A Survey of Text Watermarking in the Era of Large Language Models

AIWEI LIU*, Tsinghua University, Beijing, China
LEYI PAN*, Tsinghua University, Beijing, China
YIJIAN LU, The Chinese University of Hong Kong, Hong Kong, Hong Kong
JINGJING LI, The Chinese University of Hong Kong, Hong Kong, Hong Kong
XUMING HU, The Hong Kong University of Science and Technology - Guangzhou Campus, Guangzhou, China
XI ZHANG, Beijing University of Posts and Telecommunications, Beijing, China
LIJIE WEN†, Tsinghua University, Beijing, China
IRWIN KING, The Chinese University of Hong Kong, Hong Kong, Hong Kong
HUI XIONG, The Hong Kong University of Science and Technology - Guangzhou Campus, Guangzhou, China
PHILIP YU, Department of Computer Science, University of Illinois at Chicago, Chicago, United States

Text watermarking algorithms are crucial for protecting the copyright of textual content. Historically, their capabilities and application scenarios were limited. However, recent advancements in large language models (LLMs) have revolutionized these techniques. LLMs not only enhance text watermarking algorithms with their advanced abilities but also create a need for employing these algorithms to protect their own copyrights or prevent potential misuse. This paper conducts a comprehensive survey of the current state of text watermarking technology, covering four main aspects: (1) an overview and comparison of different text watermarking techniques; (2) evaluation methods for text watermarking algorithms, including their detectability, impact on text or LLM quality, robustness under target or untargeted attacks; (3) potential application scenarios for text watermarking technology; (4) current challenges and future directions for text watermarking. This survey aims to provide researchers with a thorough understanding of text watermarking technology in the era of LLM, thereby promoting its further advancement.

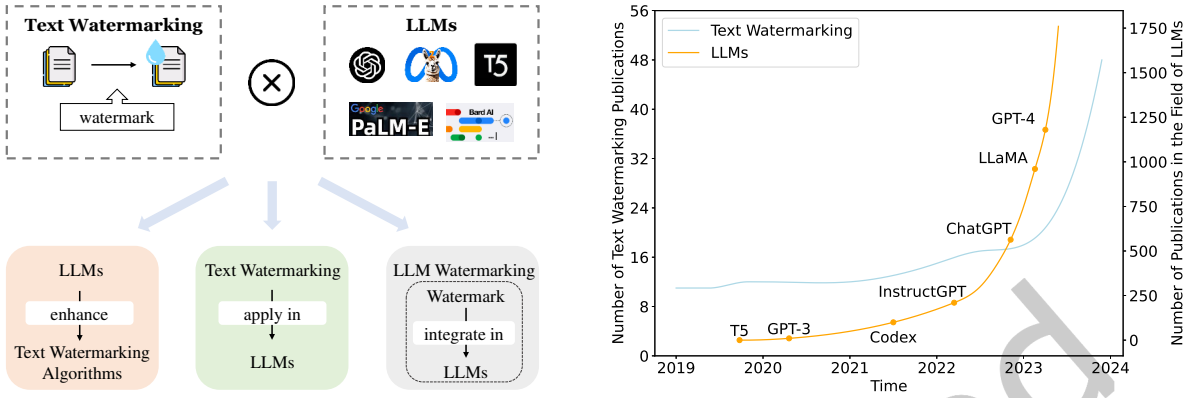
CCS Concepts: • **Security and privacy** → **Digital rights management**; • **Computing methodologies** → **Natural language processing**.

Additional Key Words and Phrases: Text Watermark, Large Language Models, Copyright Protection

*Both authors contributed equally to this research.
†Corresponding author

Authors' Contact Information: Aiwei Liu, Tsinghua University, Beijing, China; e-mail: liuaw20@mails.tsinghua.edu.cn; Leyi Pan, Tsinghua University, Beijing, Beijing, China; e-mail: panly24@mails.tsinghua.edu.cn; Yijian Lu, The Chinese University of Hong Kong, Hong Kong, Hong Kong; e-mail: luyijian@link.cuhk.edu.hk; Jingjing Li, The Chinese University of Hong Kong, Hong Kong, Hong Kong; e-mail: lijj@link.cuhk.edu.hk; Xuming Hu, The Hong Kong University of Science and Technology - Guangzhou Campus, Guangzhou, Guangdong, China; e-mail: xuminghu@hotmail.com; Xi Zhang, Beijing University of Posts and Telecommunications, Beijing, Beijing, China; e-mail: zhangx@bupt.edu.cn; Lijie Wen, Tsinghua University, Beijing, Beijing, China; e-mail: wenlj@tsinghua.edu.cn; Irwin King, The Chinese University of Hong Kong, Hong Kong, Hong Kong; e-mail: king@cse.cuhk.edu.hk; Hui Xiong, The Hong Kong University of Science and Technology - Guangzhou Campus, Guangzhou, Guangdong, China; e-mail: xionghui@hkust-gz.edu.cn; Philip Yu, Department of Computer Science, University of Illinois at Chicago, Chicago, Illinois, United States; e-mail: psyu@uic.edu.

ACM 1557-7341/2024/9-ART
https://doi.org/10.1145/3691626



(a) A description of how LLMs promote the development of text watermarking techniques and broaden their application scenarios.

(b) Number of publications in the field of text watermarking and LLMs (the data for "Number of Publications in the field of LLMs" is sourced from Zhao et al. [123])

Fig. 1. Relationships between the development of text watermarking techniques and LLMs.

1 Introduction

Text watermarking involves embedding unique, imperceptible identifiers (watermarks) into textual content. These watermarks are designed to be robust yet inconspicuous, ensuring that the integrity and ownership of the content are preserved without affecting its readability or meaning. Historically, text watermarking has played a crucial role in various domains, from copyright protection and document authentication to preventing plagiarism and unauthorized content distribution [41]. With the advancement of Large Language Models (LLMs), both the techniques and application scenarios of text watermarking have seen significant development. As shown in Figure 1(a), this primarily includes the construction of enhanced text watermarking algorithms using LLMs, the application of existing text watermarking algorithms to LLMs, and the exploration of LLM watermarking that directly embeds watermarks during text generation. The flourishing development of LLMs has propelled a thriving research landscape within the realm of text watermarking, as depicted in Figure 1(b). Especially with the advent of ChatGPT, text watermarking has notably surged into a research fervor. Specifically, this paper surveys the interplay between LLMs and text watermarking.

1.1 Why is Text Watermarking Beneficial for LLMs?

In recent years, LLMs have made significant progress in the field of natural language processing. As the parameter count of these LLMs continues to increase, their ability to understand and generate language has also substantially improved. Notable examples include GPT [80], BART [50], T5 [82], OPT [120], LaMDA [97], LLaMA [100], and GPT-4 [71]. These LLMs have achieved excellent performance in a variety of downstream tasks, including machine translation [16, 31, 31, 128], dialogue systems [36, 64, 90, 97], code generation [69, 70, 103, 111], and other tasks [51, 52, 96, 121]. A recent work even suggests that GPT-4 is an early (yet still incomplete) version of an artificial general intelligence (AGI) system [10]. However, the utilization of LLMs introduces several challenges:

- **Misuse of LLMs**: LLMs can be exploited by malicious users to create misinformation [13] or harmful content [76] and spread on the internet.
- **Intellectual Property Concerns**: Powerful LLMs are vulnerable to model extraction attacks, where attackers extract large amounts of data to train new LLMs [7].

Adding watermarks to LLM-generated text effectively alleviates these issues. Watermarks enable tracking and detection of LLM-generated text, helping to control potential misuse. Training new LLMs with watermarked text can embed these watermarks, mitigating model extraction attacks.

1.2 Why are LLMs Beneficial for Text Watermarking?

A key challenge in text watermarking is to embed watermarks without distorting the original text's meaning or readability. Traditional methods often fail to modify text without altering its semantics [4, 63, 98]. The necessity for algorithms to comprehend and control text semantics contributes to this difficulty. However, LLMs significantly alter this landscape. Due to their advanced grasp of language semantics and context, they facilitate sophisticated watermarking approaches that embed watermarks with minimal impact on the text's inherent meaning [2, 119]. This integration results in more effective and subtle watermarking techniques, preserving the text's original intent while embedding essential watermark features.

1.3 Why a Survey for Text Watermarking in the Era of LLMs?

Text watermarking technology and LLMs can effectively enhance each other. The interconnection of these two technologies includes the following aspects:

- **Watermarking LLM-Generated Text**: Text generated by LLMs can be watermarked using text watermarking algorithms [9, 68, 79, 85, 114, 115, 117].
- **Embedding Watermarks via LLMs**: LLMs themselves can be utilized to embed watermarks in texts [2, 119].
- **Direct Integration in Text Generation**: Watermark algorithms can be directly incorporated during the text generation process of LLMs [42, 56, 57, 84, 110, 124].

However, comprehensive studies exploring text watermarking in the era of LLMs are lacking. Existing surveys predominantly focus on watermarking techniques developed before the advent of LLMs [3, 41]. In this study, we present the first comprehensive survey of text watermarking algorithms in the context of large language models.

This survey is structured as follows: Section 2 introduces text watermarking definitions and key algorithm properties. Section 3 and Section 4 address two primary text watermarking categories: for existing text and for LLM-generated text. Section 5 discusses evaluation metrics for these algorithms, including detectability, quality impact and robustness under watermark attacks. Section 6 explores application scenarios, namely copyright protection and AI-generated text detection. Section 7 examines ongoing challenges and potential future research avenues in text watermarking. The survey concludes in Section 8.

2 Preliminaries of Text Watermarking

To facilitate the introduction of various text watermarking algorithms as well as its evaluation methods in subsequent sections, this section presents the definition of text watermarking algorithms and outlines the characteristics that an excellent text watermarking algorithm should possess. The taxonomy of text watermarking algorithms is also introduced in this section.

2.1 Text Watermarking Algorithms

A text watermarking algorithm typically comprises two components: a watermark generator $\mathcal{A}$, and a watermark detector $\mathcal{D}$. The watermark generator $\mathcal{A}$ takes a text $\mathbf{x}$ and a watermark message w as inputs and outputs a watermarked text $\mathbf{t}$, expressed as $\mathcal{A}(\mathbf{x}, w) = \mathbf{t}$.

This watermarked text $\mathbf{t}$ is either in a different form but semantically equivalent to the original text $\mathbf{x}$ (§3) or a newly generated text in response to $\mathbf{x}$ (§4), particularly in contexts like prompts for LLMs. The watermark message, denoted as w, can be a zero-bit watermark, signifying merely its presence or absence, or a multi-bit

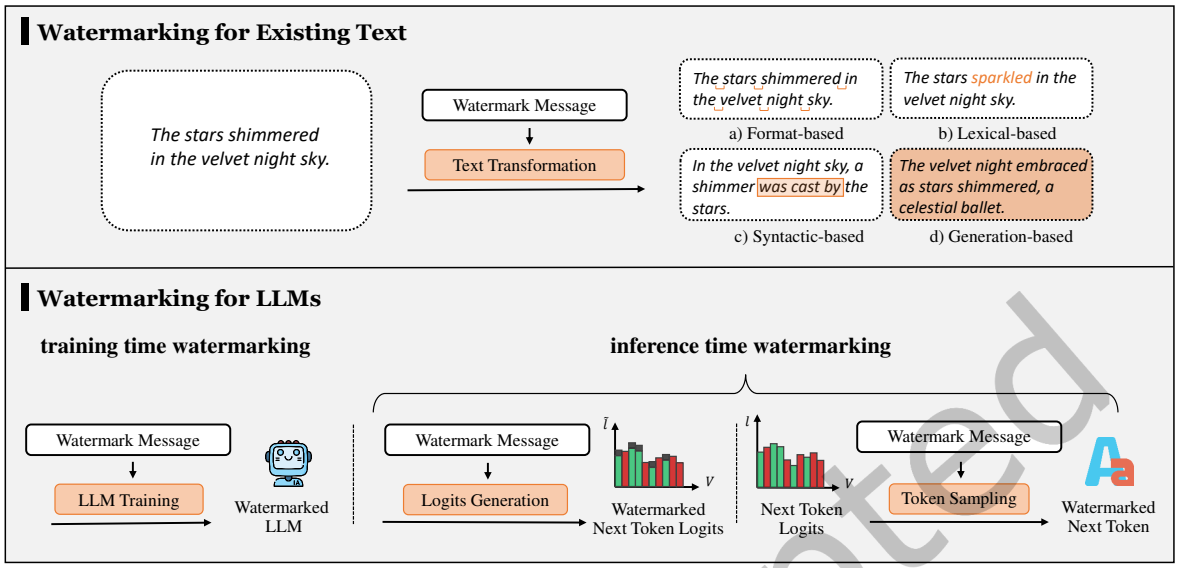


Fig. 2. This figure offers a overview of text watermarking techniques. It categorizes watermarking into two main types: for Existing Text and for LLMs.

watermark, embedding detailed, customized information. The phrase "watermark payload" denote the information volume conveyed by w.

For the watermark detector $\mathcal{D}$, its input is any text $\mathbf{t}$, and its output is its predicted watermark message for the text, denoted as $\mathcal{D}(\mathbf{t}) = w$. If the output is None, it implies that the text contains no watermark information.

2.2 Connection with Related Concepts

To further clarify the scope of text watermarking discussed in this work, this section distinguishes the mentioned text watermarking from other related concepts.

- **Steganography**: Both steganography [87] and text watermarking are important methods of information hiding. While similar, steganography typically requires higher capacity for hidden information, whereas watermarking prioritizes robustness to further text modifications.
- **LLM Watermarking**: The concept of LLM watermarking includes all forms of watermarking added to LLMs, such as their parameters [102], output embeddings [75], and text [42]. This work focuses solely on watermarking applied to the output text of LLMs.

2.3 Key Characteristics of Text Watermarking Algorithms

To further enhance understanding of the text watermarking concept, this section introduces two key characteristics: low impact on text quality and robustness to watermark removal attacks.

Low Impact on Text Quality. The quality of text should not substantially decrease after adding a watermark. Let $\mathcal{A}(\mathbf{x}, \emptyset)$ represent the text generated without a watermark. When $\mathbf{x}$ is the target text (§3), the output remains $\mathbf{x}$. For a prompt given to a LLM (§4), it denotes the LLM's output without a watermark. An effective watermarking algorithm ensures minimal impact on text quality:

$$\forall w_i,\ \mathcal{R}(\mathcal{A}(\mathbf{x}, \emptyset), \mathcal{A}(\mathbf{x}, w_i)) < \delta, \tag{1}$$

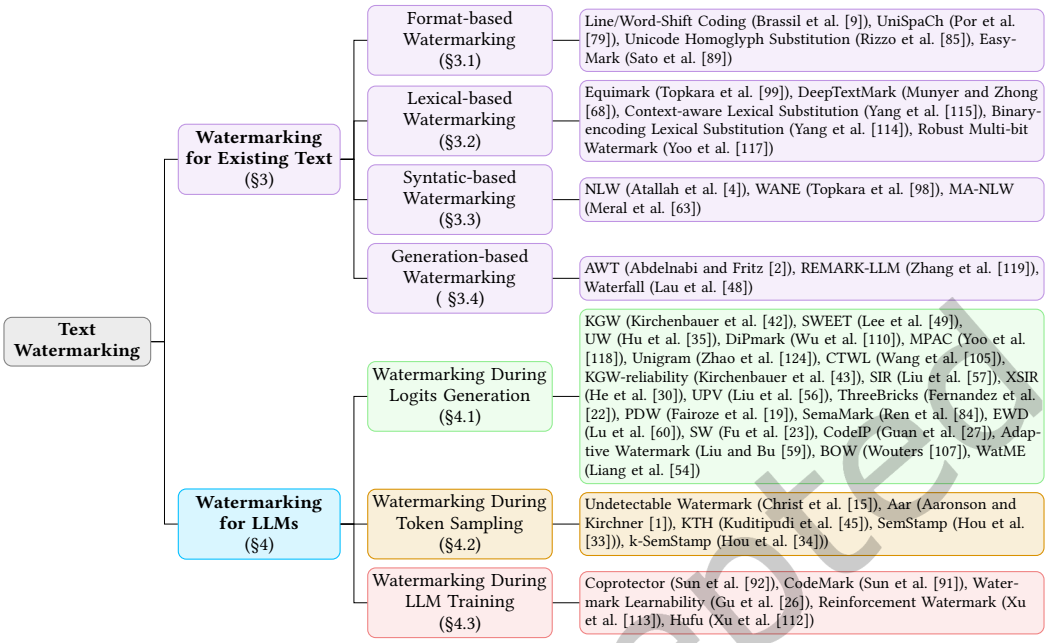


Fig. 3. Taxonomy of text watermarking methods.

where $\mathcal{R}$ is a function evaluating text quality from multiple perspectives, as will be discussed in Section 5. δ represents a threshold. If the difference in the evaluated scores of two texts is less than this threshold, they are considered to be of similar quality.

Robustness to watermark removal attack. For a text watermarking algorithm, it is crucial that the watermarked text can still be detected after some modifications. We use an operation $\mathcal{U}$ to denote the watermark removal operations, which will be detailed in Section 5. If a watermarking algorithm is robust against watermark removal attacks, it should satisfy the following conditions:

$$\forall w_i, \forall \mathbf{t} = \mathcal{A}(\mathbf{x}, w_i),\ P(\mathcal{D}(\mathcal{U}(\mathbf{t})) = w_i) > \beta, \tag{2}$$

where β is a threshold. If the probability of correctly detecting a watermarked text after text modification exceeds β, the algorithm is deemed sufficiently robust.

Additionally, there are other important characteristics of text watermarking algorithms which will be discussed in detail in Section 5.

2.4 Taxonomy of Text Watermarking Algorithms

To facilitate the organization of different text watermarking algorithms in Section 3 and Section 4, this section provides an overview of our summarized taxonomy of text watermarking algorithms. Figure 2 categorizes text watermarking methods into two primary types. The first type, **Watermarking for Existing Text**, embeds watermarks by post-processing pre-existing texts, as elaborated in Section 3. This technique typically utilizes semantically invariant transformations for watermark integration. The second type, **Watermarking for Large Language Models**, involves modifying LLMs, further detailed in Section 4. This method either embeds specific features during LLM training, or alters the inference process, producing watermarked text from the input prompt. Figure 3 presents a more detailed taxonomy of all text watermarking methods.

3 Watermarking for Existing Text

Watermarking for existing text involves modifying a generated text to produce a watermarked text. Based on the granularity of modifications, these methods are primarily categorized into four types: format-based watermarking (§3.1), lexical-based watermarking (§3.2), syntactic-based watermarking (§3.3) and generation-based watermarking (§3.4).

3.1 Format-based Watermarking

Format-based watermarking, inspired by image watermarking [6], changes the text format rather than its content to embed watermarks. For example, Brassil et al. [9] introduced line-shift and word-shift coding by adjusting text lines and words vertically and horizontally. The detection process identifies shifts by measuring distances between text line profiles or word column profiles. However, this method is limited to image-formatted text and does not embed a watermark in the text string.

To address this, Unicode codepoint insertion/replacement methods have emerged. Por et al. [79] developed `UniSpach`, which inserts Unicode space characters in various text spacings. Rizzo et al. [85] introduced a unicode homoglyph substitution method, using visually similar but differently coded text symbols (e.g., U+0043 and U+216d for 'C', U+004c and U+216c for 'L'). Recently, `EasyMark` [89], a family of simple watermarks, was proposed. It includes `WhiteMark`, which replaces a whitespace (U+0020) with another whitespace codepoint (e.g., U+2004); `VariantMark`, which uses Unicode variation selectors for CJK texts; and `PrintMark`, which embeds watermark messages in printed texts using ligatures or slightly different whitespace lengths. The detection process involves searching for specific inserted codepoints.

Though format-based watermarking methods can embed substantial payloads without changing textual content, their format modifications can be noticeable. Por et al. [79] noted the `DASH` attack's ability to highlight these changes. Thus, these methods are vulnerable to removal through canonicalization [8], such as resetting line spacing and replacing specific codepoints. Additionally, these detectable formats may be exploited for watermark forgery, reducing detection efficacy.

3.2 Lexical-based Watermarking

Format-based watermarking approaches, which only modify text's superficial format, are prone to be spotted, making them easily removable through reformatting. This highlights the need for investigating deeper watermark embedding methods in text. A number of studies have adopted word-level modifications, where selected words are replaced with their synonyms without altering the sentence's syntactic structure [68, 99, 114, 115, 117]. These are known as lexical-based watermarking approaches. Topkara et al. [99] introduced a synonym substitution method, employing `WordNet` [21] as the synonym source. The watermark detection replicates the embedding process, applying inverse rules for message extraction. Munyer and Zhong [68] enhanced semantic modeling by using a pretrained `Word2Vec` model, converting selected words into vectors and identifying n-nearest vectors as replacement candidates. They employed a binary classifier with a pretrained `BERT` model and transformer blocks for watermark detection.

The aforementioned watermarking methods relying on context-independent synonym substitution (`WordNet` & `Word2Vec`) often neglect the context of target words, potentially compromising sentence semantics and text quality. To address this, context-aware lexical substitution has been incorporated into text watermarking. Yang et al. [115] introduced a BERT-based infill model for generating contextually appropriate lexical substitutions. The watermark detection algorithm parallels the generation process, identifying watermark-bearing words, generating substitutes, and applying inverse rules for message extraction. Yang et al. [114] streamlined watermark detection by encoding each word with a random binary value and substituting bit-0 words with context-based synonyms representing bit-1. As non-watermarked text adheres to a Bernoulli distribution, altered during watermarking,

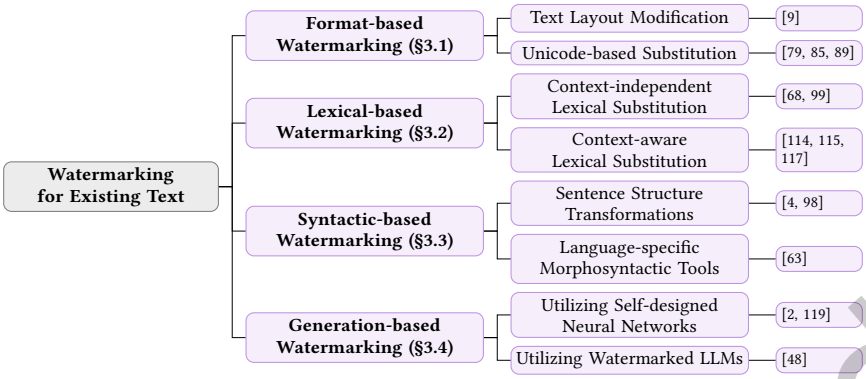


Fig. 4. Taxonomy of watermarking for existing text.

statistical tests can effectively detect watermarks. Yoo et al. [117] enhanced robustness against removal attacks by fine-tuning a BERT-based infill model with keyword-preserving and syntactically invariant corruptions, achieving superior robustness compared to earlier methods.

3.3 Syntactic-based Watermarking

The lexical-based methods aim to embed watermarks by substituting specific words, maintaining the sentence's syntax. Yet, these approaches, relying exclusively on lexical substitution, might not be robust against straightforward watermark removal tactics like random synonym replacement. Consequently, several studies have explored embedding watermarks in a manner that resists removal, notably by modifying the text's syntax structure. These methods are known as syntactic-based watermarking approaches. Atallah et al. [4] introduced three typical syntax transformations–*Adjunct Movement*, *Clefting* and *Passivization*–to embed watermark messages.

Each transformation type is assigned a unique message bit: *Adjunct Movement* to 0, *Clefting* to 1, and *Passivization* to 2. In watermark detection, both original and altered texts are converted into syntax trees, and their structures are compared for message extraction. Expanding this concept, Topkara et al. [98] introduced additional syntax transformations: *Activization* and *Topicalization*. Moreover, research extends beyond English, with Meral et al. [63] analyzing 20 morphosyntactic tools in Turkish, highlighting that languages with significant suffixation and agglutination, such as Turkish, are well-suited for syntactic watermarking.

While syntactic-based watermarking effectively embeds watermarks in a concealed manner, it heavily depends on a language's grammatical rules, often requiring language-specific customization. Frequent syntactic changes in some texts may also alter their original style and fluency.

3.4 Generation-based Watermarking

The aforementioned methods have made significant strides in text watermarking. However, they often rely on specific rules that can lead to unnatural modifications, potentially degrading text quality. If these clues are detected by human attackers, they might design watermark removal attacks or attempt to forge watermarks. A groundbreaking advancement would be generating watermarked text directly from the original text and the watermark message, a technique gradually becoming feasible with the development of pretrained language models.

One approach involves designing neural networks trained to take the original text and the watermark message as inputs and output the watermarked text. Abdelnabi and Fritz [2] developed `AWT`, an end-to-end watermarking scheme using a transformer encoder to encode sentences and merge sentence and message embeddings. This

composite is processed by a transformer decoder to generate watermarked text. For detection, the text undergoes transformer encoder layers to retrieve the secret message. Extending AWT, Zhang et al. [119] addressed disparities between dense watermarked text distributions and sparse one-hot watermark encodings with REMARK-LLM. This method uses a pretrained LLM for watermark insertion and introduces a reparameterization step using Gumbel-Softmax [38] to yield sparser token distributions. A transformer-based decoder extracts messages from these embeddings. REMARK-LLM can embed double the signatures of AWT while maintaining detection efficacy, enhancing watermark payload capacity.

With LLMs' increasing capabilities in following instructions and generating high-quality text, they are becoming viable alternatives to self-designed neural networks for embedding watermarks. Lau et al. [48] introduced WATERFALL, which uses a watermarked LLM to paraphrase the original text, embedding a watermark while preserving semantic content. This approach combines vocabulary permutation with a novel orthogonal watermarking perturbation method to achieve high detectability and robustness. The powerful paraphrasing capabilities of LLMs enhance the naturalness of the generated text, resulting in smoother and more fluent watermarked content.

4 Watermarking for LLMs

While we've explored watermarking for existing text (§3), the rise of LLM-generated content calls for watermarking techniques during the generation process. Watermarking during text generation often yields more natural text, akin to generation-based methods (§3.4). This approach allows LLMs to generate watermarked text directly, which can be defined as:

$$\mathcal{A}(\mathbf{x}, w) = M_w(\mathbf{x}) = \mathbf{t}, \tag{3}$$

where w is the watermark message, x is the prompt, and M_w is a LLM with an embedded watermark. For simplicity, we assume the watermarked text is directly generated by this LLM.

To provide a better understanding of how to add a watermark to a LLM, we first provide an overview of the process used for generating text with an LLM. Specifically, this involves three steps, LLM training, logits generation and token sampling:

- **Step1: LLM Training.** This step involves training a LLM, M, with a dataset D. Training objectives vary based on the application, with next token prediction being the most common [81].
- **Step2: Logits Generation.** With the trained LLM M, given a prompt $\mathbf{x}$ and a sequence of prior tokens $\mathbf{t}^{0:(i-1)}$, the LLM predicts the next token $\mathbf{t}^{(i)}$'s probability distribution in the vocabulary $\mathcal{V}$, expressed as logits $\mathbf{l}^{(i)}$:

$$\mathbf{l}^{(i)} = M(\mathbf{x}, \mathbf{t}^{0:(i-1)}). \tag{4}$$

- **Step3: Token Sampling.** The next token $\mathbf{t}^{(i)}$ is selected from $\mathbf{l}^{(i)}$, using methods like nucleus sampling [32], greedy decoding, or beam search. The sampling process is denoted as:

$$\mathbf{t}^{(i)} = S(\text{softmax}(\mathbf{l}^{(i)})). \tag{5}$$

Through these steps, LLM M generates a token $\mathbf{t}^{(i)}$. For multiple tokens, logits generation and token sampling are iteratively repeated.

In aligning with the three critical phases of text generation using LLMs, watermarking techniques for LLMs are similarly categorized into three distinct types. These are: watermarking during LLM training, during logits generation, and during token sampling. Detailed discussions of these watermarking methods are presented in Section 4.3, 4.1, and 4.2, respectively.

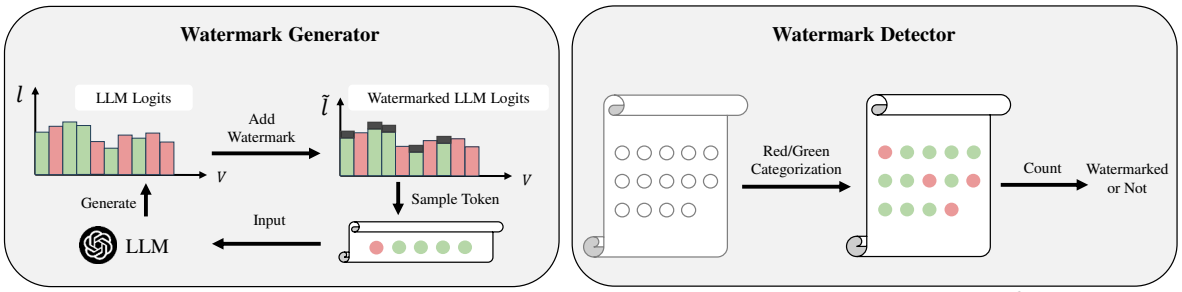


Fig. 5. A more illustrative description of the KGW [42] algorithm.

4.1 Watermarking during Logits Generation

Watermarking during logits generation refers to the insertion of a watermark message w into the logits generated by LLMs. This technique, which does not require modifying the LLM parameters, is more versatile and cost-effective than training time watermarking methods.

In this context, the watermarking algorithm $\mathcal{A}$ alters the logits from the LLM to incorporate the watermark message w. The modified logits $\widetilde{\mathbf{l}^{(i)}}$ can be computed as follows:

$$\widetilde{\mathbf{l}^{(i)}} = \mathcal{A}(M(\mathbf{x}, \mathbf{t}^{0:(i-1)}), w) = M_w(\mathbf{x}, \mathbf{t}^{0:(i-1)}), \tag{6}$$

where $\widetilde{\mathbf{l}^{(i)}}$ is assumed to be produced by a watermarked LLM M_w.

Kirchenbauer et al. [42] introduced the first LLM watermarking technique based on logits modification, termed KGW. This method partitions the vocabulary into a red list (R) and a green list (G) at each token position, using a hash function that depends on the preceding token. For the i^{th} token generation by M_w, a bias δ is applied to the logits of tokens in G. The adjusted logit value, $\widetilde{\mathbf{l}_j^{(i)}}$, for a token v_j at position i is calculated as follows:

$$\widetilde{\mathbf{l}_j^{(i)}} = M_w(\mathbf{x}, \mathbf{t}^{0:(i-1)}) = \begin{cases} M(\mathbf{x}, \mathbf{t}^{0:(i-1)})[j] + \delta, & v_j \in G \\ M(\mathbf{x}, \mathbf{t}^{0:(i-1)})[j], & v_j \in R \end{cases} \tag{7}$$

This algorithm biases towards green tokens, leading to a higher proportion in watermarked texts. The detector categorizes each token as red or green using the hash function and calculates the green token ratio with the z-metric, defined as:

$$z = (|s|_G - \gamma T)/\sqrt{T\gamma(1-\gamma)} \tag{8}$$

where T is the length of the text, γ is the ratio of the green list. A text exceeding a certain green token threshold is deemed watermarked.

KGW's detection method showed low false positive ($< 3 \times 10^{-3}\%$) and false negative ($< 1\%$) rates in tests. Yet, real-world application challenges necessitate further optimization and design. The following outlines five optimization objectives, along with the improvements and explorations in watermark algorithms under each objective. The taxonomy of this section is depicted in Figure 6.

4.1.1 Enhancing Watermark Detectability. Although KGW reported high detection performance, it showed weaknesses under more rigorous detection conditions, necessitating targeted optimization of z-score computation.

One critical issue is the discrepancy between theoretical and actual false positive rates (FPRs) when the theoretical FPR is extremely low (e.g., below 10^{-6}). This occurs because KGW assumes the z-score follows a Gaussian distribution, but this is only valid when the token length approaches infinity, and is therefore often inaccurate in practice. To resolve this, Fernandez et al. [22] developed a non-asymptotic statistical test that adopts

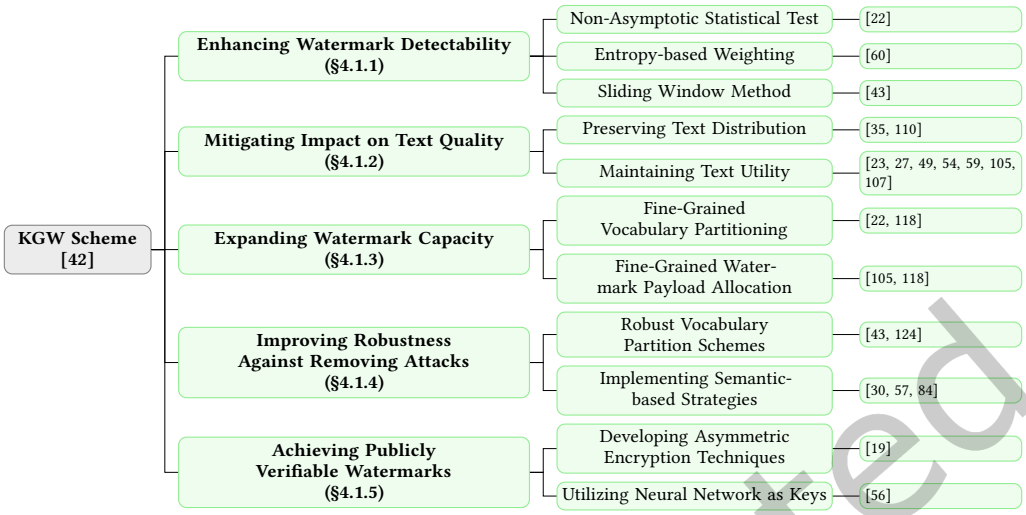


Fig. 6. Taxonomy of watermarking during logits generation. The root node represents the fundamental watermarking scheme KGW [42], followed by branches illustrating five optimizing objectives.

the accurate binomial distribution and corrects z-score calculations accordingly, aligning theoretical and actual FPRs in low-FPR scenarios.

Another challenge with KGW is its performance in low-entropy scenarios, such as code generation and machine translation. In these cases, logits vectors often display uneven distributions, reducing the impact of bias on green tokens and lowering watermark detection sensitivity. The EWD [60] method addresses this by assigning weights to tokens based on their entropy during detection, enhancing sensitivity by emphasizing high-entropy tokens in z-score calculations.

Additionally, KGW struggles when watermarked text is mixed with extensive non-watermarked text, diluting watermark strength and affecting z-score calculations. To mitigate this, the WinMax [43] sliding window-based method calculates z-scores across different window sizes within the text and selects the maximum z-score as the final value, improving detection accuracy.

4.1.2 Mitigating Impact on Text Quality. Watermarking can impact text quality by introducing unnatural word choices or patterns that degrade readability, coherence and text utility. An optimization perspective to mitigate this impact is to ensure that the text distribution of watermarked content remains consistent with the original output distribution of the LLM. Specifically, this means that the expected watermarked logits equal the original LLM's logits:

$$\mathop{\mathbb{E}}_{k\sim K}[M_w(\mathbf{x}, \mathbf{t}^{0:(i-1)})] = M(\mathbf{x}, \mathbf{t}^{0:(i-1)}), \tag{9}$$

where each key k represents a unique red-green list split. This ensures that, in expectation, the watermark does not negatively affect text quality.

Hu et al. [35] noted that the KGW algorithm [42] introduces bias in its logits modification, altering the text distribution. The bias in KGW arises from applying a uniform δ to green list tokens, which disproportionately affects low-probability tokens, leading to overall bias. To counter this, Hu et al. [35] introduced two unbiased reweighting methods: δ-reweight, which samples a one-hot distribution from the original logits, and γ-reweight, which halves the probability distribution range, thereby doubling the probabilities of the remaining tokens. Similarly, Wu et al. [110] proposed the α-reweight method, which discards tokens with probabilities below α

and adjusts the rest. These methods are theoretically unbiased, preserving the original text distribution and, consequently, maintaining text quality.

The aforementioned optimization approach is relatively theoretical. A more practical perspective focuses on mitigating the impact on text utility. One class of methods selectively applies watermarks to tokens, avoiding positions where suitable tokens are sparse. Entropy has been proposed as a criterion for watermark application, bypassing low-entropy positions [49, 59, 105]. Wouters [107] goes further by considering not only entropy but also the probability distribution of the red and green lists, refraining from watermarking when the probability of red tokens is too high. Another class of method is to maintain text utility by reducing the likelihood of suitable words being banned. For example, Liang et al. [54] proposes using mutual exclusion rules for red-green partitioning, aiming to evenly distribute semantically similar words between the red and green lists. This approach aims to prevent situations where all appropriate words are allocated to the red list. Moreover, Fu et al. [23] suggests employing semantic-aware watermarking, which involves adding words with higher relevance to the context into the green list, thus increasing the likelihood of appropriate tokens being selected. Furthermore, Guan et al. [27] suggests adding an extra bias to suitable tokens, reducing the probability of their exclusion.

4.1.3 Expanding Watermark Capacity. The KGW watermark algorithm [42] can only verify watermark presence, classifying it as a zero-bit watermark. Yet, many applications require watermarks to convey additional information like copyright details, timestamps, or identifiers, leading to the need for multi-bit watermarks capable of extracting meaningful data.

One possible solution is to employ fine-grained vocabulary partitioning, expanding from a binary red-green partition to a multi-color partition [22]. To encode b bits of information, the vocabulary needs to be divided into 2^b groups, with different watermark information reflecting preferences for tokens from different groups. Another solution is to use fine-grained watermark payload allocation [105], dividing the text into multiple chunks, with each chunk encoding a portion of the bit information. However, if either of these approaches is too fine-grained in its partitioning, the encoding strength may be insufficient, leading to reduced watermark extraction accuracy. To mitigate this issue, Yoo et al. [118] attempts to combine these two methods, ensuring an adequate watermark capacity while avoiding overly fine-grained partitioning of any kind, and aims to find an optimal balance of granularity through experimentation.

4.1.4 Improving Robustness Against Removing Attacks. As discussed in Section 2, an effective watermarking algorithm must be robust against removal attacks, ensuring the watermark remains detectable. These attacks usually modify the text without altering its semantic content. While the KGW algorithm [42] exhibited some robustness in their experiments, there is still room for improvement.

One optimization approach is to adopt more robust vocabulary partition schemes. The original KGW [42] algorithm utilizes all token information within the preceding context window to map to a hash value for red-green partitioning. Kirchenbauer et al. [43] further elaborated on more robust partition strategies, such as using only the samllest token id in the preceding context window for hashing, which is more resilient to text editing. Additionally, Zhao et al. [124] proved that a fixed global split of red and green lists offers greater resistance to removal attacks.

As watermark removal attacks usually preserve the semantic content of the text, several studies have developed methods to integrate semantic information into the design of watermarking algorithms. For example, Liu et al. [57] trained a watermarked LLM that directly converts text semantics into red-green partitions, ensuring that similar text semantics result in similar partition outcomes, thereby achieving robustness. He et al. [30] improves the watermark LLM by adding constraints that ensure semantically similar tokens fall into the same color list, thereby further enhancing robustness. Ren et al. [84] converted semantic embeddings into semantic values through weighted embedding pooling followed by discretizing using NE-Ring, and then divided the vocabulary into red-list and green-list based on these semantic values.

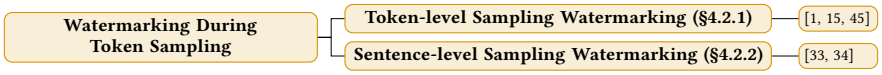


Fig. 7. Taxonomy of watermarking during token sampling.

4.1.5 Achieving Publicly Verifiable Watermarks. Achieving publicly verifiable watermarks is significant as it allows anyone to authenticate the origin and integrity of the content without requiring access to a secret key. This reduces service overhead in private detection scenarios (where the detector is placed behind an API) and its transparency also enhances trust and accountability. Most previous watermarking algorithms cannot achieve public verifiability because their watermark generation details are involved in detection (e.g., the hash key in KGW). As a result, exposing the detector also means exposing the generator, making the watermark vulnerable to targeted removal or forgery.

To achieve publicly verifiable watermarks, Fairoze et al. [19] have utilized digital signature technology from the field of cryptography. This approach involves generating watermarks using a private key and verifying them with a public key. However, verification via a public key relies on features extracted from the text, which can still be exploited to some extent to forge watermarks. Further advancing this field, Liu et al. [56] proposed the use of different neural networks for watermark generation and detection. Due to the black-box nature of neural networks, the details of watermark generation are not exposed, which could defend against watermark forgeries in public detection scenarios.

4.2 Watermarking during Token Sampling

The previous section primarily focused on incorporating watermarks during the logits generation phase for LLMs. In this section, we will introduce a technique of watermarking during token sampling, which does not alter the logits but utilize watermark message to guide the sampling process. Based on the granularity of guiding, this technique can be divided into two main approaches (as depicted in Figure 7): token-level sampling watermarking (§4.2.1), which embeds watermarks during each token's sampling, and sentence-level sampling watermarking (§4.2.2), which uses watermark message to guide the sampling of entire sentences.

4.2.1 Token-level Sampling Watermarking. The principle of incorporating watermarks during the token sampling phase is derived from the randomness inherent in token sampling. In this scenario, watermarks can be introduced using a fixed random seed, where a pseudo-random number generator produces a sequence of pseudo-random numbers to guide the sampling of each token. For watermark detection, it is only necessary to assess the alignment between the text tokens and the pseudo-random numbers, specifically evaluating whether the choice of each token in the text matches with the corresponding value in the random number sequence.

For instance, Christ et al. [15] proposed a watermarking algorithm designed for a toy LLM with a vocabulary consisting of only the digits 0 and 1, with the pseudo-random numbers represented as a series of values $u \in [0, 1]$. If the predicted probability for a certain position exceeds the corresponding pseudo-random number, then 1 is sampled at that position, otherwise 0. In the detection of watermarks, it can be determined whether the values of the pseudo-random numbers corresponding to the positions with 1 in the binary tokens are significantly higher than those with 0. Around the same time, Aaronson and Kirchner [1] proposed a watermarking algorithm based on a similar idea, but suitable for real LLMs. In this case, the LLM (with a vocabulary size of $|V|$) outputs a probability vector $p_i = (p_{i1}, ..., p_{i|V|})$ at position i, and the pseudo-random sequence at position i is also transformed from a single number (0 or 1) into a pseudo-random vector $r_i = (r_{i1}, ..., r_{i|V|})$. During sampling, exp-minimum sampling is applied to choose the token j that maximizes $r_{ij}^{1/p_{ij}}$. To check a watermark, the alignment between the text and the pseudo-random vector sequence is assessed to see if it exceeds a threshold.

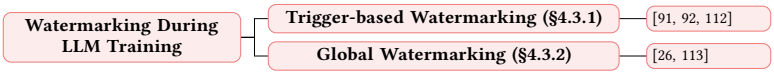


Fig. 8. Taxonomy of watermarking during LLM training.

However, these methods still faces two challenges: 1) the detection algorithm is not robust enough against watermark removal attacks, which involves certain text modifications, and 2) due to the fixed nature of pseudo-random numbers, the LLM with watermark will generate the same text for the same prompt each time, thereby losing the inherent randomness in text generation by LLM. To address these issues, Kuditipudi et al. [45] proposed the use of a pseudo-random number sequence significantly longer than the text, randomly selecting a starting position from the sequence for each watermark insertion to introduce randomness. Additionally, during watermark detection, they incorporate a soft notion of edit distance (i.e., Levenshtein distance) into the computation of the alignment between text and the pseudo-random number sequence. This approach significantly enhances the robustness of the watermarking algorithm against watermark removal attacks.

4.2.2 Sentence-level Sampling Watermarking. Token-level sampling watermarking algorithms may not be robust to token-level text edits. However, since such edits often don't significantly change sentence semantics, some methods leverage sentence-level sampling watermarking to achieve better robustness.

Building on this idea, SemStamp [33] partitions the semantic embedding space into a watermarked region and a non-watermarked region. The algorithm performs sentence-level rejection sampling until the sampled sentence falls within the watermarked region. SemStamp employs Locality-Sensitive Hashing to randomly partition these regions, which can result in semantically similar sentences being placed into different regions, thereby diminishing the robustness of the watermarking algorithm. To address this issue, k-SemStamp [34] utilizes k-means clustering to divide the semantic space into k regions, ensuring that semantically similar sentences are grouped into the same region. Each region is then designated as either a watermarked region or a non-watermarked region. This approach ensures that a sentence remains within the same region even after semantic-preserving modifications, thereby enhancing robustness against semantic-invariant text editing attacks.

Current research in sampling-based watermarking is limited, indicating room for advancement. The effectiveness and robustness of these methods warrant further exploration through experiments and real-world applications.

4.3 Watermarking during LLM Training

Although adding watermarks during the logits generation (§4.1) and token sampling (§4.2) stages can be effective during inference, they are not suitable for open-source LLMs. This is because watermarking code added after the logits output can be easily removed. Therefore, for open-source LLMs, watermarks must be embedded into the LLM's parameters during training. As shown in Figure 8, training-time watermarking can be categorized into trigger-based watermarks, which are effective only for specific inputs, and global watermarks, which are intended to work for all inputs.

4.3.1 Trigger-based Watermarking. Trigger-based Watermarking is a type of backdoor watermarking that introduces specific triggers into an LLM. When these triggers appear in the input, the LLM exhibits specific behaviors (specific formats or outputs). Such watermarks can be added by dataset providers to protect dataset copyright, or by LLM providers to protect LLM copyright.

Sun et al. [92] proposed CoProtector for the code generation task, using word-level or sentence-level modifications in the code as triggers to generate corrupting code, which typically has incorrect functionality. Further, Sun et al. [91] proposed CodeMark, which uses semantically invariant code transformations as triggers, ensuring

the correct functionality of the code while embedding a trigger-based watermark with minimal impact on LLM performance.

In the context of protecting LLM copyright, Xu et al. [112] proposed the Hufu watermark. This watermark does not rely on specific input triggers but uses a particular input format as a trigger. It leverages the permutation equivariance property of transformers, training the LLM to recognize a specific permutation as a watermark.

4.3.2 Global Watermarking. Although trigger-based text watermarking is effective in many cases, it only works when specific triggers are present and cannot work for all inputs.

Global watermarks can add detectable markers to all content generated by LLMs, enabling content tracking. Gu et al. [26] explored the learnability of watermarks, investigating whether LLMs can directly learn to generate watermarked text. They proposed two learning methods: sampling-based watermark distillation and logit-based watermark distillation. These methods offer the possibility of transforming inference-time watermarking into inherent LLM parameters. Xu et al. [113] further proposed using reinforcement learning to optimize LLM watermarks. They used reinforcement learning techniques to optimize LLMs based on feedback from watermark detectors, embedding watermarks into the LLM. Experimental results show that this method achieves near-perfect watermark detection and strong resistance to interference, significantly improving watermark effectiveness and robustness. However, due to the black-box nature of LLMs, this watermark training method may be less stable on out-of-distribution data compared to inference-time watermarks.

5 Evaluation Metrics for Text Watermarking

In Sections 3 and 4, we provided a comprehensive overview of existing text watermarking techniques. A thorough evaluation of text watermarking algorithms is crucial. As illustrated in Figure 9, this section details the evaluation metrics from multiple perspectives: (1) the detectability of watermarking algorithms (§5.1), (2) the impact of watermarking on the quality of targeted texts (§5.2) and LLMs (§5.3 and §5.4), and (3) the robustness of watermarking algorithms against untargeted (§5.5) and targeted watermark attacks (§5.6). In addition, we enumerate representative evaluation benchmarks and tools (§5.7).

5.1 Detectability

For text watermarking algorithms, the basic requirement is that the watermarked text can be detected. In this section, we will summarize how watermarking algorithms measure their detectability. We will introduce the detection metrics for zero-shot and multi-bit watermarking algorithms, as well as the watermark size, which indicates how long the text needs to be for detection.

5.1.1 Zero-bit Watermark. In zero-bit watermarking, the goal is to detect the presence of a watermark. Current watermark algorithms typically provide a detector that uses hypothesis testing to generate a z-score or p-value [42, 45, 56], along with a threshold to distinguish whether a text contains a watermark. For testing, a dataset with an equal number of watermarked and human texts is usually constructed. The detector is then used to evaluate this dataset, calculating the F1 score and corresponding false positive and false negative rates. The false positive rate, which indicates the probability of misclassifying human text as watermarked, is particularly important as it can have more severe consequences than false negatives.

The challenge with this detection method lies in selecting an appropriate threshold, as different methods may have different threshold selection approaches. For example, some studies [42, 56, 57, 124] report F1 scores at fixed false positive rates of 1% and 10%, while others [57] show the best F1 scores across all thresholds to facilitate a fairer comparison of algorithm performance.

5.1.2 Multi-bit Watermark. In multi-bit watermarking methods [2, 85, 105, 115, 117, 118], the watermark detection algorithm must not only detect the presence of a watermark but also extract specific information. For example, a

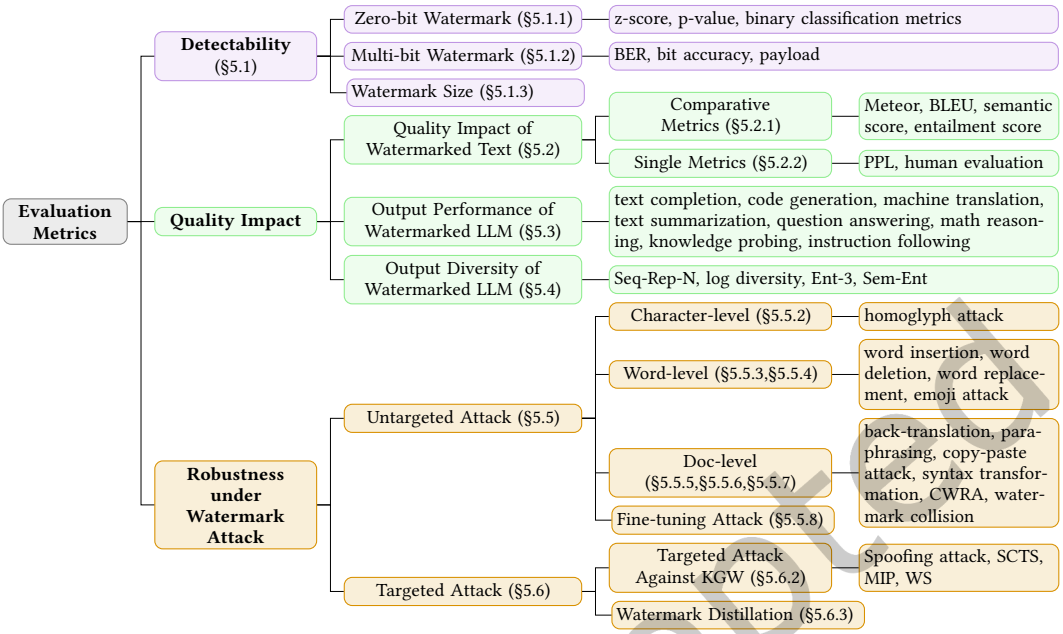


Fig. 9. Taxonomy of Evaluation Metrics for Text Watermarking.

watermarked text might encode specific data like *"This text is generated by GPT-4 on June 6 by the Administrator"* [105]. Common detection metrics include bit error rate (BER) [117] and bit accuracy [2, 118]. For a watermark message w encoded as n bits, represented as $w = b_1 b_2 \dots b_n$, where each b_i is binary, BER refers to the probability of incorrectly predicted bits, while bit accuracy refers to the proportion of correctly predicted bits.

Additionally, the bit capacity or payload of a watermark algorithm is a key evaluation metric, typically referred to as Bits Per Watermark [115, 117] or code rate [85, 105]. Payload is calculated by dividing the total number of bits of the watermark information by the number of tokens.

5.1.3 Watermark Size. Generally, for a text watermarking algorithm, the longer the text, the easier it is for the watermark to be detected because longer texts provide more modification space. Thus, determining how long a text needs to be for the watermark to be reliably detected becomes a crucial metric, known as the watermark size. Piet et al. [78] studied the watermark size of current mainstream watermarking algorithms, exploring the minimum length required to achieve a 2% false positive rate. The study found that among the KGW [42], Aar [1], and KTH [45] algorithms, KGW requires the shortest detection length, indicating that KGW has the best watermark size. Currently, there is still limited research on watermark size, and future work is recommended to include watermark size as an evaluation metric.

Typically, for a text watermarking algorithm, high detectability is a relatively low requirement. More importantly, these algorithms should have minimal impact on text quality and high robustness against various attacks. In the following two sections, we will separately introduce how to evaluate the quality of watermarked texts (§5.2) and the quality assessment of watermarked LLMs (§5.3).

5.2 Quality Impact of Watermarked Text

The quality evaluation of watermarked text primarily targets the series of algorithms for watermarking existing text (§3). In these algorithms, the input is a non-watermarked text, and the output is the modified watermarked

Table 1. The quality evaluation metrics for different text watermarking algorithms regarding their impact on text or LLM quality. *w. Ext. LLM* indicates whether external LLMs are used, *Diverse Eval* indicates whether text diversity is evaluated, *Time Cmplx.* indicates the evaluation time complexity: human-based metrics are high, LLM-based are medium, and just gold standard comparisons are low, while pass@k varies based on the specific generated program, *Eval Type* indicates whether the evaluation scores single texts (*Single*) or compares pairs of texts (*Comp.*), *Tested Algorithms* lists the algorithms tested on each metric, and *Quality Preserve* refers to algorithms that demonstrate the preservation of text quality under the corresponding metric, based on the analysis provided in the respective studies conducting the tests.

Evaluation Task	Metric	w. Ext. LLM?	Diversity Eval?	Time Cmplx.	Eval Type	Tested Algorithms	Quality Preserve
Quality Evaluation for Watermarked Text							
	Semantic Score	✓	✗	M.	Comp.	[2, 68, 114, 115, 117, 119]	[2, 68, 114, 115, 117, 119]
	Meteor Score	✗	✗	M.	Comp.	[2, 114]	[2]
	Entailment Score	✓	✗	M.	Comp.	[115, 117]	[115, 117]
	BLEU Score	✗	✗	L.	Comp.	[89, 98, 119]	[89, 98, 119]
	Perplexity	✓	✗	M.	Single	[89]	[89]
	Human Evaluation	✗	✗	H.	Single	[2, 117]	[2, 117]
Quality Evaluation for Watermarked LLM							
Text Completion	PPL	✓	✗	M.	Single	[26, 33, 34, 42, 54, 56, 57, 59, 60, 84, 105, 107, 118, 124]	[33, 34, 42, 47, 54, 56, 57, 60, 84, 105, 107, 118, 124]
	P-SP	✓	✗	M.	Comp.	[118]	[118]
	GPT4-Score	✓	✗	M.	Single	[19]	[19]
	Seq-Rep-N	✗	✓	L.	Single	[26]	[26]
	Log Div.	✗	✓	L.	Single	[43]	[43]
	Ent-3	✗	✓	L.	Single	[33, 34]	[33, 34]
	Sem-Ent	✓	✓	M.	Single	[33, 34]	[33, 34]
Code Generation	Pass@k	✗	✗	L.	Single	[22, 49, 60]	[22, 49, 60]
	CodeBlue	✗	✗	L.	Comp.	[27]	[27]
	Edit Sim	✗	✗	L.	Comp.	[42, 43, 124]	-
Machine Translation	BLEU	✗	✗	L.	Comp.	[35, 56, 110]	[35, 56, 110]
	BERTScore	✓	✗	M.	Comp.	[35, 110]	[35, 110]
	PPL	✓	✗	M.	Single	[35, 110]	[35, 110]
Text Summarization	BLEU	✗	✗	L.	Comp.	[30, 35, 42, 43, 110, 124]	[35, 110]
	BERTScore	✓	✗	M.	Comp.	[35, 110]	[35, 110]
	PPL	✓	✗	M.	Single	[35, 110]	[35, 110]
Question Answering	ROUGE	✗	✗	L.	Comp.	[42, 43, 124]	-
	Exact Match	✗	✗	L.	Comp.	[22]	[22]
	GPT-Truth	✓	✗	M.	Single	[54]	[54]
	GPT-Info	✓	✗	M.	Single	[54]	[54]
Math Reasoning	Accuracy	✗	✗	L.	Comp.	[22, 54]	[22, 54]
Knowledge Probing	F1 Score	✗	✗	L.	Comp.	[42, 43, 124]	-
Instruction Following	GPT4-Judge	✓	✗	M.	Comp.	[42, 43, 124]	-

text. Therefore, the key to evaluating the quality of watermarked text is comparing the quality differences between the watermarked text and the original text. There are two evaluation methods: one uses comparative metrics such as semantic score and BLEU score [2, 68, 114, 115, 117, 119]; the other involves scoring the original text and the watermarked text separately and then comparing the scores [89].

5.2.1 Comparative evaluation metrics. For comparative evaluation metrics, the main purpose is to assess the similarity between watermarked text and the original text. Based on the method of evaluating similarity, these metrics can be divided into two categories: surface feature-based metrics, such as Meteor Score [5] and BLEU Score[73], and semantic feature-based metrics, such as Semantic Score and Entailment Score.

Meteor score and BLEU score are important evaluation metrics in the field of machine translation, and when applied to text watermarking, the original text can be used as the reference text. BLEU focuses on the n-gram overlap between the target (watermarked) text and the reference text, providing a composite score by calculating precision and length penalty. However, BLEU's limitation is its overemphasis on exact matches and sensitivity to word order and slight morphological changes, which may not fully capture semantic equivalence. In contrast, the Meteor Score offers further improvements. Besides exact matches, Meteor Score also considers morphological changes (e.g., verb tenses, noun plurals) and synonym matches. It evaluates the similarity between the target text and the reference text more flexibly through word alignment. Nonetheless, both Meteor Score and BLEU Score primarily assess the similarity of texts at the surface level.

Although Meteor Score and BLEU Score effectively evaluate surface-level similarity, in some cases, evaluating surface similarity alone is insufficient; semantic impact must also be considered. Therefore, some works introduce the semantic score [2, 68, 114, 115, 117, 119]. A common method for evaluating semantic scores is to calculate semantic embeddings using LLMs and then compare these embeddings using cosine similarity. This process can be represented by the following formula:

$$\mathcal{R}_{\text{se}}(W_u, W_w) = \frac{\mathcal{M}(W_u) \cdot \mathcal{M}(W_w)}{\|\mathcal{M}(W_u)\| \times \|\mathcal{M}(W_w)\|} \tag{10}$$

where W_u and W_w represent the non-watermarked text and the watermarked text, respectively. The model $\mathcal{M}$ is typically a large language model optimized for text similarity. For example, Munyer and Zhong [68] used the Universal Sentence Encoder [11], while Abdelnabi and Fritz [2], Yang et al. [115], Yoo et al. [117] used Sentence-BERT [83], and Yang et al. [114] used all-MiniLM-L6-v2.

Additionally, to determine more fine-grained relationships between the original and modified watermarked texts, some works utilize LLMs pre-trained on Natural Language Inference (NLI) [115, 117] tasks to judge the relationship between two sentences. For example, Yoo et al. [117] used RoBERTa-Large-NLI [83] to more accurately understand and infer complex semantic relationships between texts (Entailment Score, ES). This Entailment score not only focuses on the overall similarity between two texts but also identifies subtle semantic differences.

5.2.2 Single Text evaluation metrics. Unlike comparative evaluation metrics such as BLEU Score, single text evaluation metrics focus on separately scoring the quality of the original text and the watermarked text, and then comparing these scores. Currently, the quality evaluation of watermarked text mainly uses Perplexity (PPL) or direct human evaluation.

PPL is defined as the exponentiated average negative log-likelihood of a sequence. Specifically, given a text $W = \{w_1, ..., w_N\}$, PPL can be computed using a LLM$\mathcal{M}$:

$$\mathcal{R}_{\text{PPL}}(W) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log \mathcal{M}(w_i|w_1, \ldots, w_{i-1})\right). \tag{11}$$

PPL is an important metric for evaluating text coherence and fluency. Generally, lower PPL indicates higher text quality. For more accurate evaluation, larger LLMs are typically used to calculate PPL, such as GPT-2 [114], GPT-3 [124], OPT-2.7B [42, 105], and LLaMA-13B [56, 57].

Although PPL can conveniently evaluate text quality using LLMs, due to the inherent flaws of PPL (such as misrating some repetitive texts) and the accuracy of LLM outputs, many works further employ human scoring for text evaluation [2, 117]. It should be noted that although human scoring is generally considered more accurate,

human annotators are also prone to annotation errors. Typically, multiple human annotators are required to score the same data. However, due to high annotation costs, human scoring is challenging for large-scale evaluation.

5.3 Output Performance Evaluation for Watermarked LLM

In the previous section (§5.2), we discussed how to evaluate the quality of watermarked text, which mainly pertains to watermarking for existing text (§3). For the more prevalent watermarking for existing LLM (§4) in the era of LLMs, it is usually necessary to evaluate the capabilities of the watermarked LLM. This typically involves evaluation on a series of downstream tasks, such as Text Completion [26, 33, 34, 42, 54, 56, 57, 59, 60, 84, 105, 107, 118, 124], Code Generation [22, 49, 60], and Machine Translation [35, 56, 57, 110]. In this section, we will detail the various specific downstream tasks and the evaluation metrics used for these tasks.

5.3.1 Text Completion. Since most LLMs are trained using the next word prediction paradigm, text completion is a capability that all LLMs possess. Therefore, the most common task for testing LLM capabilities is text completion. The specific approach is to provide the LLM with a text prefix as a prompt, have the LLM generate the subsequent text, and then evaluate the quality of the generated text. Currently, the evaluation of the quality of generated text typically leverages other LLMs, including PPL [26, 33, 34, 42, 54, 56, 57, 59, 60, 84, 105, 107, 118, 124]based on the likelihood generated by LLMs, the P-SP [43, 118] based on text similarity, and GPT-4-score [19], which uses the more powerful GPT-4 to directly score the text.

The calculation method for the PPL metric here is the same as the eq 11 mentioned in §5.2.2. Generally, when testing PPL, larger LLMs than the current LLM are used for evaluation, with common LLMs including LLaMA-13B [57], LLaMA-70B [56], and GPT-3 [124]. Due to the general applicability and simplicity of PPL, calculating the PPL metric after the LLM performs a text completion task is currently the most widely adopted evaluation method.

P-SP is more similar to a semantic score, used to evaluate the semantic similarity between two texts. Yoo et al. [118] used P-SP [106] to evaluate the semantic similarity between original human-written texts and watermarked texts generated by LLMs through the text completion task using these texts' prefixes. However, since the same text prefix can generate texts with different semantics, this evaluation has not been widely adopted.

Since the PPL and P-SP can only evaluate text quality from certain perspectives, where PPL focuses on text coherence and P-SP on semantic similarity with the original text. Fairoze et al. [19] adopted GPT-4 [71] to assess the quality of text generated in the text completion task. Specifically, they designed a scoring prompt for GPT-4, enabling it to output an evaluation of text quality.

5.3.2 Code Generation. Text completion is typically used in high-entropy scenarios and is less sensitive to changes in LLM capabilities. In contrast, low-entropy tasks like code generation are more sensitive to changes in LLM capabilities when adding watermarking, as even a small error in code can lead to failure or incorrect execution. For code generation, there are usually two evaluation methods: one is based on surface form matching, such as CodeBlue [27] and Edit Sim [101], and the other is based on actual execution accuracy, such as Pass@k [22, 49, 60].

CodeBLEU [27] is a metric specifically designed for evaluating the performance of code generation tasks. It extends the classic BLEU metric by incorporating the unique characteristics of code to better reflect the quality of code generation. CodeBLEU considers not only the lexical match between the generated code and the reference code but also the syntactic and semantic match. Overall, CodeBLEU is a metric based on the n-gram matching degree between the reference code and the generated watermarked code. Edit Sim [101], calculates the edit distance between two pieces of code. Both CodeBLEU and Edit Sim focus only on the surface matching degree. However, code with the same execution result can be written in different ways. Therefore, a more commonly used metric is Pass@k [22, 49, 60], which selects the top k most probable code generated by the watermarked LLM and determines the probability that one of them executes correctly.

5.3.3 Other Tasks. In addition to testing in high-entropy and low-entropy environments (text completion and code generation), many works have evaluated the capabilities of watermarked LLMs on other typical tasks. These tasks include Question Answering [22, 30, 54, 101], Machine Translation [35, 56, 57, 110], Text Summarization [30, 35, 101, 110], Math Reasoning [22, 54], Knowledge Probing [101], and Instruction Following [101].

The Question Answering task covers a wide range. For settings like TriviaQA [39], which normally requires short answers, the exact match is used for evaluation. For settings like ELI5 [20], which require longer outputs, the Rouge is generally used. The ROUGE is similar to the BLEU, as it evaluates the similarity between the reference text and the target text (watermarked text) through n-gram similarity. For some special scenarios, such as TruthfulQA [55], which is used to evaluate the truthfulness and accuracy of LLMs, the GPT-Truth [54] and GPT-Info [54] metrics generated by GPT-4 can be used for evaluation.

Similarly, Machine Translation and Text Summarization tasks are often used to evaluate the capabilities of watermarked LLMs. This is typically done by adding watermarks to LLMs fine-tuned specifically for these tasks (e.g., NLLB-200 [16]). Both tasks compare the watermarked text and reference text in terms of n-gram similarity (BLEU for Machine Translation, ROUGE for Text Summarization), semantic similarity (BERT-Score), and the PPL value of the watermarked text.

Additionally, some works evaluate the capabilities of watermarked LLMs by testing their performance on other tasks. For example, the Math Reasoning [22, 54] is used to evaluate the reasoning ability of LLMs, typically using the accuracy metric to determine if the LLM answered correctly. The Knowledge Probing [101], evaluates the knowledge capability of LLMs, usually through the F1 score as the metric. Moreover, the instruction following capability [101] of LLMs is often evaluated using GPT-Judge to compare the original LLM and the watermarked LLM in following instructions, typically assessed by win-rate metric.

5.4 Output Diversity Evaluation for Watermarked LLM

In the previous section, we primarily discussed how to evaluate the output performance of watermarked LLMs. However, another important aspect of evaluating LLM quality is evaluating the diversity of LLM outputs. Since watermarking algorithms typically prefer certain output content, watermarked LLMs often suffer from reduced output diversity, making its evaluation a significant topic. Current work mainly focuses on evaluating the output of the Text Completion task, including metrics such as Seq-Rep-N [26], Log Diversity [43], Ent-3 [33, 34], and Sem-Ent [33, 34].

5.4.1 Seq-Rep-N. Seq-Rep-N is typically used to calculate the repetition of n-grams in a sentence. It can be obtained by calculating the ratio of the number of unique n-grams to the total number of n-grams in a sentence. Specifically, the following formula can be used: $1 - \frac{\text{Number of unique 3-grams}}{\text{Total number of 3-grams}}$ [26] analyzed using the Seq-Rep-3 metric and found that the Aar [1] has significantly higher repetition compared to KGW [42] and KTH [45]. Additionally, as the required window size increases, the repetition decreases, but this sacrifices some robustness to modifications.

5.4.2 Log Diversity. Furthermore, Kirchenbauer et al. [43] proposed the log diversity metric. This metric calculates the negative logarithm multiplied by the proportion of unique n-grams from 1 to N, rather than just the repetition ratio of a specific n-gram. The following formula can be used for calculation: $\mathcal{R}_d = -\log\left(1 - \prod_{n=1}^{N}(1-u_n)\right)$, where u_n represents the ratio of the number of unique n-grams to the total number of n-grams in a given text sequence. Similarly, Kirchenbauer et al. [43] also found that log diversity increases with the window size.

5.4.3 Ent-3. Ent-3 measures the entropy of the frequency distribution of 3-grams in the text. A higher entropy value indicates greater lexical diversity. The following formula can be used for calculation: $H = -\sum_i p_i \log p_i$, where p_i is the probability of the ith 3-gram occurring. A higher Ent-3 value indicates more diverse word choices

in the generated text. Hou et al. [33] shows that their algorithm can enhance robustness without reducing diversity.

5.4.4 Sem-Ent. Semantic Entropy (Sem-Ent) measures diversity by performing clustering analysis on the semantic representations of generated text and then calculating the entropy of the cluster distribution. It uses the LLM to generate semantic representations of the text, followed by k-means clustering analysis on all semantic representations, with a specified number of cluster centers k. The entropy of the cluster distribution is then calculated based on the proportion of sentences in each cluster: $H = -\sum_i p_i \log p_i$ where p_i is the proportion of sentences in the i-th cluster. A higher Sem-Ent value indicates greater semantic diversity. Sem-Ent focuses more on semantic diversity rather than just surface-form diversity.

Notably, for clearer understanding, in Table 1 we outline evaluation metrics for quality impact mentioned in §5.2, §5.3 and §5.4, along with their various characteristics and the algorithms evaluated on each metric.

5.5 Untargeted Watermark Attacks

For a text watermarking algorithm, another important evaluation aspect is its robustness against watermark attacks. Table 2 lists different types of watermark attacks. These attacks may either unintentionally modify the watermarked text (untargeted) or attempt to crack the watermark generation method to remove or forge the watermark (targeted). In this seciton, we focus on untargeted watermark attacks, leaving targeted watermark attacks for the next section.

5.5.1 Threat Model. We first introduce the threat model for untargeted watermark attacks. We assume that the user has obtained a watermarked text. The user may or may not know that the text contains a watermark, but they do not know how the watermark was embedded. The user might modify the watermarked text, possibly at the character, word, or sentence level, or they might insert the watermarked text into a longer, non-watermarked human text. A robust text watermarking algorithm should maintain detectability as much as possible after such modifications.

5.5.2 Character-level Attack. Merely modifying characters in the text without replacing words is a basic strategy. Random character replacement is relatively easy to detect. An alternative strategy is using visually similar Unicode characters (homoglyph attacks [24]), which are harder for humans to notice but can be mitigated through normalization techniques [42]. Therefore, performing normalization preprocessing before passing the text to the watermark detector is crucial.

Character-level attacks affect different types of watermark algorithms differently. For format-based watermark algorithms, such as those using Unicode ID replacement to embed watermarks (e.g., EasyMark [89] using Unicode 0x0020 to replace 0x2004), homoglyph attacks can be a straightforward and effective method to remove watermarks. For other watermark algorithms, the impact mainly lies in the tokenizer; after character modification, the tokenizer may break down words into different token lists. This change in tokenization results poses a challenge to the detection effectiveness of many watermark algorithms. In summary, character-level attacks have minimal impact on text quality but are relatively easy to detect and can be removed by other methods. Therefore, this is not reliable for all scenarios.

5.5.3 Word-level Attack to Existing Text. Word-level attacks to existing text involve inserting, deleting, or replacing words in pre-generated watermarked text [2, 42, 45, 114, 117, 124]. These modifications are usually done at a fixed attack rate. Among all modification methods, synonym replacement is commonly used to minimize semantic impact, preferring replacement words that cause the least difference in sentence scoring (e.g., using BERT score [17]).

Table 2. The attack methods for different text watermarking algorithms. *Perform Gran.* indicates the granularity of algorithm execution, *Target Attack?* indicates whether the attack involves breaking the watermark algorithm details, *Perform Stage* indicates whether the attack is performed before or after watermark text generation, *Time Cmplx.* indicates time complexity: methods requiring LLM fine-tuning or extensive watermark rule mining are high, using LLMs to modify individual texts is medium, and rule-based text modifications are low, *Quality Impact* indicates the effect on text quality: low impact for unchanged text, character encoding changes, or LLM-based rewrites; medium impact for rule-based semantic invariant replacements or rewrites; and high impact for random word addition or deletion, *Tested Methods* lists the algorithms tested on each attack, and *Robust Methods* refers to algorithms that show robustness under the corresponding attack, based on the analysis provided in the respective studies conducting the tests.

Attack Method	Perform Gran.	Target Attack?	Perform Stage	Time Cmplx.	Quality Impact	Tested Methods	Robust Methods
Homo. Attack [24]	Char	✗	Post-Gen	L.	L.	-	-
Word Insertion	Word	✗	Post-Gen	L.	H.	[1, 2, 15, 42, 45, 48, 68, 110, 117, 119]	[2, 42, 45, 48, 68, 110, 117, 119]
Word Deletion	Word	✗	Post-Gen	L.	H.	[1, 2, 15, 42, 45, 48, 68, 110, 114, 117, 119, 124]	[2, 42, 45, 48, 68, 110, 114, 117, 119, 124]
Word Replacement	Word	✗	Post-Gen	L.	M.	[1, 2, 15, 26, 42, 43, 45, 48, 49, 54, 57, 68, 105, 110, 113, 114, 117, 119, 124]	[1, 2, 26, 42, 43, 45, 48, 49, 54, 57, 68, 110, 113, 117, 119, 124]
Emoji Attack [42]	Word	✗	Pre-Gen	M.	M.	-	-
Back-translation [18]	Doc.	✗	Post-Gen	M.	L.	[35, 42, 45, 48, 57, 60, 84, 114]	[48, 57, 60, 84, 114]
CWRA [30]	Doc.	✗	Post-Gen	M.	L.	[30, 35, 42, 57]	[30]
Syntax Transf. [93]	Doc.	✗	Post-Gen	M.	M.	[42, 124]	-
Paraphrasing	Doc.	✗	Post-Gen	M.	L.	[1, 15, 33, 34, 42, 43, 45, 48, 56, 57, 59, 84, 85, 105, 113, 114, 118, 119, 124]	[33, 34, 42, 43, 48, 56, 57, 59, 84, 113, 119, 124]
Watermark Collision [61]	Doc.	✗	Post-Gen	M.	L.	[42, 57, 124]	-
Copy-Paste [43]	Doc.	✗	Post-Gen	L.	L.	[43, 105, 118]	[43, 118]
Spoofing Attack [86]	-	✓	Pre-Gen	H.	-	[42, 56, 57, 59, 124]	[56, 57, 59]
SCTS [108]	-	✓	Pre-Gen	H.	-	[42, 43, 124]	-
MIP [122]	-	✓	Pre-Gen	H.	-	[42, 43, 124]	-
Distillation [26]	-	✓	Pre-Gen	H.	-	[1, 42, 45]	-
WS [40]	-	✓	Pre-Gen	H.	-	[42, 43, 124]	-
LLM Fine-tuning [26]	-	✗	Pre-Gen	H.	-	[26]	-

For watermarking existing text (§3), word deletion is the most effective way. A deletion rate below 0.1 has minimal impact, but exceeding 0.3 can remove the watermark [114]. However, word deletion also has the greatest impact on overall semantics, potentially removing critical information. For watermarking LLMs (§4), some watermarking methods [42, 43, 57] depend on previous tokens to determine the current token's watermark status. Word-level attacks alter both the current and preceding tokens, leading to the watermark's removal.

Since word-level modifications cannot alter the text order, the modification space is limited. Significant modifications are likely to disrupt sentence semantics. Although word-level attacks perform well in some scenarios, due to obvious quality issues, these methods may not accurately simulate real-world conditions.

5.5.4 Word-level Attack during Text Generation. Due to the inevitable impact of word-level attacks on text quality, especially with extensive modifications, recent research has begun exploring word-level attacks during text generation. These methods target watermarking algorithms for LLMs. A notable example is the emoji attack [42], where the LLM generates emojis between each token, which are then removed post-generation.

For example, a user might request the LLM to insert "😊" between each word, resulting in sentences like "There 😊 are 😊 some 😊 apples 😊 here". If "apples" is watermarked and its detection relies on the prefix (the emoji "😊"),

removing the emojis changes the prefix from "apples" to "some". For algorithms that rely on prefix generation for watermarking [42, 56], this attack would completely remove the watermark.

However, the effectiveness of emoji attacks depends on the LLM's ability to follow instructions. Advanced LLMs like GPT-4 [71] and Claude can successfully execute emoji attacks, but less capable models might produce illogical outputs. Additionally, this attack is ineffective against watermarking methods that do not rely on previous tokens [45, 124].

5.5.5 Paraphrasing Attack. Word-level attacks modify individual words to change or remove text watermarks, having limited impact. In contrast, document-level attacks involve more extensive content and structure changes, with paraphrasing being the most common method.

Paraphrasing attacks offer significant modifications but are harder to implement than word-level methods. Early techniques, like back-translation strategies [114], can introduce errors and semantic drift. To enhance the quality of paraphrased text, specialized rewriting LLMs like Dipper [44] have been created. With the rise of ChatGPT [71], many now use the *gpt-3.5-Turbo* for paraphrasing, requiring just a simple prompt. Manual rewriting offers precise semantic retention and more natural expression but is costly [43], especially for large texts.

Different watermarking algorithms respond variably to paraphrasing attacks. Format-based watermarks [9, 79, 85, 89] are particularly vulnerable, as LLMs often replace homographs with standard tokens. For LLM watermarking algorithms, token sequence dependency is crucial for robustness. Detection methods that do not depend on token order are more resilient [124], while sequence-dependent algorithms like KGW [42] are more robust when they rely less on previous tokens. Converting token ID dependency to semantic dependency can also improve robustness [57, 84]. Notably, stronger watermarks and longer watermarked texts generally provide greater resistance to attacks. Notably, human writers are generally better at paraphrasing than LLMs [43], though individual abilities vary significantly.

5.5.6 Copy Paste Attack. Unlike paraphrasing attacks, which aim to modify the watermarked text, copy-paste attacks [43] do not modify the current document but insert watermarked text into human text. This type of attack weakens the watermark detector's effectiveness by reducing the proportion of watermarked text. Current work [43] shows when watermarked text accounts for only 10% of the total text, the attack effect usually surpasses most paraphrasing attacks. If the proportion increases to 25%, is is comparable to some paraphrasing attacks. Increasing text length can improve watermark detection reliability, especially in the context of copy-paste attacks.

Some watermark detection methods can identify copy-paste attacks. For example, Kirchenbauer et al. [43] mentions a window test that calculates the watermark level of a specific text area instead of the entire text. This method is specifically designed to effectively detect watermarked text inserted into existing text, making it suitable for countering copy-paste attacks.

5.5.7 Other Document-Level Attacks. In addition to paraphrasing and copy-paste attacks, there are other document-level untargeted attack methods like watermark collision [61], cross-lingual watermark removal attacks (CWRA) [30], and syntax transformation [93].

Watermark collision [61] examines if rewritten text can be detected with a watermark when the paraphraser LLM itself carries another watermark. Experiments show varying sensitivities among watermarking algorithms to collisions, but high-intensity collisions can erase all initial watermarks [42, 57, 124], leaving only the paraphraser's watermark. This indicates that watermark collision is a powerful attack method. Cross-lingual watermark removal attack (CWRA) [30] investigates whether watermarks remain when watermarked text is translated into another language. Current watermarking algorithms are not robust against such attacks. He et al. [30] developed the X-SIR algorithm, which shows some robustness against CWRA attacks. Syntax transformation [93] targets code

by making syntactic transformations without changing functionality to erase watermarks. Current watermarks struggle to remain robust under these attacks.

5.5.8 Fine-tuning Attacks. Fine-tuning attacks mainly target training-time watermarks (§4.3). When text watermark features are hidden in the LLM's parameters, subsequent fine-tuning can likely remove these features. Gu et al. [26] found that for global training-time watermarks, even minimal fine-tuning can remove the watermark features. Enhancing the robustness of training-time watermarks against subsequent fine-tuning remains a crucial research direction.

5.6 Targeted Watermark Attacks

Untargeted attacks are modifications or other forms of attacks without any knowledge of the watermark generation and detection methods. In contrast, targeted attacks occur when a malicious user attempts to crack the watermark generation method. Once the user has cracked the watermark generation method, they can easily remove existing watermarks [42] or forge new ones.

5.6.1 Threat Model. In targeted watermark attacks, the malicious user knows that an LLM contains a watermark. They may or may not know the specific type of watermark algorithm. Additionally, the malicious user has collected a large amount of watermarked text and may or may not have access to the watermark detector. The goal of the malicious user is to infer the watermark generation method. A robust watermark algorithm should be difficult to crack.

5.6.2 Targeted Watermark Attack for KGW. Most current targeted watermark attacks focus on specific algorithms [40, 86, 108, 122]. These attacks understand the general approach of the watermark algorithm but lack specific details, such as the exact division of red-green word lists in the KGW algorithm [42]. The goal is to infer this division.

Spoofing attacks [86] statistically analyze word frequencies under a fixed prefix in watermarked text compared to normal text. High-frequency words are considered "green," while low-frequency words are "red." This method is effective for the KGW algorithm with a window size of 1 and for unigram watermarks [124]. However, it struggles with larger window sizes.

Watermark Stealing (WS) [40] reverses watermarking rules by querying the watermark model's API. It splits watermarked and non-watermarked texts into small segments and analyzes their occurrence probabilities. Words appearing more frequently in watermarked texts are considered "green." WS achieves a spoofing attack success rate of over 80% and can extract more complex watermarking algorithms [43].

Self Color Testing-based Substitution (SCTS) [108] obtains color information through specific prompt generation. For instance, prompting the LLM to generate a string containing A and B, and noting which appears more frequently, determines green words. While it can identify some words, determining the entire red-green list is complex.

Mixed Integer Programming (MIP) [122] targets advanced watermarking schemes by stealing the green list and guiding optimization through systematic constraints. This method is more efficient than frequency-based methods. However, increasing the diversity and complexity of the green list can still hinder attackers from accurately identifying and replacing green markers.

5.6.3 Watermark Distillation. The previously introduced algorithms can only crack the red-green word list of the KGW watermark algorithm [42] and cannot be used for other types of watermarks. Gu et al. [26] studied the learnability of watermarks, i.e., whether an LLM can learn the watermarks by training directly on a large amount of watermarked text, thereby simulating the watermark.

They used methods of direct sampling-based learning and further distillation using logits. The experiment shows that, given sufficient training data, most current watermark algorithms can be learned, including KGW [42], Aar [1], and KTH [45] algorithms. However, this is limited to cases where the window size is relatively small. When the window size is sufficiently large (i.e., the watermark algorithm is sufficiently complex), these algorithms are still difficult to learn.

Notably, for clearer understanding, in Table 2 we outline all the watermark attacks mentioned in §5.5 and §5.6, along with their various characteristics and the algorithms evaluated on each attack.

5.7 Benchmarks and Tools

To facilitate the unified implementation and evaluation of text watermarks, some benchmarks and toolkits have been introduced, with WaterBench [101], WaterJudge [67], Mark My Words[78] and MarkLLM [72] being notable examples.

WaterBench [101] is a comprehensive benchmark designed to evaluate the detectability of watermarks in LLMs and their impact on LLM capabilities. It sets a fixed watermark strength (e.g., 0.95) for each algorithm to ensure consistency. WaterBench includes nine tasks in five categories, covering different input and output lengths. The benchmark shows that most watermarks perform well in detection, especially in long-output tasks, but have poorer performance in short-output tasks. All watermarks reduce generation quality to some extent, particularly in open-ended tasks.

WaterJudge [67], while focusing on similar evaluation aspects, places greater emphasis on evaluating the trade-off between watermark detectability and output quality. It uses the F1 score to measure detection performance and introduces an LLM-based evaluation approach to assess quality impact. This approach measures the average probability of an LLM preferring watermarked text over unwatermarked text in specific NLG tasks. WaterJudge compares different watermarking schemes by plotting them on a detectability-quality impact graph, providing a visual representation of this trade-off.

Mark My Words [78] evaluates watermarking schemes with a focus on watermark size and robustness under attacks. It defines watermark size as the number of tokens needed to detect the watermark at a 2% false positive rate. It also measures robustness against eight simple attacks designed to remove the watermark while preserving semantic similarity.

In addition to these benchmarks, more comprehensive toolkits have emerged. MarkLLM [72] is an open-source toolkit for LLM watermarking that provides a unified framework for implementing most existing LLM watermarking algorithms [1, 30, 42, 45, 49, 56, 57, 60, 124], ensuring ease of access through user-friendly interfaces. It also offers a comprehensive suite of evaluation tools covering detectability, quality, and robustness, as well as mechanism visualization to help the public better understand LLM watermarking technology.

6 Application for Text Watermarking

In preceding sections, we outlined the implementation methods of text watermarking technologies in the era of LLMs and detailed how to thoroughly evaluate these methods. As illustrated in Figure 10, this section delves into their real-world applications, focusing on two areas: copyright protection (§6.1) and AI-generated text detection (§6.2).

6.1 Copyright Protection

6.1.1 Text Copyright. Text copyright refers to the legal protection of original written content, ensuring creators have exclusive rights. Text watermarking technology helps safeguard copyright by detecting watermarks to identify the source of the text. The most common technology used for this purpose is format-based watermarking algorithms (§3.1), which do not alter the text content, an important factor for many creators.

Fig. 10. Application for Text Watermarking.

For instance, Taleby Ahvanooey et al. [94] uses layout attributes like word spacing and formatting elements such as text color and font for watermark insertion. Mir [65] introduced an invisible digital watermark for web content, using encrypted rules embedded in HTML. Additionally, tailored approaches are sometimes needed; Iqbal et al. [37] embedded watermarks in MS-Word documents using features like variables and bookmarks.

While current methods rely on format-based watermarking, the rise of LLMs suggests that integrating watermark algorithms with these models could be a promising direction for future research and applications in text copyright protection.

6.1.2 Dataset Copyright. With the rise and widespread application of deep learning technology, the copyright of datasets has become particularly important, and protecting datasets from unauthorized use has become a critical issue. The application of text watermarking technology in this field is mainly achieved through the backdoor watermark mentioned in 4.3. Specifically, this method embeds specific triggers and target behaviors into the dataset. When an LLM trained on this dataset encounters the corresponding triggers, it will exhibit the target behaviors [91, 92, 95].

6.1.3 LLM Copyright. For the copyright protection of LLMs, preventing extraction attacks [7, 74, 116] is crucial. In these attacks, malicious users train their own LLMs using a large amount of text generated by the original LLM. Some watermarking algorithms embed watermarks in the LLM's output to prevent such attacks.

For instance, He et al. [28] proposed embedding watermarks by replacing synonyms in the generated text, choosing specific "watermark words." However, this method alters word frequency, making the watermark easier to detect and remove. To address this, He et al. [29] used contextual features from parts of speech and dependency tree analysis for word replacements, keeping token frequency unchanged. Zhao et al. [125] further introduced watermarks during the logits generation process (§4.1) by embedding periodic signals in the LLM's output logits. These signals make the watermarks more robust and covert.

Additionally, Gu et al. [26] demonstrated that watermark algorithms like KGW [42], Aar [1], and KTH [45] have learnability. Moreover, data generated with these watermarks can train LLMs that carry the same watermarks. Similarly, Sander et al. [88] indicated that watermarks similar to KGW [42] give LLMs "Radioactive" properties.

However, using watermarking algorithms like KGW [42], Aar [1], and KTH [45] to resist model extraction attacks involves a trade-off between learnability and resistance to targeted attacks (§5.6). Generally, if an LLM is easy to learn from, it is also more susceptible to spoofing or stealing and can be forged more easily.

6.2 AI Generated Text Detection

As the capabilities of LLMs grow stronger, they may be misused in an increasing number of scenarios. Typical examples include academic integrity [46, 77, 104] and LLM-generated misinformation [13, 62, 126]. In the context of academic integrity, students might use these advanced LLMs to complete assignments, papers, or even participate in exams. In the context of LLM-generated fake news, LLMs might generate and rapidly spread false

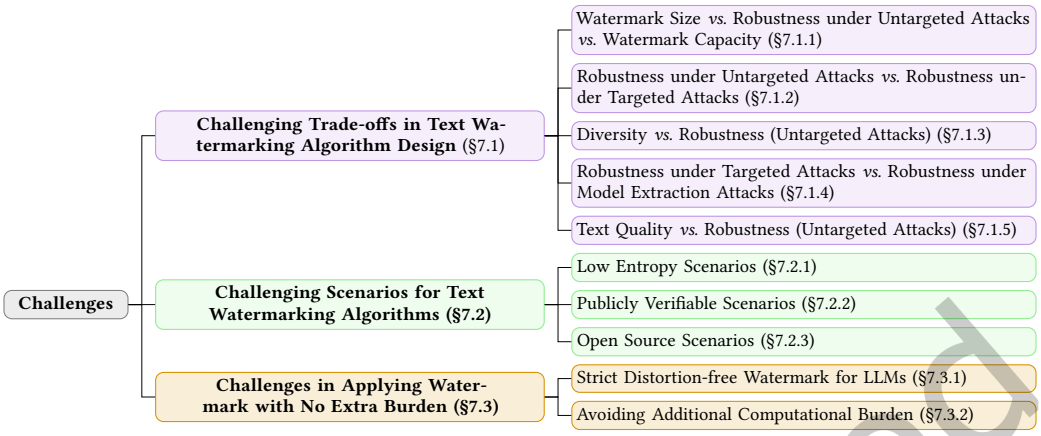


Fig. 11. Taxonomy of Challenges and Future Directions of Text Watermarking.

information. Current research indicates that some LLM misuses may not be totally mitigated [12, 58], making effective detection and tracking of LLM-generated text necessary solutions to these problems.

Theoretically, all algorithms that embed watermarks during logits generation (§4.1) and token sampling (§4.2) can be applied to these scenarios. For detecting AI-generated content, some practical online services currently adopt post-generation detection methods. These methods typically use features of LLM-generated text [66] or train classifiers to distinguish between LLM-generated text and human text [47]. There are also online text detection tools available, such as GPTZero [1]. This service distinguishes LLM-generated text from human text based on two features: text perplexity and burstiness. Since LLM text watermarks are added during the text generation process, there is currently no unified platform for detecting watermarked text. Building a unified watermark text detection platform should be an important future direction.

7 Challenges and Future Directions

Although detailed introductions to the methods, evaluations, and application scenarios of text watermarking have been provided in previous sections, numerous challenges remain in this field. As illustrated in Figure 11, these include challenging trade-offs in text watermarking algorithm design (§7.1), challenging scenarios for text watermarking algorithms (§7.2) and challenges in applying watermark with no extra burden(§7.3). These challenges will be discussed in detail below.

7.1 Challenging Trade-offs in Text Watermarking Algorithm Design

In Section 5, we explore various aspects of evaluating text watermarking algorithms. However, inherent conflicts often exist between these aspects, making it extremely difficult for an algorithm to excel in all areas. We will present the existing trade-offs and their underlying reasons, and discuss potential solutions for better balancing these aspects in future work.

7.1.1 Watermark Size, Robustness under untargeted attacks and Watermark Capacity. The trade-offs between watermark size (§5.1.3), robustness (§5.5), and watermark capacity (§5.1.2) are also important. Generally, improving any one of these three attributes will reduce the other two.

Specifically, as watermark capacity increases, more watermark information needs to be embedded in the text, which usually requires longer text lengths for the watermark to be detectable (less watermark size). Similarly,

[1] https://gptzero.me/

with higher watermark capacity, the robustness requirements of the watermarking algorithm also increase; that is, modifications to the text must not only fail to remove the watermark but also ensure that the embedded information is not altered.

The fundamental reason for contradictions among different perspectives lies in the limited suitable text space for text watermarking, usually determined by the text quality requirements. Specifically, according to Equation 1, the score difference between watermarked and non-watermarked texts under the quality evaluation function $\mathcal{R}$ should be less than a threshold β. However, the number of texts meeting this criterion is limited, denoted as $|t_\beta|$. Since the minimal impact on text quality is a crucial feature of text watermarking algorithms, there is an upper limit for $|t_\beta|$ for all watermarking algorithms. Given the watermark text space of $|t_\beta|$, we can further analyze the conflicts between different evaluation perspectives. These trade-offs are fundamental issues in text watermarking. Although some multi-bit watermarking algorithms [105, 118] can mitigate this problem to some extent, these algorithms achieve multi-bit functionality at the cost of significant robustness and watermark size.

7.1.2 Robustness under Untargeted and Targeted attacks. In sections 5.5 and 5.6, we introduced how to evaluate the robustness of watermarking algorithms under Untargeted and Targeted attacks, respectively. However, there is a trade-off between these two types of robustness: methods that achieve optimal robustness in Untargeted attacks usually do not perform well in Targeted attacks, and vice versa. For most algorithms, a key factor is how many previous tokens (window size) are relied upon to generate the watermark (e.g. red-green list in KGW [42]). For watermarking algorithms based on KGW [35, 42, 56, 57, 110] and Aar [1, 15], relying on more previous tokens makes the watermark generation details harder to be stolen, but it also makes the watermark easier to remove through text modification, and vice versa. For example, Zhao et al. [124] used a global red-green list split, which is considered very robust against various text modification attacks but is easily compromised by spoofing attacks. Fairoze et al. [19] used a complex hash algorithm, making the watermark difficult to steal, but it has low robustness against text modifications. This trade-off has been noted in many works [26, 40, 56, 57].

Some works attempt to mitigate this trade-off through more complex hash schemes, including self-hash [43], min-hash[43], and semantic hash [57]. However, these methods only mitigate the problem to a certain extent and do not fundamentally solve it. The trade-off between robustness in Untargeted and Targeted attacks can also be referred to as the trade-off between robustness and learnability [26] in some contexts [26].

7.1.3 Diversity and Robustness under Untargeted Attacks. Additionally, some work [26, 43] indicates that for LLM watermarking algorithms, there is a trade-off between output diversity and robustness against untargeted attacks. This trade-off is similar to the robustness trade-off under targeted and untargeted attacks. For some algorithms [1, 42], more complex patterns (large window size) during design lead to more diverse rules and outputs, while simpler rule patterns limit output diversity. However, this may do not apply to all algorithms. A watermarking algorithm based on previous token hash is more susceptible to this trade-off, whereas algorithms like KTH [45] using a fixed key list experience this trade-off to a lesser extent. However, the watermark detection efficiency of KTH-type algorithms decreases, mainly because the high complexity of multi time edit distance calculation. Perhaps this trade-off should also include the time complexity of detection.

7.1.4 Robustness under Targeted Attacks and Model Extraction Attacks. For LLM watermarking algorithms, there is a trade-off between robustness against target attacks and model extraction attacks. This was briefly mentioned in §6.1.3. The key issue is the learnability of the watermark [26]. A learnable watermark can help an LLM resist model extraction attacks (by ensuring the extracted model also has the watermark features), but it also makes it easier for malicious users to extract the watermark and perform target attacks. The core of this trade-off lies in the complexity of the watermark. Complex watermark rules are more resistant to target attacks, while simple watermark rules are better at resisting model extraction attacks. This is similar to the robustness trade-off under targeted and untargeted attacks mentioned earlier.

7.1.5 Text Quality and Robustness under untargeted Attacks. Generally, enhancing the robustness of LLM text watermarking algorithms against text modifications usually means increasing the watermark strength (e.g., δ in KGW [42] or τ in Aar). However, this often results in larger modifications to the text, which may affect text quality. Another approach is to introduce more redundant information, making it lengthy and repetitive. Thus, there is typically a trade-off between the robustness of LLM text watermarking algorithms under untargeted attacks and text quality. However, current research suggests that this trade-off might be alleviated by sacrificing some time complexity in the watermark generation [25] or detection [45].

7.1.6 Future Directions. Some work has been done to mitigate these trade-offs [43, 45, 57], but there is still a significant gap to achieving an optimal watermark across all aspects. Future work should better balance the above trade-offs from two perspectives: (1) Develop algorithms specifically for individual trade-offs. For example, SIR [57] algorithm targets robustness under both untargeted and targeted attacks. (2) Design entirely new watermarking paradigms [45] that inherently achieve better balance across all trade-offs.

7.2 Challenging Scenarios for Text Watermarking Algorithms

Current watermarking algorithms generally have good detectability and robustness. However, there are still some specific scenarios where LLM watermarking algorithms struggle to achieve excellent results. These mainly include low-entropy scenarios, publicly detectable scenarios, and open-source LLM scenarios.

7.2.1 Low Entropy Scenarios. In low-entropy scenarios like code [14] or table generation [53], embedding a highly detectable watermark is more challenging. This is mainly because these texts have strict syntactic or formatting requirements, resulting in small watermark capacity. A more in-depth explanation is that for low-entropy text, the upper limit of the watermark text space is lower, making it harder for watermark generation.

Some work has attempted to consider the impact of entropy in the watermark generation [49] or detection [60] process. However, they are still limited to token-level modifications. Future methods may need a stronger understanding of formatting or grammatical requirements, thereby designing semantically invariant format transformations to expand the watermark text space.

7.2.2 Publicly Verifiable Scenarios. We have already considered the Publicly Verifiable watermark scenario in §4.1.5. In this scenario, the watermark detector is publicly available to users, with the goal that it remains difficult for users to forge the watermark. This poses greater challenges for the design of watermark algorithms. Firstly, the entire watermark algorithm must have sufficient robustness against target attacks. Additionally, since the detector is public, there are more methods to perform target attacks on the algorithm, such as using the detector to reverse-engineer the generator [56]. Although there has been some exploration in this area [19, 56], these algorithms are typically limited by their robustness against untargeted attacks (as discussed in §7.1.4).

Future work should explore more robust publicly verifiable watermark algorithms and investigate more potential watermark attack methods in Publicly Verifiable scenarios to further advance understanding in this field.

7.2.3 Open Source Scenarios. For LLM watermarking technology, the most challenging scenario is in the open-source scenarios. Embedding text watermarking in an open-source LLM can only be done by incorporating it into the LLM's parameters during training. This can be achieved by training on watermarked text or using some inference-time watermarking for distillation training. However, the biggest challenge in this process is the robustness to further fine-tuning. Gu et al. [26] discovered that training-time watermarking techniques are weak in robustness to subsequent fine-tuning, and the watermark will inevitably be completely removed after sufficient fine-tuning iterations. Exploring watermarking solutions for open-source LLMs that are robust to further fine-tuning is an important direction for future research [127].

7.2.4 Future Directions. The three scenarios mentioned are all highly challenging. Although some current methods attempt to adapt watermarking algorithms to these scenarios [19, 26, 49, 56, 60], they still have various limitations. Future algorithms should first explore the performance upper limits in different scenarios to design better adaptation strategies. Additionally, they should investigate other potentially more challenging scenarios, as LLMs are rapidly evolving and will likely present many new challenges in the future.

7.3 Challenges in Applying Watermark with No Extra Burden

For a LLM watermarking algorithm, it is crucial and challenging to ensure that it does not introduce additional burdens. This primarily means that the LLM watermark must not cause any performance loss, nor should it add any computational burden, including in terms of time and space.

7.3.1 Strict Distortion-free Watermark for LLMs. Currently, many unbiased or distortion-free watermarking [1, 35, 45] algorithms claim not to affect LLM performance. While theoretically, these watermarking algorithms are unbiased, whether single-step unbiased decoding means no impact on capabilities is questionable. Firstly, Wu et al. [109] indicates that in multi-sentence generation scenarios, these algorithms cannot be considered unbiased. Additionally, these watermarks may potentially degrade the diversity of LLM-generated text [1]. Finally, LLMs may not only use sampling-based methods (e.g. nuclear sampling) to generate text; in some scenarios, they may use beam search to generate code [70]. Whether these unbiased or distortion free watermarks [1, 45] are applicable in such scenarios is also questionable. In summary, current distortion-free watermarking works under specific sampling assumptions. More research is needed to develop general distortion-free watermarking algorithms.

7.3.2 Avoiding Additional Computational Burden. While most watermarking algorithms have a low impact on LLM inference speed [42], they only involve some hash and random number generation operations, which, if recalculated at each step, may affect latency. Some methods involve pre-calculating hash results [26], but this may have significant space overhead when the LLM's vocabulary is large or depends heavily on previous tokens. Although some algorithms theoretically have low overhead, such as Unigram [124] and KTH [45], they often sacrifice robustness against targeted attacks or increase the time required for watermark detection.

7.3.3 Future Direction. In summary, to better facilitate the practical deployment of LLM watermarking algorithms, future algorithms should carefully consider their impact on LLM performance. When designing new watermarking methods, it is crucial to account for their real-world performance implications in large-scale LLM systems.

8 Conclusion

This survey thoroughly delves into the landscape of text watermarking in the era of LLMs, encompassing its implementation, evaluation methods, applications, challenges, and future directions.

Despite the progress made, several areas require further exploration. Future research should focus on creating advanced watermarking algorithms capable of withstanding novel attack types, especially where attackers have access to sophisticated tools and knowledge. Exploring watermarking in new applications like authenticity verification of AI-generated content in social media and journalism is crucial for maintaining the integrity and trustworthiness of digital content.

In summary, text watermarking in the era of LLMs is a rapidly evolving field. Its development will be critical in ensuring the responsible and ethical use of AI technologies.

References

[1] S. Aaronson and H. Kirchner. 2022. Watermarking GPT outputs. https://www.scottaaronson.com/talks/watermark.ppt.

[2] Sahar Abdelnabi and Mario Fritz. 2021. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 121–140.

[3] Mohammed Hazim Alkawaz, Ghazali Sulong, Tanzila Saba, Abdulaziz S Almazyad, and Amjad Rehman. 2016. Concise analysis of current text automation and watermarking approaches. *Security and Communication Networks* 9, 18 (2016), 6365–6378.

[4] Mikhail J Atallah, Victor Raskin, Michael Crogan, Christian Hempelmann, Florian Kerschbaum, Dina Mohamed, and Sanket Naik. 2001. Natural language watermarking: Design, analysis, and a proof-of-concept implementation. In *Information Hiding: 4th International Workshop, IH 2001 Pittsburgh, PA, USA, April 25–27, 2001 Proceedings 4*. Springer, 185–200.

[5] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. 65–72.

[6] Mahbuba Begum and Mohammad Shorif Uddin. 2020. Digital image watermarking techniques: a review. *Information* 11, 2 (2020), 110.

[7] Lewis Birch, William Hackett, Stefan Trawicki, Neeraj Suri, and Peter Garraghan. 2023. Model leeching: An extraction attack targeting llms. *arXiv preprint arXiv:2309.10544* (2023).

[8] Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. 2022. Bad characters: Imperceptible nlp attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1987–2004.

[9] Jack T Brassil, Steven Low, Nicholas F. Maxemchuk, and Lawrence O'Gorman. 1995. Electronic marking and identification techniques to discourage document copying. *IEEE Journal on Selected Areas in Communications* 13, 8 (1995), 1495–1504.

[10] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712* (2023).

[11] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175* (2018).

[12] Canyu Chen and Kai Shu. 2023. Combating misinformation in the age of llms: Opportunities and challenges. *arXiv preprint arXiv:2311.05656* (2023).

[13] Canyu Chen and Kai Shu. 2024. Can LLM-Generated Misinformation Be Detected?. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=ccxD4mtkTU

[14] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).

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

[16] Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672* (2022).

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4171–4186.

[18] Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding Back-Translation at Scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 489–500.

[19] Jaiden Fairoze, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Mingyuan Wang. 2023. Publicly Detectable Watermarking for Language Models. Cryptology ePrint Archive, Paper 2023/1661. https://eprint.iacr.org/2023/1661 https://eprint.iacr.org/2023/1661.

[20] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long Form Question Answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 3558–3567.

[21] Christiane Fellbaum. 1998. *WordNet: An electronic lexical database*. MIT press.

[22] Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. 2023. Three bricks to consolidate watermarks for large language models. In *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 1–6.

[23] Yu Fu, Deyi Xiong, and Yue Dong. 2024. Watermarking conditional text generation for ai detection: Unveiling challenges and a semantic-aware watermark remedy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 18003–18011.

[24] Evgeniy Gabrilovich and Alex Gontmakher. 2002. The homograph attack. *Commun. ACM* 45, 2 (2002), 128.

[25] Eva Giboulot and Furon Teddy. 2024. WaterMax: breaking the LLM watermark detectability-robustness-quality trade-off. *arXiv preprint arXiv:2403.04808* (2024).

[26] Chenchen Gu, Xiang Lisa Li, Percy Liang, and Tatsunori Hashimoto. 2024. On the Learnability of Watermarks for Language Models. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=9k0krNzvlV

[27] Batu Guan, Yao Wan, Zhangqian Bi, Zheng Wang, Hongyu Zhang, Yulei Sui, Pan Zhou, and Lichao Sun. 2024. Codeip: A grammar-guided multi-bit watermark for large language models of code. *arXiv preprint arXiv:2404.15639* (2024).

[28] Xuanli He, Qiongkai Xu, Lingjuan Lyu, Fangzhao Wu, and Chenguang Wang. 2022. Protecting intellectual property of language generation apis with lexical watermark. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 10758–10766.

[29] Xuanli He, Qiongkai Xu, Yi Zeng, Lingjuan Lyu, Fangzhao Wu, Jiwei Li, and Ruoxi Jia. 2022. Cater: Intellectual property protection on text generation apis via conditional watermarks. *Advances in Neural Information Processing Systems* 35 (2022), 5431–5445.

[30] Zhiwei He, Binglin Zhou, Hongkun Hao, Aiwei Liu, Xing Wang, Zhaopeng Tu, Zhuosheng Zhang, and Rui Wang. 2024. Can watermarks survive translation? on the cross-lingual consistency of text watermark for large language models. *arXiv preprint arXiv:2402.14007* (2024).

[31] Amr Hendy, Mohamed Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, and Hany Hassan Awadalla. 2023. How good are gpt models at machine translation? a comprehensive evaluation. *arXiv preprint arXiv:2302.09210* (2023).

[32] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The Curious Case of Neural Text Degeneration. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rygGQyrFvH

[33] Abe Hou, Jingyu Zhang, Tianxing He, Yichen Wang, Yung-Sung Chuang, Hongwei Wang, Lingfeng Shen, Benjamin Van Durme, Daniel Khashabi, and Yulia Tsvetkov. 2024. SemStamp: A Semantic Watermark with Paraphrastic Robustness for Text Generation. In *NAACL 2024)*. 4067–4082.

[34] Abe Bohan Hou, Jingyu Zhang, Yichen Wang, Daniel Khashabi, and Tianxing He. 2024. k-SemStamp: A Clustering-Based Semantic Watermark for Detection of Machine-Generated Text. *arXiv preprint arXiv:2402.11399* (2024).

[35] Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. 2024. Unbiased Watermark for Large Language Models. In *The Twelfth International Conference on Learning Representations*.

[36] Vojtěch Hudeček and Ondřej Dušek. 2023. Are Large Language Models All You Need for Task-Oriented Dialogue?. In *Proceedings of the 24th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 216–228.

[37] Muhammad Munwar Iqbal, Umair Khadam, Ki Jun Han, Jihun Han, and Sohail Jabbar. 2019. A robust digital watermarking algorithm for text document copyright protection based on feature coding. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 1940–1945.

[38] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rkE3y85ee

[39] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551* (2017).

[40] Nikola Jovanović, Robin Staab, and Martin Vechev. 2024. Watermark stealing in large language models. *arXiv preprint arXiv:2402.19361* (2024).

[41] Nurul Shamimi Kamaruddin, Amirrudin Kamsin, Lip Yee Por, and Hameedur Rahman. 2018. A review of text watermarking: theory, methods, and applications. *IEEE Access* 6 (2018), 8011–8028.

[42] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A Watermark for Large Language Models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 17061–17084. https://proceedings.mlr.press/v202/kirchenbauer23a.html

[43] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. 2024. On the Reliability of Watermarks for Large Language Models. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=DEJIDCmWOz

[44] Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Advances in Neural Information Processing Systems* 36 (2023).

[45] Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2024. Robust Distortion-free Watermarks for Language Models. *Transactions on Machine Learning Research* (2024). https://openreview.net/forum?id=FpaCL1MO2C

[46] Rahul Kumar, Sarah Elaine Eaton, Michael Mindzak, and Ryan Morrison. 2024. Academic integrity and artificial intelligence: An overview. *Second Handbook of Academic Integrity* (2024), 1583–1596.

[47] Zhixin Lai, Xuesheng Zhang, and Suiyao Chen. 2024. Adaptive ensembles of fine-tuned transformers for llm-generated text detection. *arXiv preprint arXiv:2403.13335* (2024).

[48] Gregory Kang Ruey Lau, Xinyuan Niu, Hieu Dao, Jiangwei Chen, Chuan-Sheng Foo, and Bryan Kian Hsiang Low. 2024. Waterfall: Framework for Robust and Scalable Text Watermarking. *arXiv preprint arXiv:2407.04411* (2024).

[49] Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. 2023. Who Wrote this Code? Watermarking for Code Generation. *arXiv preprint arXiv:2305.15060* (2023).

[50] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 7871–7880.

[51] Jingjing Li, Zichao Li, Tao Ge, Irwin King, and Michael R Lyu. 2022. Text Revision by On-the-Fly Representation Optimization. (2022), 10956–10964.
[52] Jingjing Li, Zichao Li, Lili Mou, Xin Jiang, Michael Lyu, and Irwin King. 2020. Unsupervised text generation by learning from search. *Advances in Neural Information Processing Systems* 33 (2020), 10820–10831.
[53] Tong Li, Zhihao Wang, Liangying Shao, Xuling Zheng, Xiaoli Wang, and Jinsong Su. 2023. A sequence-to-sequence&set model for text-to-table generation. In *Findings of the Association for Computational Linguistics: ACL 2023*. 5358–5370.
[54] CHEN Liang, Yatao Bian, Yang Deng, Deng Cai, Shuaiyi Li, Peilin Zhao, and Kam-Fai Wong. 2024. WatME: Towards Lossless Watermarking Through Lexical Redundancy. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*.
[55] Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958* (2021).
[56] Aiwei Liu, Leyi Pan, Xuming Hu, Shu'ang Li, Lijie Wen, Irwin King, and Philip S. Yu. 2023. An Unforgeable Publicly Verifiable Watermark for Large Language Models. arXiv:2307.16230 [cs.CL]
[57] Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. 2024. A Semantic Invariant Robust Watermark for Large Language Models. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=6p8lpe4MNf
[58] Aiwei Liu, Qiang Sheng, and Xuming Hu. 2024. Preventing and Detecting Misinformation Generated by Large Language Models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 3001–3004.
[59] Yepeng Liu and Yuheng Bu. 2024. Adaptive Text Watermark for Large Language Models. In *Forty-first International Conference on Machine Learning*. https://openreview.net/forum?id=7emOSb5UfX
[60] Yijian Lu, Aiwei Liu, Dianzhi Yu, Jingjing Li, and Irwin King. 2024. An Entropy-based Text Watermarking Detection Method. *arXiv preprint arXiv:2403.13485* (2024).
[61] Yiyang Luo, Ke Lin, and Chao Gu. 2024. Lost in Overlap: Exploring Watermark Collision in LLMs. *arXiv preprint arXiv:2403.10020* (2024).
[62] David Megías, Minoru Kuribayashi, Andrea Rosales, and Wojciech Mazurczyk. 2021. DISSIMILAR: Towards fake news detection using information hiding, signal processing and machine learning. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*. 1–9.
[63] Hasan Mesut Meral, Bülent Sankur, A Sumru Özsoy, Tunga Güngör, and Emre Sevinç. 2009. Natural language watermarking via morphosyntactic alterations. *Computer Speech & Language* 23, 1 (2009), 107–125.
[64] Fei Mi, Yitong Li, Yulong Zeng, Jingyan Zhou, Yasheng Wang, Chuanfei Xu, Lifeng Shang, Xin Jiang, Shiqi Zhao, and Qun Liu. 2022. PanGu-Bot: Efficient Generative Dialogue Pre-training from Pre-trained Language Model. *arXiv preprint arXiv:2203.17090* (2022).
[65] Nighat Mir. 2014. Copyright for web content using invisible text watermarking. *Computers in Human Behavior* 30 (2014), 648–653.
[66] Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *International Conference on Machine Learning*. PMLR, 24950–24962.
[67] Piotr Molenda, Adian Liusie, and Mark Gales. 2024. WaterJudge: Quality-Detection Trade-off when Watermarking Large Language Models. In *Findings of the Association for Computational Linguistics: NAACL 2024*. 3515–3525.
[68] Travis Munyer and Xin Zhong. 2023. Deeptextmark: Deep learning based text watermarking for detection of large language model generated text. *arXiv preprint arXiv:2305.05773* (2023).
[69] Ansong Ni, Srini Iyer, Dragomir Radev, Veselin Stoyanov, Wen-tau Yih, Sida Wang, and Xi Victoria Lin. 2023. Lever: Learning to verify language-to-code generation with execution. In *International Conference on Machine Learning*. PMLR, 26106–26128.
[70] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2023. CodeGen: An Open Large Language Model for Code with Multi-Turn Program Synthesis. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=iaYcJKpY2B_
[71] OpenAI. 2023. GPT-4 Technical Report. *ArXiv* abs/2303.08774 (2023). https://api.semanticscholar.org/CorpusID:257532815
[72] Leyi Pan, Aiwei Liu, Zhiwei He, Zitian Gao, Xuandong Zhao, Yijian Lu, Binglin Zhou, Shuliang Liu, Xuming Hu, Lijie Wen, et al. 2024. MarkLLM: An Open-Source Toolkit for LLM Watermarking. *arXiv preprint arXiv:2405.10051* (2024).
[73] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 311–318.
[74] Vaidehi Patil, Peter Hase, and Mohit Bansal. 2024. Can Sensitive Information Be Deleted From LLMs? Objectives for Defending Against Extraction Attacks. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=7erlRDoaV8
[75] Wenjun Peng, Jingwei Yi, Fangzhao Wu, Shangxi Wu, Bin Bin Zhu, Lingjuan Lyu, Binxing Jiao, Tong Xu, Guangzhong Sun, and Xing Xie. 2023. Are You Copying My Model? Protecting the Copyright of Large Language Models for EaaS via Backdoor Watermark. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 7653–7668.
[76] Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. 2022. Red Teaming Language Models with Language Models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 3419–3448.

[77] Mike Perkins. 2023. Academic Integrity considerations of AI Large Language Models in the post-pandemic era: ChatGPT and beyond. *Journal of University Teaching & Learning Practice* 20, 2 (2023), 07.
[78] Julien Piet, Chawin Sitawarin, Vivian Fang, Norman Mu, and David Wagner. 2023. Mark my words: Analyzing and evaluating language model watermarks. *arXiv preprint arXiv:2312.00273* (2023).
[79] Lip Yee Por, KokSheik Wong, and Kok Onn Chee. 2012. UniSpaCh: A text-based data hiding method using Unicode space characters. *Journal of Systems and Software* 85, 5 (2012), 1075–1082. https://doi.org/10.1016/j.jss.2011.12.023
[80] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).
[81] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.
[82] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.
[83] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 3982–3992.
[84] Jie Ren, Han Xu, Yiding Liu, Yingqian Cui, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. 2024. A Robust Semantics-based Watermark for Large Language Model against Paraphrasing. In *Findings of the Association for Computational Linguistics: NAACL 2024*. 613–625.
[85] Stefano Giovanni Rizzo, Flavio Bertini, and Danilo Montesi. 2016. Content-preserving text watermarking through unicode homoglyph substitution. In *Proceedings of the 20th International Database Engineering & Applications Symposium*.
[86] Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2023. Can AI-Generated Text be Reliably Detected? arXiv:2303.11156 [cs.CL]
[87] Fatih Şahin, Taner Çevik, and Mustafa Takaoğlu. 2021. Review of the Literature on the Steganography Concept. *International Journal of Computer Applications* 975 (2021), 8887.
[88] Tom Sander, Pierre Fernandez, Alain Durmus, Matthijs Douze, and Teddy Furon. 2024. Watermarking Makes Language Models Radioactive. *arXiv preprint arXiv:2402.14904* (2024).
[89] Ryoma Sato, Yuki Takezawa, Han Bao, Kenta Niwa, and Makoto Yamada. 2023. Embarrassingly Simple Text Watermarks. *arXiv preprint arXiv:2310.08920* (2023).
[90] Kurt Shuster, Mojtaba Komeili, Leonard Adolphs, Stephen Roller, Arthur Szlam, and Jason Weston. 2022. Language Models that Seek for Knowledge: Modular Search & Generation for Dialogue and Prompt Completion. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. 373–393.
[91] Zhensu Sun, Xiaoning Du, Fu Song, and Li Li. 2023. CodeMark: Imperceptible Watermarking for Code Datasets against Neural Code Completion Models. *arXiv preprint arXiv:2308.14401* (2023).
[92] Zhensu Sun, Xiaoning Du, Fu Song, Mingze Ni, and Li Li. 2022. Coprotector: Protect open-source code against unauthorized training usage with data poisoning. In *Proceedings of the ACM Web Conference 2022*. 652–660.
[93] Tarun Suresh, Shubham Ugare, Gagandeep Singh, and Sasa Misailovic. 2024. Is Watermarking LLM-Generated Code Robust?. In *The Second Tiny Papers Track at ICLR 2024*. https://openreview.net/forum?id=8PhI1PzSYY
[94] Milad Taleby Ahvanooey, Qianmu Li, Hiuk Jae Shim, and Yanyan Huang. 2018. A comparative analysis of information hiding techniques for copyright protection of text documents. *Security and Communication Networks* 2018 (2018).
[95] Ruixiang Tang, Qizhang Feng, Ninghao Liu, Fan Yang, and Xia Hu. 2023. Did You Train on My Dataset? Towards Public Dataset Protection with Clean-Label Backdoor Watermarking. *arXiv preprint arXiv:2303.11470* (2023).
[96] Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. 2023. Large language models in medicine. *Nature medicine* 29, 8 (2023), 1930–1940.
[97] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239* (2022).
[98] Mercan Topkara, Umut Topkara, and Mikhail J. Atallah. 2006. Words Are Not Enough: Sentence Level Natural Language Watermarking. In *Proceedings of the 4th ACM International Workshop on Contents Protection and Security* (Santa Barbara, California, USA) *(MCPS '06)*. Association for Computing Machinery, New York, NY, USA, 37–46. https://doi.org/10.1145/1178766.1178777
[99] Umut Topkara, Mercan Topkara, and Mikhail J Atallah. 2006. The hiding virtues of ambiguity: quantifiably resilient watermarking of natural language text through synonym substitutions. In *Proceedings of the 8th workshop on Multimedia and security*. 164–174.
[100] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).
[101] Shangqing Tu, Yuliang Sun, Yushi Bai, Jifan Yu, Lei Hou, and Juanzi Li. 2023. WaterBench: Towards Holistic Evaluation of Watermarks for Large Language Models. *arXiv preprint arXiv:2311.07138* (2023).

[102] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. 2017. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on international conference on multimedia retrieval*. 269–277.
[103] Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. 2022. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Chi conference on human factors in computing systems extended abstracts*. 1–7.
[104] Christoforos Vasilatos, Manaar Alam, Talal Rahwan, Yasir Zaki, and Michail Maniatakos. 2023. HowkGPT: Investigating the Detection of ChatGPT-generated University Student Homework through Context-Aware Perplexity Analysis. *arXiv preprint arXiv:2305.18226* (2023).
[105] Lean Wang, Wenkai Yang, Deli Chen, Hao Zhou, Yankai Lin, Fandong Meng, Jie Zhou, and Xu Sun. 2024. Towards Codable Watermarking for Injecting Multi-Bits Information to LLMs. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=JYu5Flqm9D
[106] John Wieting, Kevin Gimpel, Graham Neubig, and Taylor Berg-Kirkpatrick. 2022. Paraphrastic Representations at Scale. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.
[107] Bram Wouters. 2023. Optimizing watermarks for large language models. *arXiv preprint arXiv:2312.17295* (2023).
[108] Qilong Wu and Varun Chandrasekaran. 2024. Bypassing LLM Watermarks with Color-Aware Substitutions. *arXiv preprint arXiv:2403.14719* (2024).
[109] Yihan Wu, Ruibo Chen, Zhengmian Hu, Yanshuo Chen, Junfeng Guo, Hongyang Zhang, and Heng Huang. 2024. Distortion-free Watermarks are not Truly Distortion-free under Watermark Key Collisions. *arXiv preprint arXiv:2406.02603* (2024).
[110] Yihan Wu, Zhengmian Hu, Hongyang Zhang, and Heng Huang. 2023. DiPmark: A Stealthy, Efficient and Resilient Watermark for Large Language Models. *arXiv preprint arXiv:2310.07710* (2023).
[111] Frank F Xu, Uri Alon, Graham Neubig, and Vincent Josua Hellendoorn. 2022. A systematic evaluation of large language models of code. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*.
[112] Hengyuan Xu, Liyao Xiang, Xingjun Ma, Borui Yang, and Baochun Li. 2024. Hufu: A Modality-Agnositc Watermarking System for Pre-Trained Transformers via Permutation Equivariance. *arXiv preprint arXiv:2403.05842* (2024).
[113] Xiaojun Xu, Yuanshun Yao, and Yang Liu. 2024. Learning to watermark llm-generated text via reinforcement learning. *arXiv preprint arXiv:2403.10553* (2024).
[114] Xi Yang, Kejiang Chen, Weiming Zhang, Chang Liu, Yuang Qi, Jie Zhang, Han Fang, and Nenghai Yu. 2023. Watermarking Text Generated by Black-Box Language Models. *arXiv preprint arXiv:2305.08883* (2023).
[115] Xi Yang, Jie Zhang, Kejiang Chen, Weiming Zhang, Zehua Ma, Feng Wang, and Nenghai Yu. 2022. Tracing text provenance via context-aware lexical substitution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 11613–11621.
[116] Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. 2024. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing* (2024), 100211.
[117] KiYoon Yoo, Wonhyuk Ahn, Jiho Jang, and Nojun Kwak. 2023. 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)*. 2092–2115.
[118] KiYoon Yoo, Wonhyuk Ahn, and Nojun Kwak. 2023. Advancing Beyond Identification: Multi-bit Watermark for Language Models. *arXiv preprint arXiv:2308.00221* (2023).
[119] Ruisi Zhang, Shehzeen Samarah Hussain, Paarth Neekhara, and Farinaz Koushanfar. 2023. REMARK-LLM: A Robust and Efficient Watermarking Framework for Generative Large Language Models. *arXiv preprint arXiv:2310.12362* (2023).
[120] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068* (2022).
[121] Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen Mckeown, and Tatsunori B Hashimoto. 2024. Benchmarking Large Language Models for News Summarization. *Transactions of the Association for Computational Linguistics* 11 (2024), 39–57.
[122] Zhaoxi Zhang, Xiaomei Zhang, Yanjun Zhang, Leo Yu Zhang, Chao Chen, Shengshan Hu, Asif Gill, and Shirui Pan. 2024. Large Language Model Watermark Stealing With Mixed Integer Programming. *arXiv preprint arXiv:2405.19677* (2024).
[123] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. arXiv:2303.18223 [cs.CL]
[124] Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang. 2024. Provable Robust Watermarking for AI-Generated Text. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=SsmT8aO45L
[125] Xuandong Zhao, Yu-Xiang Wang, and Lei Li. 2023. Protecting language generation models via invisible watermarking. In *International Conference on Machine Learning*. PMLR, 42187–42199.
[126] Jiawei Zhou, Yixuan Zhang, Qianni Luo, Andrea G Parker, and Munmun De Choudhury. 2023. Synthetic Lies: Understanding AI-Generated Misinformation and Evaluating Algorithmic and Human Solutions. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (<conf-loc>, <city>Hamburg</city>, <country>Germany</country>, </conf-loc>) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 436, 20 pages. https://doi.org/10.1145/3544548.3581318

[127] Banghua Zhu, Norman Mu, Jiantao Jiao, and David Wagner. 2024. Generative AI security: challenges and countermeasures. *arXiv preprint arXiv:2402.12617* (2024).
[128] Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. 2020. Incorporating bert into neural machine translation. *arXiv preprint arXiv:2002.06823* (2020).

Received 28 January 2024; revised 22 August 2024; accepted 25 August 2024