written texts and GPT-3.5 Turbo-generated ChatGPT Abstract in Section 3. We also investigate whether trained REMARK-LLM can watermark unseen texts generated by other LLM families in Table 8. We use 2k instruction prompts from Alpaca Dataset [40] and watermark responses from three state-of-the-art open-source LLMs: OPT [48], OpenOrca [27], and LLaMA-2 [43]. Those models are trained with different architectures and different training datasets. The prompts for generating those responses do not overlap with REMARK-LLM's training dataset HC3.

The watermarking is performed for long text sequences by using the instruction prompts as input and limiting the LLMs to predict the next 640 tokens. REMARK-LLM insert 64 bits into their generated texts and report the watermarking performance in Table 8. We find REMARK-LLM can successfully insert and extract watermarks from texts generated by different LLM architectures. REMARK-LLM also keeps high-quality semantic preservation and coherence. Our results indicate that once REMARK-LLM is trained on the large LLM-generated corpus like HC3, the trained REMARK-LLM is agnostic to the LLM architectures and data sources. REMARK-LLM can thereby be used in real-world LLM watermarking scenarios.

5.2.7 Watermarking Examples

We present the original LLM-generated and watermarked texts in Table 7 and highlight their differences. The first two

Original Text	Watermarked Text
It can be hard to explain it in simple terms. But I'll do my best! During inflation, the universe expands at an incredibly fast rate. But it's important to note that this expansion is not like the movement of objects through space.	It can be hard directly explain exactly in simple terms. But I'll try my best! During time, the universe expands at an infinite incredibly fast rate. But it's important to understand that this expansion is not about the movement of objects through space itself.
In the context of financial investments, "headwinds" refer to negative factors that can potentially hinder the performance of an investment. These may include economic conditions, regulatory changes, market trends, or other external factors that can work against the investment.	In the context of stocks investing, "headwinds" refer for negative factors that can potentially impact the value of an investment. These factors include economic impacts, regulatory issues, market conditions, and other external factors that work against the investment.
The paper discusses Colombeau's generalized function on arbitrary manifolds. We first define the space of Colombeau's generalized functions by quotienting out by a suitable ideal endowed with a ring structure.	The paper introduced Colombeau's generalization function on arbitrary manifolds. We first study the space of Colombeau's generalized functions by particle out ideal endowed with a ring structure.
This paper presents a novel methodology for constructing super throats using non trivial scalar fields. By introducing these fields, we are able to achieve unprecedented control over the dynamics of the throats.	This research presents a novel approach for constructing super throats through non trivial scalar fields. By utilizing these fields, we are able to obtain precise control over the dynamics of the throats.

Table 7: Watermarked Text Examples. All of the watermarked texts achieve 100% WER. The first two examples are randomly taken from the HC3 test set, and the last two are randomly taken from the ChatGPT Abstract test set. The edited words are highlighted in yellow.

LLMs	WER(%)	BERT-S	BLEU-4
OPT-2.7B [48]	93.42	0.91	0.34
OpenOrca-7B [27]	93.70	0.92	0.35
LLaMA-2-7B [43]	91.18	0.91	0.39

Table 8: REMARK-LLM performance in watermarking texts generated by different LLM architectures.

texts are taken from the HC3 test set, and the last two are taken from the ChatGPT Abstract test set. All texts are embedded with 8-bit message signatures using REMARK-LLM trained on HC3's training set at the segment level. For all the watermarked messages presented in Table 7, the watermarks are successfully extracted with 100% WER. We include more watermarking examples in the Appendix A.

From Table 7, we find that the watermarked texts are readable and possess the same semantics as the original texts. REMARK-LLM not only learns to replace the word with their synonyms but also adds/deletes words or replaces other non-synonyms to ensure coherence and preserve the semantic structure. For example, in the first example, "itself" is added after "space"; in the second example, "may" is replaced by "factors"; and in the third example, "by a suitable" is deleted. Those changes do not affect the overall quality of the watermarked texts but help to accommodate longer watermark signature sequences.

5.3 Ablation Studies

In this section, we study how different hyperparameter choices affect the performance of REMARK-LLM during both training and inference time.

5.3.1 Training-time Ablation Study

In this section, we study how different training coefficients affect the watermarking performance. The results in this section are all trained on the HC3 dataset with 8 bits inserted in an 80-token segment. The inference hyperparameters follow the default settings from Appendix B and are reported on the ChatGPT Abstract dataset to avoid overfitting.

Effectiveness of Watermark Backbones We use the default training settings but change the Seq2Seq Model's backbone from T5-base to T5-small and T5-large. For the two replaced models, T5-small has 8 attention heads, 6 layers, and 512 feedforward dimensions, whereas T5-large has 16 attention heads, 24 layers, and 1024 feedforward dimensions.

From Table 9, as the backbone Seq2Seq model becomes larger, REMARK-LLM gets better text coherence and better watermark extraction rates. This is because larger backbone models have more parameters to learn how to add watermarks in the feature space and achieve better performance. This improvement is significant when the model is switched from T5-small to T5-base, and is marginal when the model is changed from T5-base to T5-large.

Backbone	WER(%)	BERT-S	BLEU-4
T5-small	91.60	0.89	0.25
T5-base	93.53	0.91	0.29
T5-large	93.77	0.91	0.32

Table 9: The effectiveness of different model backbones in REMARK-LLM performance.

Effectiveness of Masking Percentages and Gumbel Temperatures We compare how different masking percentages and the Gumbel-Softmax temperatures affect the overall text coherence and watermark extraction accuracy. The results are summarized in Table 10, where we change the input masking percentage from 30% to 70% and the Gumbel-Softmax temperatures from 0.1 to 0.5.

We find that: (1) As the masking percentage becomes larger, at the same Gumbel-Softmax temperature, the coherence between LLM-generated texts and the original texts worsens, whereas the watermarking extraction rates become higher.

Given that REMARK-LLM has more masked space, it introduces more alterations to the original texts, thereby can accommodate more signatures. (2) As the Gumbel-Softmax temperature becomes larger, at the same masking percentage, the coherence of the texts becomes worse, but more watermark signature bits are extracted from the watermarked texts. However, if the temperature rises to 0.5, REMARK-LLM fails to learn the one-hot encoding from Seq2Seq output and achieves worse watermark extractions. Therefore, we trade off the semantic preservation and the watermarking extraction rates during REMARK-LLM training. By default, we choose a masking percentage of 50% and a Gumbel-Softmax temperature of 0.3.

Masking	Temp	WER(%)	BERT-S	BLEU-4
30%	0.1	89.78	0.91	0.34
	0.3	89.97	0.91	0.31
	0.5	83.09	0.91	0.31
50%	0.1	91.18	0.91	0.31
	0.3	93.53	0.91	0.29
	0.5	85.43	0.90	0.29
70%	0.1	94.74	0.89	0.26
	0.3	94.98	0.87	0.17
	0.5	87.30	0.86	0.17

Table 10: The effectiveness of different masking percentages and gumbel noises in REMARK-LLM performance.

Effectiveness of Loss Coefficients We include the watermarking performance under different loss coefficients in Table 11. The w_1 semantic loss coefficient increases from 0.3 to 0.7, whereas the w_2 message recovery loss coefficient decreases from 0.7 to 0.3. From here, we find that larger w_1 indicates better semantic coherence between the original and watermarked texts and a larger w_2 results in higher watermark extraction rates. Therefore, we choose $w_1 = 0.5$ and $w_2 = 0.5$ to obtain a balance between the text semantic coherence and watermark extractions.

w_1	w_2	WER(%)	BERT-S	BLEU-4
0.3	0.7	94.78	0.89	0.25
0.5	0.5	93.53	0.91	0.29
0.7	0.3	91.76	0.92	0.33

Table 11: The effectiveness of different loss coefficients in REMARK-LLM performance.

5.3.2 Inference-time Ablation Study

In this section, we study how different coefficients in the inference time affect the REMARK-LLM performance. The training hyperparameters follow the default settings from Appendix B. We change the token masking percentage in Algorithm 1. The results are summarized in Table 12, where we change the masking input percentage from 0.3 to 0.7.

From here, we find that as the masking percentage grows, REMARK-LLM tends to get better WER. However, the coherence and semantic preservation of the texts are compromised, as reflected by the lower BLEU-4 and BERT-S. Therefore, we choose the masking percentage during inference to be 50% to ensure semantic preservation as well as high watermark extraction rates.

Masking	WER(%)	BERT-S	BLEU-4
30%	91.73	0.93	0.45
50%	93.53	0.91	0.29
70%	96.22	0.86	0.12

Table 12: The effectiveness of different inference hyperparameters in REMARK-LLM performance.

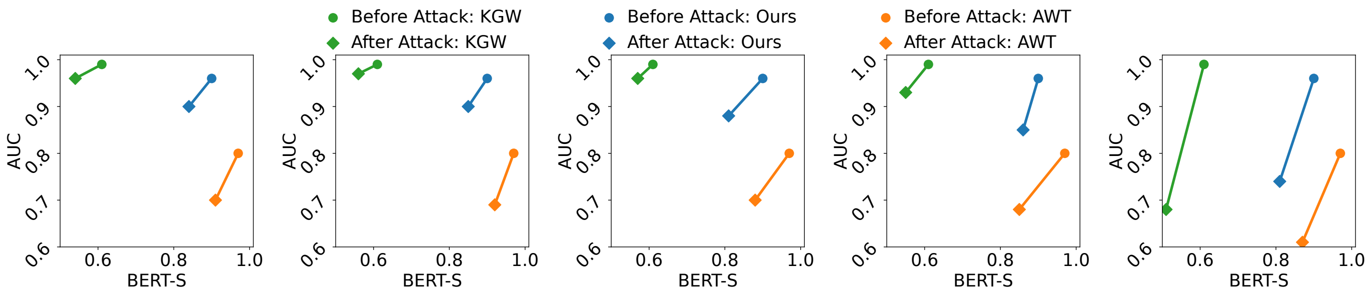
5.4 Attack Evaluations

In this section, we evaluate the watermarking frameworks' **Robustness** under watermark removal attacks and **Undetectability** under watermark detection attacks.

5.4.1 Watermark Removal Attacks

In this section, we demonstrate the resiliency of REMARK-LLM under watermark removal attacks. The adversary attempts to remove the signatures in the watermarked texts by performing the following attack: (1) **Text Deletion Attack**: randomly delete words in the watermarked texts with a probability of 6%; (2) **Text Addition Attack**: randomly add words in the watermarked texts with a probability of 6%; (3) **Text Replacement Attack**: randomly replace words with their synonyms by leveraging Word2Vec [25] pre-trained on Google News corpus [26] (3 billion running words); (4) **Text Rephrase Attack**: using open-source NLP tool T5-large [37] to rephrase the watermarked texts with prompt “paraphrase the texts:" and accept the rephrase if the BERT-S between the watermarked and rephrased sentence is higher than 0.85; and (5) **Re-watermark Attack**: training a local version of REMARK-LLM with WikiText-2 dataset, and re-watermark the texts with new signatures. We report REMARK-LLM's performance under the aforementioned attacks using REMARK-LLM trained on the HC3 dataset and report the attack performance on the ChatGPT dataset.

The adversary's goal is to remove the signature and maintain the semantics and readability of the resultant texts. Therefore, we do not consider the emojis or special characters attack proposed in KGW [18], which adds additional symbols into the watermarked contents to compromise the overall text readability. Furthermore, the adversary uses LLM-generated texts to reduce the time for batch-generating spam content. Therefore, we do not consider the scenario where extra human labor is involved in the attack to rewrite the texts for watermark removal, which deviates from the adversary's initial objectives.



(a) Text Deletion Attack (b) Text Addition Attack (c) Text Replacement Attack (d) Text Rephrase Attack (e) Re-watermark Attack

Figure 4: Watermarking performance under different attacks, including watermark extraction measured by AUC and semantic coherence measured by BERT-S. The attacks are performed on the ChatGPT Abstract dataset with frameworks trained on the HC3 dataset. KGW [18] is the inference-time watermarking framework. AWT [1] is the neural-based watermark framework. From left to right, we study text edit attacks (deletion, addition, and replacement), text rephrase attacks and re-watermark attacks.

Text Deletion Attack From Figure 4a, we find that (1) REMARK-LLM achieves an AUC of 0.90 under Text Deletion Attack. This demonstrates that 90% of watermarking and non-watermarking texts can be successfully classified, providing sufficient ownership proof for LLM proprietors. (2) While KGW achieves 0.96 AUC, the semantic preservation evaluated by BERT-S even before the attack is 0.61. Therefore, KGW failed to provide effective watermarking to LLM-generated content. (3) AWT successfully maintains the semantics with BERT-S larger than 0.90, but the AUC drops to 0.70, demonstrating worse resilience towards text deletion attacks.

Text Addition Attack From Figure 4b, we find (1) REMARK-LLM maintains high watermark extraction rates, yielding an average AUC of 0.90 and average BERT-S of 0.90 before attack. (2) Similar to the text deletion attack, AWT's average AUC degrades to 0.69, which is 23% lower compared with REMARK-LLM and exhibits worse resilience toward text addition attack. (3) While KGW also demonstrates an average AUC of 0.97 after attacks, it fails to maintain semantic coherence in generated texts before and after attacks. This indicates that by including malicious transformations during training, REMARK-LLM becomes robust to unseen threats and maintains semantic coherence during watermarking.

Text Replacement Attack The replacement attack poses a more significant threat to watermark extraction than the prior two approaches, as depicted in Figure 4c. (1) Firstly, AWT and REMARK-LLM watermarking extraction relies on combinations of textual tokens to decode the inserted messages from their respective feature space. By replacing words with synonyms, some of the features used for watermark extraction are compromised, thus resulting in worse AUC. However, the replacement attack does come with a cost of 4-5% BERT-S degradations. (2) REMARK-LLM maintains an AUC of 0.88 with a 0.90 BERT-S score, demonstrating its resilience under such attacks. In contrast, AWT's AUC degrades to 0.70, and KGW's BERT-S degrades from 0.61 to 0.57. This demonstrates that REMARK-LLM maintains a good trade-off between robustness and semantic preservation of watermarked content.

Text Rephrase Attack Figure 4d demonstrates that text rephrase attack results in stronger WER degradation compared to text editing attacks. (1) Our findings show that REMARK-LLM remains resistant to rephrase attacks and consistently maintains an AUC of 0.85. This translates to an average z-score of 5.6 for long text sequences, highlighting the robustness of REMARK-LLM in establishing ownership proof. (2) Compared to baselines, KGW's maintains an AUC of 0.92, but its BERT-S drops to 0.61 even before the attack. (3) Conversely, while AWT preserves semantic information before attacks, its AUC drops to 0.68 after the attack.

Re-watermark Attack We highlight re-watermarking performance in Figure 4e. Re-watermarking degrades the WER by rephrasing the text with REMARK-LLM and tries to break the text features used for watermark verification. Therefore, the WER for all three methods significantly degrade. We find that REMARK-LLM maintains an AUC of 0.74, whereas KGW degrades to 0.68 and AWT degrades to 0.61, demonstrating REMARK-LLM's robustness to re-watermarking attacks.

5.4.2 Watermark Detection Attacks

In this section, we include more security evaluations on REMARK-LLM. Instead of removing the watermark, the adversary intends to know if watermarks are presented in specific texts. One of the most straightforward ways is to use the message decoding module to detect if the watermarks can be successfully extracted. However, when the adversary is a malicious end-user, he/she will not have access to REMARK-LLM's trained components. To detect the watermark presence, the adversary uses (1) statistical analysis of watermarked texts and (2) machine-learning models to classify watermarked and non-watermarked texts.

Statistical analysis We show top-word distribution from the original LLM-generated texts and the watermarked texts in

Figure 5. The texts are taken from the ChatGPT abstract dataset with REMARK-LLM train on HC3. From here, we can find the distributions are close, meaning the adversaries cannot distinguish if texts are watermarked or not by simply analyzing the word frequency distributions.

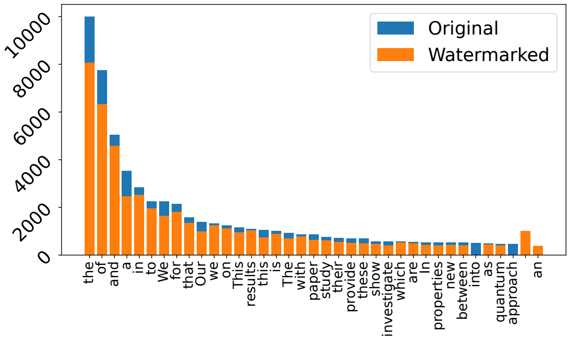


Figure 5: Word frequency distribution of original LLM-generated texts and corresponding watermarked texts.

Machine-learning classification We use machine learning models to classify 2k samples with 1k watermarked and 1k LLM-generated texts. Those texts are non-paired, meaning the watermarked version of the LLM-generated texts is not shown in the training dataset. This is a common setting in which the adversary cannot access original LLM-generated texts. The adversary can only request the LLM API for watermarked texts and search unwatermarked texts online. We use another 500 LLM-generated texts and their corresponding watermarked texts to form the test set. The performance of different machine learning models is shown in Table 13. We train a 3-layer transformer [41] (8 heads with 512 dimensions), a BERT-base [9], and a BERT-large [9] on the training dataset and report the detection performance on the test dataset.

From Table 13, we can find that the accuracy is 50%, and the F1-Score is 0 for all three models. This suggests that machine-learning models cannot confidently classify whether the texts are watermarked or not. Therefore, the watermark detection attacks fail to detect whether the LLM-generated contents are watermarked or not.

Model	Acc. (%)	F1-Score
Transformer [41]	50.00	0
BERT-base [9]	50.00	0
BERT-large [9]	50.00	0

Table 13: Classification performance on watermarked and non-watermarked texts. This shows the watermark detection attacks failed to detect whether the LLM-generated contents are watermarked.

5.5 Overall Evaluations

In the previous experiments, we evaluate different watermarking frameworks' performance in terms of (i) effectiveness, (ii) fidelity, (iii) efficiency, (iv) robustness, and (v) undetectability. Their corresponding results are summarized in Figure 6. The effectiveness is measured by WER, fidelity is measured by BERT-S, and the robustness is measured by the average AUC after removal attacks on the ChatGPT dataset as in Table 4 and Figure 4. Undetectability is measured by 1-F1-score with ML-based classification and efficiency is the normalization of average WM insertion time in Table 5. Higher numbers are desirable for all metrics. We can observe that prior works demonstrate very sensitive trade-offs among all the criteria. However, REMARK-LLM maintains high performance over all the requirements, making it an ideal toolkit for real-world watermarking applications.

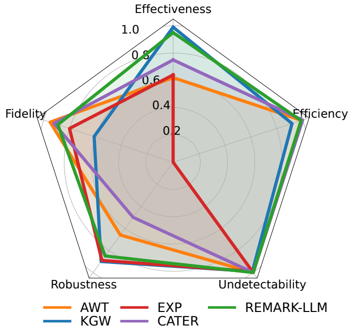


Figure 6: Overall evaluations of different frameworks. REMARK-LLM maintains good performance over all the criteria for watermarking, whereas other baselines demonstrate sensitive trade-offs.

6 Conclusion and Future Work

We present REMARK-LLM, a robust and efficient framework for watermarking text contents generated by LLMs. REMARK-LLM trains message encoding, reparameterization, and message decoding modules jointly, with the aim to accommodate the watermark signatures into the LLM-generated content while ensuring semantic coherence. The message encoding module facilitates the embedding of watermarks into LLM-generated content, while the decoding module extracts messages from these watermarked texts. Comprehensive experiments have demonstrated that REMARK-LLM can embed up to 64 binary bits within 640-token texts without compromising semantics and coherence. Furthermore, REMARK-LLM exhibits transferability in watermarking unseen data sources and LLM architectures without additional fine-tuning and showcases resilience against various watermark detection and removal attacks.

We recommend future work in exploring the transferability of REMARK-LLM towards watermarking more domain-specific data, such as code snippets and medical data, by incorporating additional training objectives. We also recommend future work to investigate adaptive approaches for verifying watermarked contents within the documents.

Acknowledgments

We would like to thank our anonymous shepherd and reviewers for their valuable comments and feedback.

References

[1] Sahar Abdelnabi and Mario Fritz. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. In *42nd IEEE Symposium on Security and Privacy*, 2021.

[2] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[3] David R Anderson, Kenneth P Burnham, and William L Thompson. Null hypothesis testing: problems, prevalence, and an alternative. *The journal of wildlife management*, pages 912–923, 2000.

[4] Rishi Bommasani et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[5] David J Chalmers. Syntactic transformations on distributed representations. *Connectionist Natural Language Processing: Readings from Connection Science*, pages 46–55, 1992.

[6] Huili Chen, Bita Darvish Rouhani, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, pages 105–113, 2019.

[7] Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. *arXiv preprint arXiv:2306.09194*, 2023.

[8] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In *ASPLOS*, pages 485–497, 2019.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arxiv:2301.07597*, 2023.

[11] Xuanli He et al. Protecting intellectual property of language generation apis with lexical watermark. In *AAAI*.

[12] Xuanli He et al. Cater: Intellectual property protection on text generation apis via conditional watermarks. *Advances in Neural Information Processing Systems*, 35:5431–5445, 2022.

[13] Xuanli He, Qiongkai Xu, Lingjuan Lyu, Fangzhao Wu, and Chenguang Wang. Protecting intellectual property of language generation apis with lexical watermark. *arXiv preprint arXiv:2112.02701*, 2021.

[14] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[15] Robin Keskisärkkä. Automatic text simplification via synonym replacement, 2012.

[16] Young-Won Kim, Kyung-Ae Moon, and Il-Seok Oh. A text watermarking algorithm based on word classification and inter-word space statistics. In *ICDAR*, pages 775–779. Citeseer, 2003.

[17] John Kirchenbauer et al. On the reliability of watermarks for large language models. *arXiv preprint arXiv:2306.04634*, 2023.

[18] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. *arXiv preprint arXiv:2301.10226*, 2023.

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

[20] Thomas Lancaster. Artificial intelligence, text generation tools and chatgpt–does digital watermarking offer a solution? *International Journal for Educational Integrity*, 19(1):10, 2023.

[21] Zongjie Li, Chaozheng Wang, Shuai Wang, and Cuiyun Gao. Protecting intellectual property of large language model-based code generation apis via watermarks. In *CCS*, pages 2336–2350, 2023.

[22] Aiwei Liu, Leyi Pan, Xuming Hu, Shu'ang Li, Lijie Wen, Irwin King, and Philip S Yu. A private watermark for large language models. *arXiv preprint arXiv:2307.16230*, 2023.

[23] David Megıas et al. Architecture of a fake news detection system combining digital watermarking, signal processing, and machine learning. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 13(1):33–55, 2022.

[24] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[27] Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4, 2023.

[28] Travis Munyer and Xin Zhong. Deeptextmark: Deep learning based text watermarking for detection of large language model generated text. *arXiv preprint arXiv:2305.05773*, 2023.

[29] Avanika Narayan, Ines Chami, Laurel Orr, Simran Arora, and Christopher Ré. Can foundation models wrangle your data? *arXiv preprint arXiv:2205.09911*, 2022.

[30] Paarth Neekhara et al. Facesigns: semi-fragile neural watermarks for media authentication and countering deepfakes. *arXiv preprint arXiv:2204.01960*, 2022.

[31] Nicolai Thorer Sivesind. Chatgpt-generated-abstracts, 2023.

[32] R OpenAI. Gpt-4 technical report. *arXiv*, pages 2303–08774, 2023.

[33] OpenAI Team. GPT-4. https://openai.com/research/gpt-4.

[34] Laurel J Orr, Karan Goel, and Christopher Ré. Data management opportunities for foundation models. In *CIDR*, 2022.

[35] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318, 2002.

[36] Tong Qiao, Yuyan Ma, Ning Zheng, Hanzhou Wu, Yanli Chen, Ming Xu, and Xiangyang Luo. A novel model watermarking for protecting generative adversarial network. *Computers & Security*, 127:103102, 2023.

[37] Colin Raffel et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[38] John Schulman et al. Chatgpt: Optimizing language models for dialogue. *OpenAI blog*, 2022.

[39] Ruixiang Tang, Yu-Neng Chuang, and Xia Hu. The science of detecting llm-generated texts. *Communications of the ACM*, 2024.

[40] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

[41] Hugging Face Team. Transformers: State-of-the-art natural language processing. In *EMNLP*.

[42] Torch Contributors. PyTorch. https://pytorch.org/, 2023. Last Access on December 26, 2022.

[43] Hugo Touvron et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[44] Lijun Wu, Jinhua Zhu, et al. Machine translation with weakly paired documents. In *EMNLP*, pages 4375–4384, 2019.

[45] Xi Yang, Kejiang Chen, Weiming Zhang, Chang Liu, Yuang Qi, Jie Zhang, Han Fang, and Nenghai Yu. Watermarking text generated by black-box language models. *arXiv preprint arXiv:2305.08883*, 2023.

[46] KiYoon Yoo, Wonhyuk Ahn, Jiho Jang, and Nojun Kwak. Robust multi-bit natural language watermarking through invariant features. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2092–2115, 2023.

[47] Jie Zhang et al. Deep model intellectual property protection via deep watermarking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4005–4020, 2021.

[48] Susan Zhang et al. Opt: Open pre-trained transformer language models, 2022.

[49] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

[50] Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for ai-generated text. *arXiv preprint arXiv:2306.17439*, 2023.

[51] Xuandong Zhao, Yu-Xiang Wang, and Lei Li. Protecting language generation models via invisible watermarking. *arXiv preprint arXiv:2302.03162*, 2023.

Appendix

A Visualization Examples

In this section, we include more watermarked examples generated by REMARK-LLM from different datasets. We first show the non-wateramrked and watermark text pairs in Table 16. Then, we visualize their distributions in the feature space in Figure 8.

More watermarked examples We include more REMARK-LLM's watermarked examples in Table 16, where all of the watermarks are successfully extracted. From here, we can see the watermarked texts are fluent and preserve the semantics of the original texts.

Feature Space Visualization The distribution of the watermarked texts and the original LLM-generated texts at the feature embedding space is shown in Figure 8. We first obtain the texts' embeddings and reduce them to two dimensions using Principal Component Analysis (PCA) [2]. Those reduced dimensions are plotted in Figure 8. From the figure, we can find the watermarked texts and LLM-generated texts are close to each other in the feature embedding space. This ensures the watermarked texts do not alter the meaning of the original texts. Their distribution patterns are different, which is the basis for the message decoding module to learn how to predict message bits from the watermarked texts.

B Experimental Setups

In this section, we first introduce the infrastructure for training REMARK-LLM. Then, we include REMARK-LLM's detailed architecture configurations and the REMARK-LLM's hyperparameters at both training and inference time.

Hardware Infrastructure Our code is implemented using PyTorch [42] version 1.9.0 and open-source transformer models from huggingface [41]. The training and inference of our watermarking models are performed on NVIDIA RTX A6000 GPUs with Ubuntu 20.04.5 LTS and AMD Ryzen Threadripper 3990X 64-Core Processors.

REMARK-LLM Architecture We include REMARK-LLM architecture details in Table 14. Our T5 model [37] is initialized using the official pre-trained checkpoints from huggingface [41]. The rest of the submodule models are trained from scratch. For the transformer-based models, *n_head* is the number of heads for attention modules, N is the number of layers, and *d_model* is the feedforward dimension.

REMARK-LLM Training Details We add additional training details and the hyperparameters in Table 15. The watermark message is encoded vis the L1 loss between the input and extracted message from the Message Decoding module. The watermarked texts ensure their semantic similarity by minimizing the cross-entropy between the original texts and the watermarked distribution from the Message Encoding module. For the text transformation, we set 33% possibility for $\mathcal{S}_t(T+M)$ to be Text Replacement Attacks, 33% possibility to be Text Deletion Attacks, and 34% possibility to be Text Addition Attacks. The loss curve in Figure 7 is the plot for training REMARK-LLM on the ChatGPT Abstract Dataset and inserting different message bits. We can find both the message recovery loss and the semantic loss converge at the end of the training.

SubModule	Backbones	Input Size	Output Size
Message $\mathbf{R}_m$	Linear	ML	512
Mapping $\mathbf{R}_e$	Linear	TL	ED

SubModule	Backbones	*n_head*	N	*d_model*
Seq2Seq $\mathbf{S}_e$	T5 Encoder	12	12	768
Seq2Seq $\mathbf{S}_d$	T5 Decoder	12	12	768
Extractor $\mathbf{E}$	Transformer	8	3	512

Table 14: REMARK-LLM architecture details. ML is short for Message Length, TL is short for Token Length, ED is short for Tokenizer Embedding Dimension

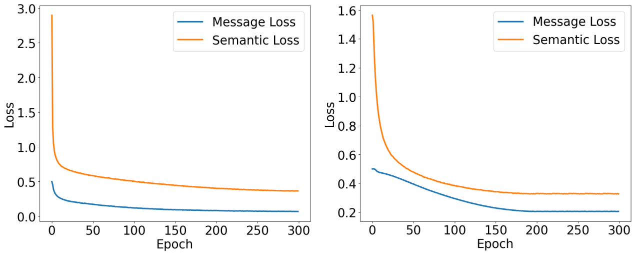


Figure 7: The loss curve for training REMARK-LLM on the HC3 dataset at the segment level and inserting 8-bit and 16-bit messages, respectively. The message recovery loss is L_M and the semantic loss is L_S.

Watermark Insertion and Extraction For the optimized beam search algorithm in the inference time, the detailed hyperparameters for watermark insertion are in Table 15. We choose the texts with the highest watermark extraction accuracy to be the watermarked texts for output. For the watermark extraction, we use the trained $\mathbf{R}_e$ to map the one-hot token distribution to the embedding space and extract the watermarks via the extractor $\mathbf{E}$.

Training-time	Settings
Epoch, Batch size	300, 16
w_w, w_t w_1, w_2	0.7, 0.3, 1, 1
Maximum Token Size	80
Optimizer, Learning rate	AdamW, 3e-5
Gumbel Temperature	0.3
Mask Percentage	50%

Inference-time	Settings
Beam Width, Repeat	5, 5
Gumbel Temperature	{1, 1, 1.5, 1.5, 2}
Mask Percentage	50%

Table 15: REMARK-LLM Hyperparameters during training/inference.

Original Text	Watermarked Text
This means that you may not be able to buy or sell stocks or other investments, pay bills, or make other financial transactions. However, there may be some limited options for you to handle your finances.	This means that you simply not be able to buy and sell stocks , other investments, pay bills, or make any significant transactions. However, there might be some other options available to manage your finances.
LinkedIn is a social networking platform that is primarily used for professional networking. It is a place where people can create a profile, connect with other professionals, and share their professional experiences and skills. LinkedIn is often used by job seekers to find employment opportunities and by employers to find and recruit qualified candidates for job openings. LinkedIn can also be used to connect with industry experts.	LinkedIn is a social networking website that is widely used by business networking. It is a place where people can build a profile, connect with other professionals, and share their professional experiences and knowledge. It is often used by job seekers to find employment, and by businessman to search and recruit qualified candidates for job postings. It can also be used to connect with industry professionals.
Hedge funds are investment funds that use a variety of strategies to try to generate higher returns for their investors. They are usually started by a group of investment professionals who have experience in the financial industry and want to start their own business. To start a hedge fund, the founders typically need to raise money from investors, such as wealthy individuals or institutional investors like pension funds.	Hedge funds are investment companies that use a variety of strategies inside try to generate higher returns on their investors. They are usually started when a group of investors professionals who have experience in the financial sector directly want to start own business. To start a hedge fund, the founders typically need to have money from investors, such as wealthy individuals or companies investors like pension funds.
WEP, WPA, and WPA2 are different types of security protocols that can be used to protect a wireless network. Here's a breakdown of the main advantages and disadvantages of each: WEP (Wired Equivalent Privacy): Advantages: WEP was one of the first security protocols.	WEP, WPA and WPA2 are different types of security protocols can be used to secure wireless network. Here's a simple of some main differences and disadvantages of each: WEP (Wire Equivalent Privacy): Advantages: WEP was one of the first security protocols.
In this paper, we investigate the behavior of the Goldstone modes and the Higgs condensation beyond the one-loop approximation in the context of the Standard Model of particle physics. We show that, even though the one-loop calculation provides a reasonable description of the phenomenon in most cases.	In this paper, we discuss the behavior the Goldstone project in the Higgs condensation beyond the one-loop approximation in the context of super dynamics Model of particle physics. We show that, at the one-loop represents a reasonable explanation of the phenomenon in most cases.
A version of Sonic the Hedgehog was developed by Ancient and released in 1991 for Sega's 8 @-@ bit consoles, the Master System and Game. Its plot and gameplay mechanics are similar to the 16 @-@ bit version, with different level themes and digital assets.	The version of Sonic the Hedgehog was released by Ancient Come in 1991 for Sega's 16- ((bit maps, the original Game. The plot and the mechanics alone similar to the 16– bit console — but different level settings and digital assets.
The artist you are thinking about is Salvador Dal, a prominent Spanish artist who is best known for his unique style and surrealist paintings. Three of his most recognized works are: 1. The Persistence of Memory: Also known as "Soft Watches," this painting was created in 1931. It features melting clocks.	The one you are thinking of is William Dali, a prominent Spanish artist who is best known for his complex and surreist elements. Three of his most famous works include: 1. The Pers of Memory: Also known as "Soft Watches" this painting was created in 1913. It features melting clocks.
Here are four popular social media platforms: 1. Facebook: Facebook is a popular platform for connecting with friends, family, and other acquaintances. Users can share updates, photos, and videos, and engage with others through comments, likes, and shares. 2. Instagram: Instagram is a photo and video-sharing platform where users can share their stories and daily life.	Here are four common social media platforms: Facebook: Facebook is a popular platform for connecting with friends, family, and other connecteds. Users can share updates, tips, and videos, and connect with others through posts, likes, and shares. Instagram: Instagram is a photo sharing and picture-sharing website where users can share their stories and everyday life.
A neural network is an artificial intelligence (AI) model inspired by the human brain and its structure and functionality. It is designed to process information and make predictions or decisions based on the input data. It consists of multiple layers of neurons (or nodes), which are connected by weights and biases.	A neural network, an artificial intelligence (AI) system motivated by the human mind and its complex and function. It is designed to process data and make decisions or predictions based on input data. It consists of multiple layers of layers (called nodes), which perform assigned by weights and biases.
Create a question about the consequences of global warming. The question could be about: - The impact of global warming on individual countries or regions - The effects of global warming on animal and plant species - The influence of global warming on human societies and economies - The role of global warming in exacerbating natural disasters.	Create a question about the consequences of global warming. The following might include: "- the impact of global warming on different countries or regions - The effects of global warming on animal and plant populations - The role of global warming on human health and societies - The role of global warming in exaggerating natural disasters ".

Table 16: More Watermarked Text Examples from REMARK-LLM. The edited words are highlighted in yellow. The first six examples are randomly taken from ChatGPT abstract, Human Abstract, HC3, and WikiText-2 datasets. The last three are randomly taken from Alpaca Datasets instructed LLaMA-2 generated texts.

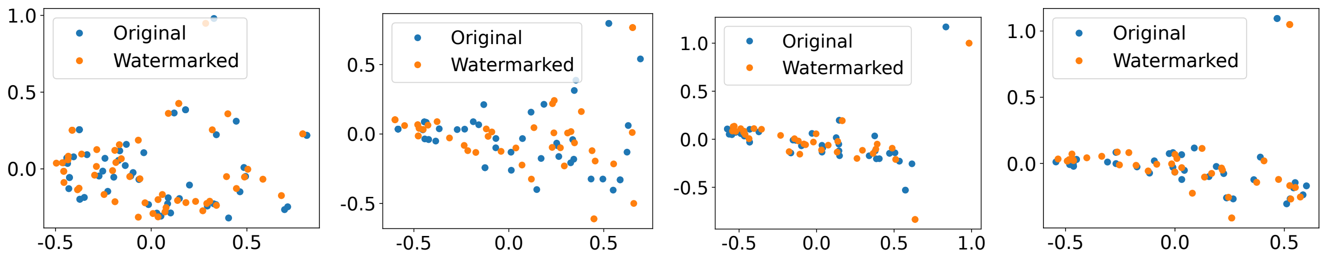


Figure 8: The visualization of the Original LLM-generated texts and the Watermarked texts at the feature embedding level. From left to right, the figures correspond to the examples in Table 7