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

AIWEI LIU* and LEYI PAN*, Tsinghua University, China
YIJIAN LU and JINGJING LI, The Chinese University of Hong Kong, China
XUMING HU and LIJIE WEN, Tsinghua University, China
IRWIN KING, The Chinese University of Hong Kong, China
PHILIP S. YU, University of Illinois at Chicago, United States



Text watermarking algorithms play a crucial role in the copyright protection of textual content, yet their capabilities and application scenarios have been limited historically. The recent developments in large language models (LLMs) have opened new opportunities for the advancement of text watermarking techniques. LLMs not only enhance the capabilities of text watermarking algorithms through their text understanding and generation abilities but also necessitate the use of text watermarking algorithms for their own copyright protection. 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 success rates, impact on text quality, robustness, and unforgeability; (3) potential application scenarios for text watermarking technology; (4) current challenges and future directions for development. This survey aims to provide researchers with a thorough understanding of text watermarking technology, thereby promoting its further advancement.

CCS Concepts: • **Computing methodologies → Natural language processing**; • **Security and privacy → Social aspects of security and privacy**.

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

**ACM Reference Format:**
Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Lijie Wen, Irwin King, and Philip S. Yu. 2024. A Survey of Text Watermarking in the Era of Large Language Models. 1, 1 (January 2024), 39 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 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 [30]. With the advancement of Large Language Models (LLMs), both the techniques and application scenarios

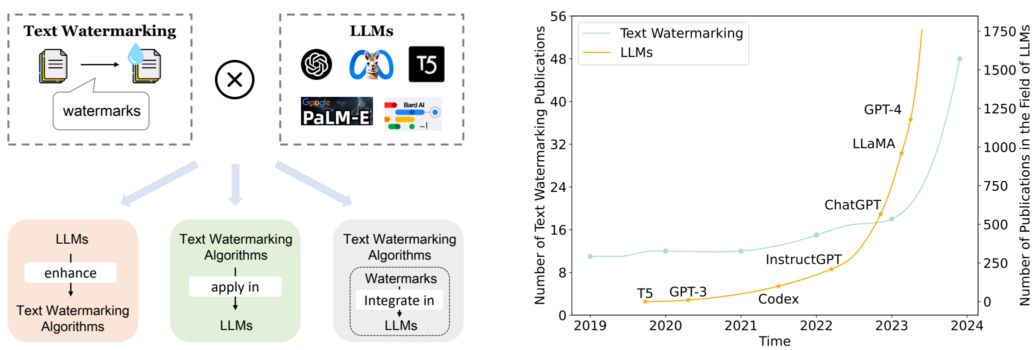
*Both authors contributed equally to this research.

Authors' addresses: Aiwei Liu, liuaw20@mails.tsinghua.edu.cn; Leyi Pan, ply20@mails.tsinghua.edu.cn, Tsinghua University, Beijing, China, 100084; Yijian Lu, luyijian@link.cuhk.edu.hk; Jingjing Li, lijj@link.cuhk.edu.hk, The Chinese University of Hong Kong, Hong Kong, China, 999077; Xuming Hu, hxm19@mails.tsinghua.edu.cn; Lijie Wen, wenlj@tsinghua.edu.cn, Tsinghua University, Beijing, China, 100084; Irwin King, king@cse.cuhk.edu.hk, The Chinese University of Hong Kong, Hong Kong, China, 999077; Philip S. Yu, psyu@cs.uic.edu, University of Illinois at Chicago, Chicago, United States, 60607.


XXXX-XXXX/2024/1-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

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 text watermarking algorithms that are more closely integrated with LLMs. 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. To help better understanding the mutually beneficial relationship between LLMs and text watermarking, this paper provides a survey of text watermarking techniques in the era of large language models.

In the subsequent content of this section, we will separately discuss why text watermarking benefits the application of LLMs (section 1.1), why utilizing LLMs can lead to the development of superior text watermarking algorithms (section 1.2), and the contributions of this survey along with the organization of the following sections (section 1.3).




(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. [100])

Fig. 1. Relationships between the development of text watermarking techniques and Large Language Models (LLMs).

## 1.1 Why is Text Watermarking Beneficial for LLMs?

In recent years, large language models (LLMs) have made significant progress in the field of natural language processing. As the parameter count of these large language models continues to increase, their ability to understand and generate language has also substantially improved. Notable examples include GPT [63], BART [38], T5 [65], OPT [97], LaMDA [79], LLaMA [82], and GPT4 [58]. These large language models have achieved excellent performance in a variety of downstream tasks, including machine translation [11, 22, 22, 106], dialogue systems [27, 51, 71, 79], code generation [56, 57, 84, 89], and other tasks [39, 40, 78, 99]. A recent work even suggests that GPT-4 is an early (yet still incomplete) version of an artificial general intelligence (AGI) system [7].

However, the extensive use of LLMs has also introduced a set of problems and challenges. Firstly, the rapid generation of high-quality text by LLMs can facilitate the rapid spread of false information [49]. Secondly, the issue of intellectual property related to large models is of vital importance. This includes the copyright of datasets [77] used for training large models and the addition of intellectual property rights to prevent the extraction of knowledge from the models

[103]. If effective tagging and detection methods for LLM-generated text could be implemented, it would significantly aid in mitigating the aforementioned issues. Text watermarking emerges as a promising solution to address these challenges. By embedding a unique, identifiable, and non-obtrusive marker (watermark) within the LLM-generated text, watermarking can enable the tracking and attribution of content produced by LLMs.

### 1.2 Why are LLMs Beneficial for Text Watermarking?

One of the main challenges in text watermarking is embedding watermarks without altering the original meaning or readability of the text. This requirement presents a significant challenge, as traditional text watermarking methods often struggle to modify the text without changing its semantics [3, 50, 80]. This is due to the need for text watermarking algorithms to have a strong understanding and control over the text's semantics. Here, Large Language Models emerge as a game-changer. With their advanced understanding of language semantics and context, LLMs enable more sophisticated text watermarking methods that seamlessly integrate watermarks with minimal compromise to the text's original meaning [1, 96]. This synergy allows for the development of watermarking techniques that are not only more effective but also more subtle, ensuring that the text remains as intended while still carrying the necessary watermark features.

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

Although text watermarking technology and large language models can effectively enhance each other, for instance, text generated by LLMs can be watermarked using text watermarking algorithms [6, 55, 62, 68, 90, 91, 93], or LLMs themselves can be utilized to embed watermarks in texts [1, 96]. Additionally, watermark algorithms can be directly incorporated during the text generation process of LLMs [31, 43, 44, 67, 88, 101]. However, up to date, no studies have attempted to comprehensively explore and understand text watermarking from a broader perspective. The current surveys on text watermarking mainly focus on techniques prior to the era of large language models [2, 30].

Therefore, in this work, we provide the first comprehensive survey of text watermarking algorithms in the era of large language models, covering the detailed definition of text watermarking algorithms and the interconnections between different kinds of text watermarking methods. Given the complexity and diversity of text watermarking technology, we have also detailed how to evaluate text watermarking algorithms from different perspectives, including success rate, robustness, impact on text quality and unforgeability. Additionally, we have introduced the current application scenarios of text watermarking algorithms including copyright protection, fake news detection, and academic integrity. This survey can provide researchers with a high-level understanding of text watermarking algorithms and a comparison of the similarities and differences between different text watermarking algorithms. Researchers who merely wish to employ text watermarking technology can select the appropriate algorithms and application scenarios based on the introduction provided in this survey.

**Organization of this survey**. The remainder of this survey is organized as follows: Section 2 introduces the definition of text watermarking and the essential properties of text watermarking algorithms. Section 3 and Section 4 discuss two significant types of text watermarking algorithms: text watermarking for existing text and text watermarking for LLMs. Section 5 elaborates on the different evaluation perspectives of text watermarking algorithms, including success rate, impact on text quality, robustness, and unforgeability. Section 6 presents the application scenarios of text watermarking algorithms, covering copyright protection, academic integrity and fake news detection. Section 7 analyzes the challenges still faced by current text watermarking research and explores future research directions. Finally, Section 8 concludes the survey.

## 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:

$$\mathcal{A}(\mathbf{x}, w) = \mathbf{t}. \tag{1}$$

The watermarked text, denoted as $\mathbf{t}$, can be derived in two ways: it may be a modified version of the input text $\mathbf{x}$, where $\mathbf{x}$ is the original text, or it can be a new text generated in response to $\mathbf{x}$, such as when $\mathbf{x}$ serves as a prompt for a Large Language Model. The watermark message, denoted as $w$, can be a zero-bit watermark, signifying merely its presence or absence, or a multi-bit watermark, embedding detailed, customized information. The term 'watermark payload' will be used henceforth to describe the quantity of information conveyed by the watermark message.

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 Key Characteristics of Text Watermarking Algorithms

To facilitate a unified understanding of the objectives in designing text watermarking algorithms, this section introduces the key characteristics that a watermarking algorithm should possess. These primarily include a high success rate of watermark detection, minimal impact on text quality, robustness of the detector against text modifications, and unforgeability.

**High success rate of watermark detection.** The high success rate of a watermark algorithm indicates that its detector $\mathcal{D}$ can accurately detect the generated watermark text $\mathbf{t}$. For a zero-bit watermark message, this accuracy is usually measured using binary classification. When $w$ consists of multiple bits, the evaluation commonly used is bit accuracy. If a watermarking algorithm maintains a high success rate at a large bit number, it implies that it possesses a high payload.

**Low impact on text quality.** We use $\mathcal{A}(\mathbf{x}, None)$ to denote the text generated without adding a watermark. If $\mathbf{x}$ is the text to be modified, the output will be $\mathbf{x}$ itself. If $\mathbf{x}$ is a prompt for LLM, it would be the text generated by LLM without a watermark. A watermark algorithm that does not significantly affect text quality will the following condition:

$$\forall w_i,\ \mathcal{R}(\mathcal{A}(\mathbf{x}, \text{None}), \mathcal{A}(\mathbf{x}, w_i)) < \delta \tag{2}$$

where $\mathcal{R}$ is a function evaluating text quality from multiple perspectives, as will be discussed in seciton 4. $\delta$ 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.** We use an operation $\mathcal{U}$ to denote the watermark removal operations, which will be detailed in seciton 4. If a watermarking algorithm is robust against watermark removal attacks, it should satisfy the following conditions:

$$\forall w_i, \forall \mathbf{t} = \mathcal{A}(\mathbf{x}, w_i),\ P(\mathcal{D}(\mathcal{U}(\mathbf{t})) = w_i) > \beta. \tag{3}$$

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

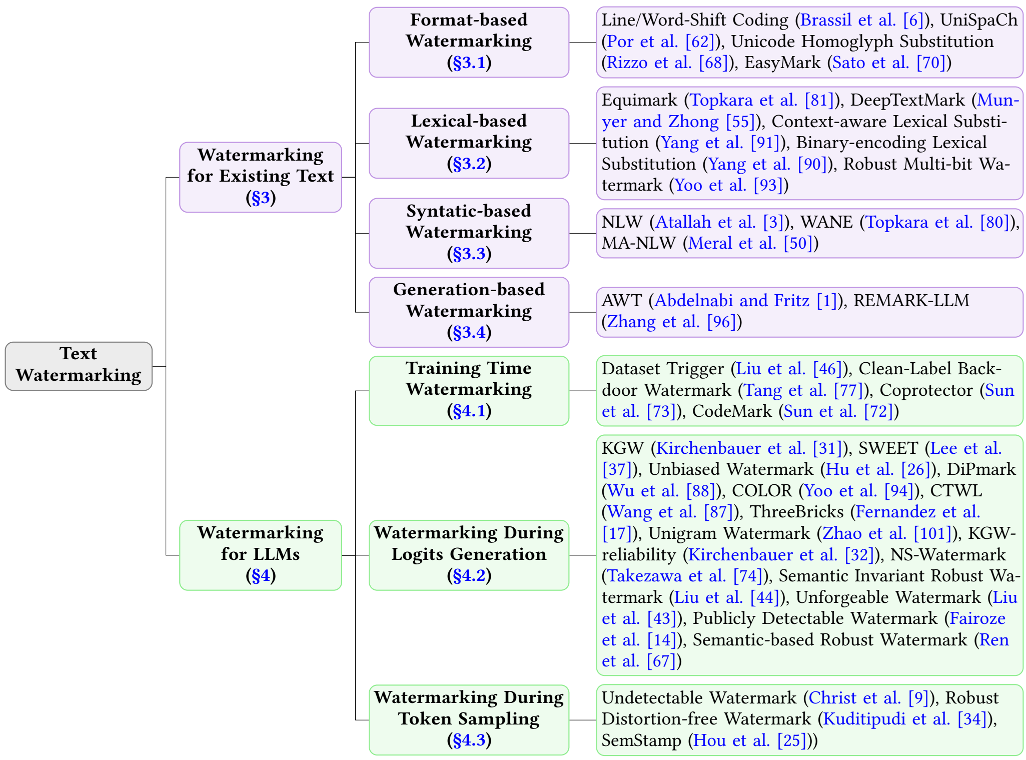



Fig. 2. The text watermarking methods can be broadly divided into two categories: Watermarking for Existing Text (section 3) and Watermarking for LLMs (section 4).

**Unforgeability.** Unforgeability refers to the difficulty for a third party to counterfeit text watermarks. Typically, if the watermark's generator $\mathcal{A}$ is acquired by an attacker, the watermark can certainly be forged. Therefore, the detailed definition of unforgeability is whether an attacker, unable to access the watermark's generator, can counterfeit the watermark. This usually divides into two scenarios: in one, the attacker cannot obtain the detector $\mathcal{D}$ and is limited to watermark detection, known as the private detection scenario. In the second scenario, the attacker has acquired the watermark's detector, referred to as the public detection scenario.

Although an ideal text watermarking algorithm should possess all four aforementioned characteristics, it is challenging to balance them. Enhancing one aspect might impact performance in another. In subsequent sections, we will delve deeper into how different watermark algorithms strike a balance among these characteristics.

## 2.3 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.

As illustrated in Figure 2, text watermarking algorithms can be broadly classified into two categories. The first category, **Watermarking for Existing Text**, focuses on embedding watermarks into pre-existing texts. Detailed in Section 3, this method typically employs semantically invariant transformations to incorporate watermarks seamlessly into the existing text.

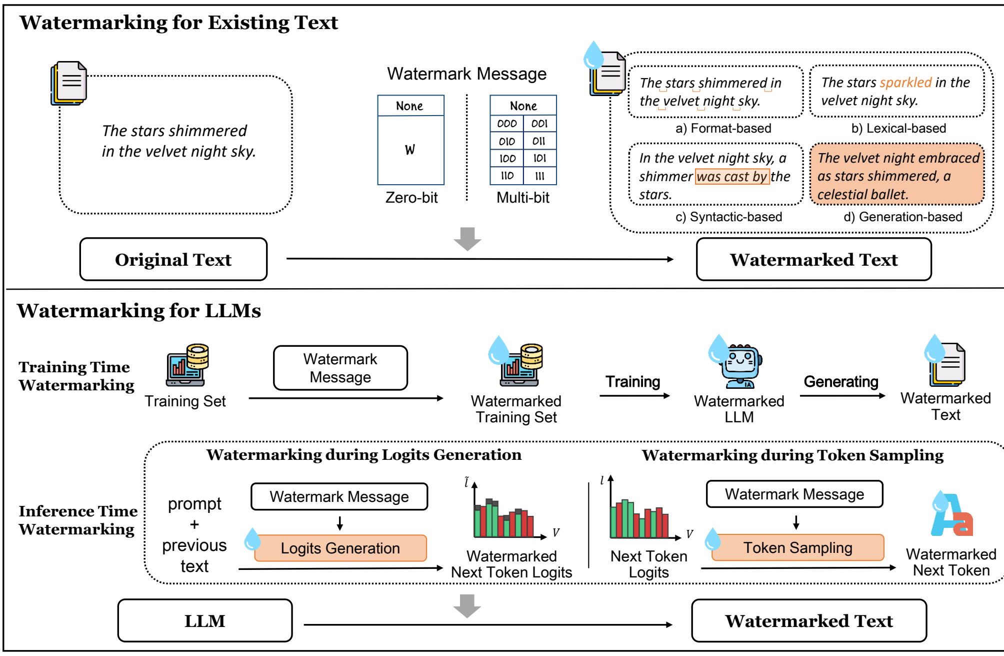



Fig. 3. A more illustrative explanation of various text watermarking methods. Watermarking for Existing Text (section 3) involves modifying existing text to embed watermarks, primarily through format-based approaches (section 3.1) such as white-space substitution, lexical-based approaches (section 3.2) such as synonym substitution, syntactic-based approaches (section 3.3) e.g. passivization, and generation-based approaches (section 3.4) that directly generated watermarked text through pretrained language models. Watermarking for LLMs (section 4) refers to embedding watermarks in Large Language Models, ensuring the text generated includes these watermarks, which can be implemented during training time (section 4.1), during logits generation (section 4.2), or during token sampling (section 4.3).

The second category, **Watermarking for Large Language Models (LLMs)**, involves alterations to the large language models. As we'll explore in Section 4, this approach either introduces specific features into the training dataset or modifies the text generation process of LLMs. In essence, it creates watermarked text **t** in response to an input prompt **x**.

Figure 3 offers a detailed illustration of these methods, emphasizing the nuances of current text watermarking techniques. Notably, both the 'watermarking during logits generation' and 'watermarking during token sampling' methods apply watermarks at the LLM inference stage, a process collectively referred to as 'inference time watermarking' in this context. The dashed line box under inference time watermarking represents the detailed process of how the watermarked LLM generates watermarked text.

## 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 (section 3.1), lexical-based watermarking (section 3.2), syntactic-based watermarking (section 3.3) and generation-based watermarking (section 3.4).

### 3.1 Format-based Watermarking

Format-based watermarking approaches are inspired by image watermarking technology [4]. It does not modify the content of the text but introduces changes to its format that are difficult for humans to detect, thereby embedding a watermark. For example, Brassil et al. [6] proposed line-shift coding and word shift-coding techniques, achieved by vertically shifting the positions of text lines or horizontally shifting the locations of words within text lines. Correspondingly, the watermark detection process involves measuring the distance between adjacent text line profiles or between adjacent word column profiles to detect shifts. However, this approach is limited to embedding watermarks in image-formatted text and cannot truly return a text string with an embedded watermark. Considering this, various watermarking methods that rely on the insertion or replacement of Unicode codepoints have been introduced. Por et al. [62] proposed a watermarking scheme named UniSpach, which inserts Unicode space characters into inter-sentence, inter-word, end-of-line and inter-paragraph spacings. A study by Rizzo et al. [68] presented a unicode homoglyph substitution text watermarking method. It exploits the fact that text symbols that are similar in appearance could have different Unicode codepoints. For instance, both U+0043 and U+216d visually represent letter 'C', while U+004c and U+216c appear as letter 'L'. Following this, a family of simple watermarks named EASYMARK [70] is proposed recently, composed of three different methods: WHITEMARK, VARIANTMARK and PRINTMARK. Specifically, WHITEMARK replaces a whitespace (U+0020) with another codepoint of a whitespace (e.g. U+2004). VARIANTMARK leverages variation selectors of Unicode to embed hard-to-perceive format into CJK texts. PRINTMARK, coping with printed texts, uses ligature or whitespaces with slightly different lengths to embed watermark messages. Correspondingly, the watermark detection process involves searching for and counting the certain codepoints that have been inserted within the text. As these watermarking methods relied on the richness of Unicode encoding, their watermark payload are often quite large.

However, although these format-based watermarking methods allow for the simple and effective embedding of large payload watermarks in text without altering its specific content, modifications to the format can be easily spotted in certain scenarios, as mentioned by Por et al. [62] in the case of DASH attack. As a result, these specially designed formats could be effortlessly removed through canonicalization [5], such as reseting line spacing, searching and replacing certain codepoints through the entire text, etc. The spotted formats might also be used to forge watermark, further leading to the failure of effective detection.

### 3.2 Lexical-based Watermarking

Format-based approaches only modify the surface format of the text, making them easily spotted and, consequently, more vulnerable to targeted removal by reformatting. Therefore, it becomes imperative to explore alternative methods that enable deeper insertion of watermarks within text. Several studies applied word-level modifications by replacing selected words with their alternatives without changing the sentence syntax structure [16, 55, 81, 90, 91, 93]. We refer to these methods as lexical-based watermarking approaches. Topkara et al. [81] presented a synonym substitution text watermarking approach, using the linguistic database WordNet[16] as its synonym dictionary. The watermark detection process essentially replicates the watermark message embedding procedure, with the distinction that inverse rules are employed during the message extraction phase. In order to conduct semantic modeling more effectively, Munyer and Zhong [55] utilized a pretrained Word2Vec model instead of WordNet to find alternatives. In particular, they converted the selected words into Word2Vec vectors and collected the n-nearest vectors to form replacement word set.

Noticeably, they trained a binary classifier as watermark detector, using pretrained BERT model and transformer blocks as neural network components.

However, the aforementioned watermarking approaches, which depended on context-independents synonym substitution (WordNet & Word2Vec), tend to overlook the context of the target words when generating substitute candidates. This oversight may result in a failure to preserve the overall semantics of the sentence, consequently diminishing the quality of the text. In response to this issue, context-aware lexical substitution is introduced into text watermarking techniques. Yang et al. [91] proposed a novel BERT-based infill model to generate lexical substitution candidates, taking the overall sentence's meaning into account. The watermark detection algorithm mimics the watermark generation process, first locating words containing the embedded watermark message, then generating substitute candidates, and finally applying the inverse embedding rules to extract the watermark message. To simplify the watermark detection process, Yang et al. [90] developed a watermarking scheme by first computing a random binary encoding for each word, then replacing the words representing bit-0 with context-based synonyms that represent bit-1. Since the encoding computed for non-watermarked text adhere to a Bernoulli distribution and the distribution is altered during the watermarking process, statistical tests can be employed directly to detect the presence of the watermark. To further improve the robustness of watermark algorithms against watermark removal attacks, a study by Yoo et al. [93] fine-tuned BERT-based infill model with keyword-preserving and syntactically-invariant corruptions, which achieved a state-of-the-art robustness compared to previous approaches.

Lexical-based watermarking approaches embed watermarks by substituting synonyms in the text. However, the scope for semantically non-altering synonym replacements that do not affect the text structure is likely limited. Consequently, the capacity for the watermarks in lexical-based approaches is restricted, often necessitating a trade-off with text quality.

### 3.3 Syntactic-based Watermarking

The lexical-based approaches attempted to embed watermark messages by replacing certain words without altering the syntax structure of the sentence. However, methods relying solely on lexical substitution are likely to lack robustness against simple watermark removal attacks, such as random synonym replacements. In the light of this, some studies attempt to embed watermarks in text in a manner that is more challenging to remove, specifically by altering the syntax structures of the text. These methods are known as syntactic-based watermarking approaches. Atallah et al. [3] introduced three typical syntax transformations–*Adjunct Movement*, *Clefting* and *Passivization*–to embed watermark messages, where:

- *Adjunct Movement* refers to the shifting of adjuncts to different positions within a sentence. For instance, the adverbial phrase 'often' can be placed in multiple locations in the sentence *The dog often chased the cat.*
- *Clefting* is a transformational process used to emphasize a specific part of a sentence, often the subject. For example, the sentence *The dog chased the cat* can be transformed into *It was the dog that chased the cat* to highlight *the dog.*
- *Passivization* involves converting active voice sentences with transitive verbs into the passive voice. For instance, the active sentence *The dog chased the cat* can be transformed into the passive form *The cat was chased by the dog.*

Each transformation type corresponds to a specific message bit. For instance, Adjunct Movement corresponds to 0, Clefting stands for 1, and Passivization represents 2. During watermark detection, the original and the modified text are both transformed into syntax trees. The syntax structures are subsequently compared sentence by sentence to extract watermark messages. Following this, Topkara et al. [80] expanded the watermark payload by additionally introducing two syntax transformation type: Activization and Topicalizaiton. Moreover, effort has not been solely made about adding watermarks into English corpus. Meral et al. [50] investigated 20 morphosyntactic tools in Turkish. They noted that languages with high suffixation and agglutination, like Turkish, provide ample opportunities for syntactic-based watermarking.

Syntactic-based watermarking approaches can embed watermarks into existing texts in a relatively concealed manner. However, this method relies significantly on the grammatical rules of a language, potentially necessitating customization for each language. In certain texts, frequent syntactic alterations might also impact the original style and fluency of the text.

### 3.4 Generation-based Watermarking

The aforementioned methods have indeed made significant strides in the field of text watermarking. However, these methods are still quite reliant on specific rules, which may lead to unnatural modifications in some contexts. On one hand, the unnatural modifications might lead to degradation of text quality. On the other hand, if these clues are observed by human attackers, there is a higher likelihood of them to design watermark removal attacks or attempt to forge watermarks deliberately and specifically. A groundbreaking advancement would be generating watermarked text directly from the original text and the watermark message. With the rapid development of pretrained language models, such techniques are gradually becoming feasible. In the realm of generation-based approaches, the encoded original text and the watermark message are typically fed into a pretrained language model, which subsequently generates the watermarked text end-to-end.

Abdelnabi and Fritz [1] introduced an end-to-end watermarking scheme named AWT. It harnesses a transformer encoder for encoding the original sentence. It then combined the sentence embedding and message embedding, feeding this composite input into a transformer decoder to derive the watermarked text. During the watermark detection process, the watermarked text is fed into transformer encoder layers to obtain the secret message. Building on top of AWT, Zhang et al. [96] noticed the gap between the dense distributions of the watermarked text and the sparse distributions of the one-hot watermark message encodings. To bridge this gap, they presented a watermarking method named REMARK-LLM. Likewise, its watermarking process inserts watermark message into the original text via a pretrained language model. Noticeably, a reparameterization step is introduced to transform the distribution of the generated watermarked tokens into a sparser distribution using Gumbel-Softmax [29]. Then a decoder based on the transformer architecture is employed to extract the concealed messages from these embeddings. The reparameterization step allows REMARK-LLM to successfully embed two times more signatures into original text compared to prior art AWT and at the meantime maintain detection effectiveness, making a significant progress in expanding watermark payload.

## 4 WATERMARKING FOR LLMS

In the above section, we discussed watermarking methods for existing text. With more and more texts directly generated by large language models, studying text watermarking techniques for large models has become a trend. Unlike the method of modifying existing text to add a watermark, the *watermarking for LLMs* technology directly enables LLM-generated text to contain a watermark. Specifically, given a watermark message $w$ and a prompt $\mathbf{x}$, the process of *watermarking for LLMs* is defined by the following expression:

$$\mathcal{A}(\mathbf{x}, w) = M_w(\mathbf{x}) = \mathbf{t}. \tag{4}$$

To facilitate explanation, we assume that the watermarked text is directly generated by a language language model $M_w$ with an embedded watermark message.

To provide a better understanding of how to add a watermark to a Large Language Model, 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.** The process involves training a large language model M using dataset D. The specific objectives of training can vary depending on the application context. Currently, the most prevalent training objective employed is next token prediction [64].
- **Step2: logits generation.** Given a trained large language model M, a prompt $\mathbf{x}$, and a sequence of previously generated tokens $\mathbf{t}^{0:(i-1)}$, the LLM generates a probability distribution over the next token $\mathbf{t}^{(i)}$ in the vocabulary $\mathcal{V}$, represented as logits $\mathbf{l}^{(i)}$:
$$\mathbf{l}^{(i)} = M(\mathbf{x}, \mathbf{t}^{0:(i-1)}). \tag{5}$$
- **Step3: token sampling.** The next token $\mathbf{t}^{(i)}$ is sampled from the logits $\mathbf{l}^{(i)}$ which could be achieved using nucleus sampling [24], choosing the token with the highest probability (greedy decode), or using other decode algorithms such as beam search to select a list of tokens with the highest probability. Here we use $S$ to denote the token sampling process:
$$\mathbf{t}^{(i)} = S(\text{softmax}(\mathbf{l}^{(i)})). \tag{6}$$

Through these steps, the large language model M can produce a single token $\mathbf{t}^{(i)}$. To generate multiple tokens, we could simply repeat the logits generation and token sampling process iteratively.

Considering that there are three important steps during the utilization of LLM to generation text, watermarking for LLMs techniques could also be divided into three distinct types correspondingly. Specifically, we refer to the three distinct types as training time watermarking, watermarking during logits generation, and watermarking during token sampling. These three watermarking techniques will be elaborated in sections 4.1, 4.2, and 4.3, respectively.

## 4.1 Training Time Watermarking

The objective of training time watermarking is to embed a watermark message, denoted as $w$, into a LLM during its training phase. This process is typically accomplished by embedding the watermark message $w$ into the training dataset. Specifically, embedding a watermark message into a given dataset $D$ typically follows the following steps. The process begins with the extraction of a subset $D_s$ from the dataset. A watermark message is then embedded onto this subset through a watermark embedding function $W$, producing the corresponding watermarked subset $\widetilde{D}_s$, denoted as $\widetilde{D}_s = W(D_s, w)$. Consequently, the watermarked dataset $D_w$ is defined as the union of the original dataset $D$ and the watermarked subset $\widetilde{D}_s$, minus the non-watermarked subset $D_s$, as expressed by the following formula:
$$D_w = (D \setminus D_s) \cup \widetilde{D}_s \tag{7}$$
Subsequently, the large language model (LLM) trained on the watermarked dataset $D_w$ is embedded with the watermark message $w$, denoted as $M_w$. Typically, $M_w$ exhibits characteristics of the watermark message $w$ when operating on datasets with a distribution similar to $\widetilde{D}_s$, enabling the wateramrk detection. In this process, the most critical aspect is the design of the watermark embedding function $W$, specifically how to transform $D_s$ into $\widetilde{D}_s$. Presently, methods for designing $W$ predominantly draw on the concept of a backdoor attack. Within the training set $\widetilde{D}_s = \{(x, y)\}$, for an input $x$, a specific trigger is introduced to embed a recognizable feature which manifests in the corresponding output and is detected in the subsequent verification process. Depending on whether the introduced trigger disrupts the original label $y$, the current methods can be broadly classified into two categories.

The first category of methods introduces a trigger to x, compromising the corresponding label y for that segment of x. For instance, Liu et al. [46] proposed a training time watermark algorithm for text classification tasks, randomly selecting a subset of text data to insert triggers at the character-level, word-level, or sentence-level. These triggers are distinct features, and they uniformly changed the labels of these data to a specific $y_t$. Sun et al. [73] implemented a similar approach for code generation, applying word-level or sentence-level triggers to text, and employing methods like code corrupting to disrupt the associated code. Although these techniques are effective for detecting models using trigger-inclusive text, they compromise part of the dataset's labels, potentially degrading the model's inherent capabilities.

To address this issue, methods that produce low distortion on labels are required. Tang et al. [77] initially adopt adversarial learning to identify examples within a certain category C that are prone to be wrongly classified by models and then add triggers to these samples without altering their original labels. For code generation tasks, Sun et al. [72] conduct semantically invariant transformations of the code, such as employing different forms of syntactic sugar. This allows for the matching and detection of triggers in text with varying code styles. Currently, the objective of training time watermarking is to protect the copyright of datasets from unauthorized use.

Although Training Time Watermarking can embed watermark messages into large language models, it has several distinct disadvantages. Firstly, the watermarked LLM $M_w$, can only produce watermarked outputs for a subset of inputs. Secondly, the watermark payload is considerably limited, typically only indicating the presence of a watermark and not containing more extensive information. Thirdly, the watermark message is challenging to modify: altering the watermark message $w$ often requires retraining the model, which is a costly process. Therefore, the application of Training Time Watermarking remains quite limited.

## 4.2 Watermarking during Logits Generation

Watermarking during logits generation refers to the insertion of a watermark message $w$ into the logits (i.e., probabilities of tokens in the vocabulary $\mathcal{V}$) generated by large language models. This approach does not necessitate alterations to the parameters of the LLM $M$, rendering it more flexible and cost-effective compared to training time watermarking methods.

In the scenario of watermarking during logits generation, the watermarking algorithm $\mathcal{A}$ modifies the logits produced by the LLM. Specifically, the watermark's message $w$ is embedded into the logits generated by LLM. The watermarked logits $\widetilde{\mathbf{l}^{(i)}}$, could be calculated using the following formula:

$$\widetilde{\mathbf{l}^{(i)}} = \mathcal{A}(M(\mathbf{x}, \mathbf{t}^{0:(i-1)}), w) = M_w(\mathbf{x}, \mathbf{t}^{0:(i-1)}), \tag{8}$$

where we assume the watermarked logits $\widetilde{\mathbf{l}^{(i)}}$ is generated by a watermarked LLM $M_w$.

Kirchenbauer et al. [31] proposed the first watermarking method based on LLM logits modification, referred to as KGW. There, the watermark generator randomly splits the vocabulary set into a red list and a green list at each position depending on the previous token using a hash function. As illustrated in Figure 4, when $M_w$ generates the $i^{th}$ token, a small bias $\delta$ is added to the logits of the tokens in the green list. Let $G$ represent the green list and $R$ stands for the red list. Then the logits value of token $v_j$ at position $i$ would be calculated as follows:

$$\widetilde{\mathbf{l}_j^{(i)}} = M_w(\mathbf{x}, \mathbf{t}^{0:(i-1)}) = \begin{cases} M(\mathbf{x}, \mathbf{t}^{0:(i-1)})[j] + \delta, & v_j \in G \\ M(\mathbf{x}, \mathbf{t}^{0:(i-1)})[j], & v_j \in R \end{cases} \tag{9}$$

As the watermark algorithm favors the logits of green tokens, the watermarked text is expected to contain a higher proportion of green tokens compared to the unwatermarked text. Consequently, the watermark detector assesses whether a text is watermarked by first utilizing the hash function

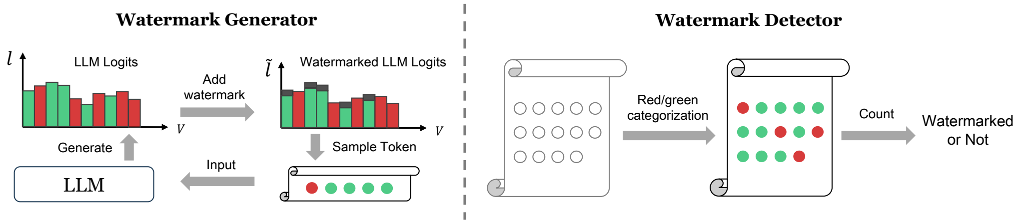



Fig. 4. A more illustrative description of the KGW [31] algorithm.

to categorize each token as red or green sequentially. Following this, it calculates the green token proportion using the z-metric. If the green token proportion surpasses a specific threshold, the text is classified as watermarked.

Although the KGW watermark detection method demonstrated exceptional performance in its test scenarios, achieving a false positive rate (human text misclassified as watermark text) of less than $3 \times 10^{-3}\%$, and a false negative rate (watermark text not detected) of less than 1%, there remain numerous highly challenging scenarios in real-world applications that necessitate specialized optimization and design of the watermark algorithm to effectively address them. Below, we list four representative scenarios and thoroughly introduce the improvements and explorations made to watermark algorithms in these contexts, which are illustrated in Figure 5.

4.2.1 Watermarking low-entropy text. Low-entropy scenarios are situations such as code generation and formatted document generation, where the generated text is relatively deterministic. Entropy serves as a quantification of textual uncertainty, which is calculated as follows:

$$H^{(i)} = -\sum_{j=1}^{|\mathcal{V}|} P_j^{(i)} \log P_j^{(i)}, \tag{10}$$

where $P_j^{(i)}$ denotes the probability value of token $v_j$ at position $i$. A decrease in entropy corresponds to a heightened level of certainty within the produced text. In real applications, it is also necessary to add watermark when using LLM to generate low-entropy text. Under low-entropy scenarios, significant modifications are not allowed, since it may notably degrade text quality. Therefore, watermarking and detecting watermark in low-entropy text pose a challenging task.

To address this problem, Lee et al. [37] suggested calculating the entropy before adding preference to logits of green tokens (i.e. set G in Equation 9). If the entropy $H^{(i)}$ is lower than a threshold $H$, the logits vector remains unmodified. Similarly, Wang et al. [87] employed a balance-marking strategy, constraining the choice of vocabulary to a subset where the cumulative model log probabilities exceed a certain percentage. This approach effectively allows for the watermarking on only high-entropy tokens while bypassing low-entropy tokens. For instance, consider a specific case where there is only one word candidate; in such a scenario, the vocabulary subset is limited to this single candidate. The above methods [37, 87], by considering the influence of text entropy during the watermarking process, achieves a relatively low impact on the quality of the text. However, this method is still ineffective when the number of high-entropy tokens in the text is low. Adding watermarks to low-entropy text remains a challenge.

4.2.2 Watermark with multi-bit payload. The KGW[31] watermark algorithm introduced by Equation 9, could only determine the presence of a watermark and no additional information, categorizing it as a zero-bit payload watermark. However, practical applications often demand that watermarks carry additional information such as copyright details, timestamps, or specific identifiers. This necessitates watermarking techniques that not only detect the presence of watermarks but also extract meaningful information, which could be categorized as multi-bit payload watermark.

To achieve this, one possible solution is to establish multiple splits for dividing the vocabulary into a red list and a green list. Specifically, we can set N types of splits for the vocabulary, such as $[(G_1, R_1), \ldots, (G_N, R_N)]$, where each split could performing LLM watermarking following the way introduced in Equation 9. Each split corresponds to a specific watermark message, endowing the watermark with a $log_2N$-bit payload. For instance, Wang et al. [87] permits the user to input a watermark message $m$ with $log_2N$ bits. Subsequently, based on the hash value of this message $m$, the vocabulary is divided. However, during detection phase, this method requires iterating through all $N$ possible messages, calculating the correlation between the text and the red/green splits corresponding to each message respectively, thus is not efficient. To mitigate this issue, Fernandez et al. [17] proposed the Cyclic Shift method, where distinct messages are generated by cyclically shifting an initial message. This approach reduces redundant computational steps and enhances detection efficiency through parallel processing.

However, these methods, due to the necessity of iterating each possible message, encounter heightened computational complexity with the growing watermark payload. To address this, Yoo et al. [94] proposed allocating different bits of the message to various positions in the watermark text, allowing independent detection of different bit altogether in a single detection round. Then the information corresponding to each bit is concatenated to obtain the final message. Specifically, the message is modeled as a sequence $m = \Sigma^b$ where $\Sigma = \{0, 1\}$. The process of dividing the vocabulary at each generation step is then split into two phrases: the first involves choosing a position in the message, $\Sigma^b[i]$, based on the existing random seed, and the second involves dividing the vocabulary according to this $\Sigma^b[i]$ and random seed. This approach significantly improves efficiency by enabling the parallel detection of bit information. Furthermore, Yoo et al. [94] also suggest dividing the vocabulary into more parts, so that each position $\Sigma^b[i]$ in $m$ can contain more information, further enhancing the payload of the text watermark.

4.2.3 Preventing watermark removal attack. As discussed in section 2, an effective watermarking algorithm must possess sufficient robustness against watermark removal attacks, ensuring that the watermark text remains detectable post-attack. These attacks typically involve modifications to the text without altering its semantic content, which will be introduced in section 5.3. Although the KGW algorithm [31] demonstrated some robustness to watermark removal attacks in their experiments, there remains room for improvement in its robustness.

In particular, for the watermarking during logits generation methods [31, 32, 43, 44, 101], the most significant factor affecting robustness is how to determine the modifications to logits. More specifically, how to divide the red-green list mentioned in the KGW method [31]. In the original KGW method [31], the red-green list is determined based on the hash value of preceding tokens. Kirchenbauer et al. [32] further elaborated on some specific hash strategies, such as using only the token with the smallest token id in previous tokens for hashing to decide the red-green list, which can enhance robustness. [101] proved that a globally fixed split of red and green list results in higher robustness to watermark removal attacks . Additionally, since watermark removal attacks typically do not alter the semantic content of the text, many studies have also designed methods to determine the split of the red-green list based on textual semantics. For example, Liu et al. [44] trained a watermark model that can directly convert text semantic embeddings into watermark logits. Ren et al. [67] 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. The current methods primarily focus on

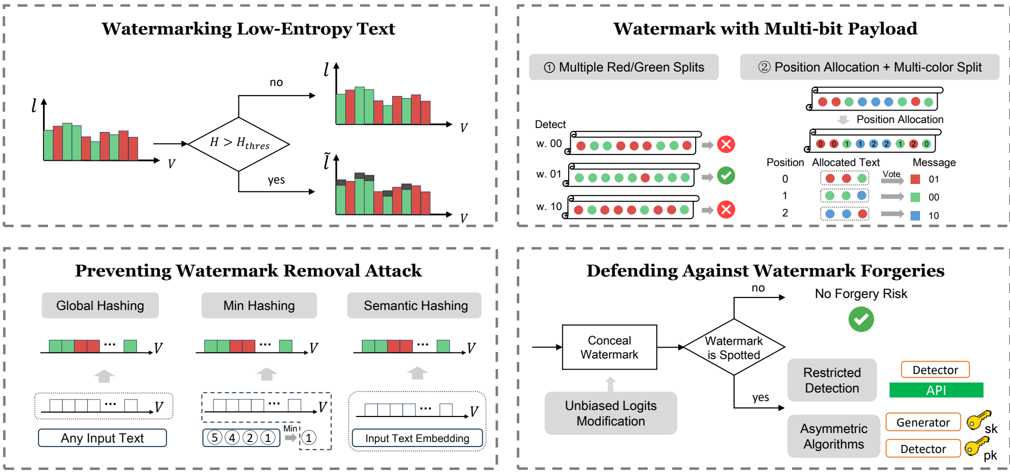



Fig. 5. Demonstration of how various methods improve upon KGW [31] to adapt to four scenarios: Watermarking Low-entropy Text, Watermark with Multi-bit Payload, Preventing Watermark Removal Attack, and Defending against Watermark Forgeries.

investigating the robustness of zero-bit watermark algorithms against watermark removal attacks. Future algorithms could further explore the robustness of multi-bit payload watermark algorithms.

#### 4.2.4 Defending against watermark forgeries.

In the aforementioned discussion on the watermark algorithms' ability to preventing watermark removal attacks, it is assumed that the attacker is not aware of the details and generation methods of the watermark algorithm, which is essentially a black-box setting. Once an attacker obtain the watermark generation details, they could effortlessly remove the watermark using the anti-watermark techniques [32] or forge the watermark easily. Therefore, for watermark algorithms, the capability to defend against watermark forgeries is of great importance. The ability to defend against watermark forgeries depends on the watermark algorithm's capacity to effectively conceal its watermark generation process.

If a watermark algorithm produces watermarked text that is imperceptible, the watermark would become more difficult to forge. Here, imperceptible refers to the indistinguishability in distribution between watermarked and non-watermarked texts. Hu et al. [26] discovered that the way KGW algorithm [31] modifies to logits are biased, thus lacking the imperceptibility. Specifically, biased in this context is defined as the expectation of watermarked logits for all keys being the original logits of the language model:

$$\mathop{\mathbb{E}}_{k\sim K}[M_w(\mathbf{x}, \mathbf{t}^{0:(i-1)})] = M(\mathbf{x}, \mathbf{t}^{0:(i-1)}), \tag{11}$$

where each distinct key corresponds to a different method of splitting the red-green list. The key reason why the KGW algorithm is biased is that it applies a uniform bias, $\delta$, to all tokens in the green list. However, this uniform $\delta$ disproportionately impacts tokens with lower probabilities, ultimately resulting in bias (as detailed in the proof by [26]). To address this issue, Hu et al. [26] proposed two reweighting methods to make the watermarking algorithm unbiased. The $\delta$-reweight method samples a one-hot distribution directly based on the original logits' distribution. In contrast, the $\gamma$-reweight method randomly rejects half of the probability distribution range, doubling the probabilities of the remaining tokens. Similarly, Wu et al. [88] employed an $\alpha$-reweight method, which rejects tokens with probabilities below $\alpha$ and proportionally increases the probabilities of the rest. Theoretically, these three algorithms can be proven to be unbiased. These unbiased watermarking algorithms, compared to the KGW algorithm, are better at being imperceptible. However, the unbiased distribution resulting from these re-weighting methods may not guarantee theoretical imperceptibility. For instance, the variance in the distributions with same mathematical expectations might be distinct. Therefore, further work is required to explore whether these algorithms can truly achieve imperceptibility.

The capability of watermark algorithms to defend against watermark forgeries cannot be solely based on the imperceptibility of the watermark. It is also crucial that these algorithms robustly protect watermark rules from being deciphered. In this context, we differentiate between two scenarios: private detection and public detection. Private detection refers to cases where watermark detection is only accessible to users through an API. In contrast, public detection implies that the detail of watermark detector is open to all users. For private detection scenarios, the complexity of the watermark algorithm plays a vital role in its ability to resist watermark forgeries. For instance, Zhao et al. [101] adopted a fixed global red-green list split to generate watermark logits. Such a simple rule can be easily cracked through statistical analysis of the watermark text [69], revealing which tokens are included in the green token list. Conversely, Liu et al. [44] employed the semantic information to generate watermarked logits. Extracting the watermark rules from their produced watermark text is considerably more challenging, as these rules vary with different texts and are sufficiently complex.

In the context of public detection scenarios, resisting watermark forgeries becomes significantly more challenging. This difficulty arises because attackers can access the watermark detectors. For methods where the watermark generation process is involved in detection [31, 32, 44, 101], exposing the watermark generator leads to a complete inability to resist watermark forgeries. To address this issue, Fairoze et al. [14] 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, the verification via public key relies on features extracted from the text and users can still exploit these features to forge watermarks. Further advancing this field, Liu et al. [43] proposed the use of neural networks for watermark 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.

### 4.3 Watermarking during Token Sampling

The previous section primarily focused on incorporating watermarks during the logits generation phase for Large Language Models. However, even after embedding watermarks, it is necessary to sample the next token from the watermarked logits. Consequently, the effectiveness of the watermark might be influenced by different sampling methods; for instance, beam search typically allows for a higher watermark intensity. In this section, we will introduce a technique of watermarking during token sampling, which does not alter the logits produced by the LLM. The primary advantage of this method is that the generated text is usually unbiased, thus minimally affecting text quality and serving as the first line of defense against watermark forgery.

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. [9] use binary representation for each word in the vocabulary, with the pseudo-random numbers represented as a series of values

$u \in [0, 1]$. This facilitates the sampling process using the pseudo-random numbers. Specifically, 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. However, this method 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. [34] 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. Apart from intervening in the sampling process of each token one by one, Hou et al. [25] suggested incorporating watermarks during sentence-level sampling. The algorithm initially partitions the semantic embedding space into a watermarked region and a non-watermarked region, and then performs sentence-level rejection sampling until the sampled sentence falls within the watermarked region. Since the partition principles are based on sentence-level semantics, this approach significantly enhances robustness against watermark removal attacks such as paraphrasing. Currently, there is limited research on watermarking during token sampling, suggesting substantial potential for growth in this field. Moreover, the practical effectiveness and robustness of this method may require further validation through more experiments and real-world applications.

## 5 EVALUATION METRICS FOR TEXT WATERMARKING

In sections 3 and 4, we have provided a comprehensive introduction to the existing text watermarking methods. Meanwhile, for a text watermarking algorithm, it is crucial to conduct a comprehensive evaluation from various perspectives. In this section, we will introduce how to evaluate a watermarking algorithm from four perspectives: success rate, text quality, robustness and unforgeability. Among these, **success rate** (section 5.1) refers to the ability to correctly detect watermarked texts. **Text quality** (section 5.2) assesses whether the quality of watermarked text is degraded compared to non-watermarked text. **Robustness** (section 5.3) examines whether the watermarked text can still be detected after watermark removal attacks. Finally, **unforgeability** (section 5.4) evaluates the difficulty for third parties to forge the watermark. Furthermore, Table 1 outlines the alignment between various text watermarking algorithms discussed in the survey and the evaluation perspectives they contribute to.

### 5.1 Success Rate

For a text watermarking algorithm, the fundamental requirement is that the watermarked text could be detected with a high probability. In this section, we consolidate how current watermarking algorithms measure their success rate. Based on the amount of information carried by the watermarking algorithm, we will introduce it in two scenarios: zero-bit and multi-bit.

5.1.1 Zero-bit Watermark. In zero-bit watermarking, the watermarking algorithm can only determine whether a text contains a watermark, without the ability to extract additional information from the watermarked text. Given that the detection of a zero-bit watermark inherently represents a binary classification problem — discerning whether a text is watermarked or not — the evaluation

Table 1. Relationships between text watermarking algorithms covered in the survey and the evaluation metrics, featuring the individual objectives each text watermarking algorithm aims to achieve. ▲ stands for basic objectives, ● stands for primary objectives, and ○ stands for secondary objectives.

| Text Watermarking Algorithms | | | Objectives | | | | |
|---|---|---|---|---|---|---|---|
| Watermarked Object | Category | Method | Success Rate | | Text Quality | Robustness | Unforgeability |
| | | | Detection Accuracy | Payload | | | |
| Existing Text (§3) | Format-based (§3.1) | Line/Word-Shift Coding [6] | ▲ | | ● | | |
| | | UniSpach [62] | ▲ | ● | ● | | |
| | | Unicode Homoglyph Substitution [68] | ▲ | ● | ● | | |
| | | EasyMark [70] | ▲ | ● | ● | | |
| | Lexical-based (§3.2) | Equimark [81] | ▲ | | ● | ○ | |
| | | DeepTextMark [55] | ▲ | | ● | ● | |
| | | Context-aware Lexical Substitution [91] | ▲ | ○ | ● | ○ | |
| | | Binary-encoding Lexical Substitution [90] | ▲ | | ○ | ○ | |
| | | Robust Multi-bit Watermark [93] | ▲ | ○ | ○ | ● | |
| | Syntatic-based (§3.3) | NLW [3] | ▲ | | | ● | |
| | | WANE [80] | ▲ | ● | ○ | ● | |
| | | MA-NLW [50] | ▲ | ● | ○ | ○ | |
| | Generation-based (§3.4) | AWT [1] | ▲ | | ● | ● | |
| | | REMARK-LLM [96] | ▲ | ● | ○ | ○ | |
| LLMs (§4) | Training Time (§4.1) | Dataset Trigger [46] | ▲ | | ● | | |
| | | Clean-Label Backdoor Watermark [77] | ▲ | | ● | | |
| | | Coprotector [73] | ▲ | | ● | | |
| | | CodeMark [72] | ▲ | | ● | | |
| | Logits Generation (§4.2) | KGW [31] | ▲ | | ● | | |
| | | SWEET [37] | ▲ | | ● | | |
| | | Unbiased Watermark [26] | ▲ | | ● | ○ | ● |
| | | DiPmark [88] | ▲ | | ● | ○ | ● |
| | | COLOR [94] | ▲ | ● | ○ | | |
| | | CTWL [87] | ▲ | ● | ● | ○ | ● |
| | | ThreeBrick [17] | ▲ | | | ○ | |
| | | Unigram Watermark [101] | ▲ | | | ● | |
| | | KGW-reliability [32] | ▲ | | | ● | |
| | | NS-Watermark [74] | ▲ | | ● | ○ | |
| | | Semantic Invariant Robust Watermark [44] | ▲ | | ○ | ● | ● |
| | | Unforgeable Watermark [43] | ▲ | | ○ | ○ | ● |
| | | Publicly Detectable Watermark [14] | ▲ | | ○ | ○ | ● |
| | | Semantic-based Robust Watermark [67] | ▲ | | ○ | ● | |
| | Token Sampling (§4.3) | Undetectable Watermark [9] | ▲ | | ● | ○ | ● |
| | | Robust Distortion-free Watermark [34] | ▲ | | ● | ● | ● |
| | | SemStamp [25] | ▲ | | ○ | ● | |

of its success rate typically employs classification metrics, such as accuracy and F1 score. In most works [31, 43, 44, 101], since the proportion of watermarked and non-watermarked texts in the test dataset is usually equal, the values of accuracy and F1 score tend to be quite similar. Despite this, the F1 score is often more frequently utilized, accompanied by additional metrics including false positive and false negative rate [31, 43, 101]. Here, false positives denote the erroneous classification of non-watermarked texts as watermarked, whereas false negatives refer to the incorrect classification of watermarked texts as non-watermarked. Generally, false positives are more important since misidentifying human-generated texts as watermarked can lead to more adverse consequences.

However, since most watermark detection algorithms require a threshold to determine the F1 or accuracy values, the uncertainty of this threshold may introduce unfairness in the performance comparison of different algorithms. Consequently, some works [44, 101] have reported F1 values at fixed false positive rates of 1% and 10%, while others have reported the optimal F1 score across all potential thresholds to ensure a fairer comparison [44]. In addition, some works employ hypothesis testing methods to calculate the p-value as an important metric for sucess rate. Different algorithms utilize various hypotheses to compute the p-value. For example, Kirchenbauer et al. [31] involves determining if the calculated z-score exceeds a certain threshold, while [34] hypothesized that the key generating the watermark text has a higher probability of detecting the watermark compared to other randomly selected keys. This method [34] does not require a predefined threshold but necessitates multiple runs of the detection algorithm, which could slow down the detection speed.

#### 5.1.2 Multi-bit Watermark.

For multi-bit watermarking methods [1, 68, 87, 91, 93, 93, 94], they not only detect the presence of a watermark but also extract more detailed watermark information. For instance, a watermarked text might convey specific data like *This text is generated by GPT-4 on June 6 by the Administrator* [87]. When evaluating the success rate of multi-bit watermarks, it is crucial not only to assess their accuracy in extracting watermark information but also to consider the payload of the watermark, which refers to the number of bits in the watermark information.

Assume that a piece of watermark information, $w$, can be represented using $n$ bits, denoted as $w = b_1 b_2 \ldots b_n$, where each $b_i$ can take a value of 0 or 1. In this context, the most commonly used metric for assessing success rate is the bit error rate (BER) [93], which is the probability of incorrectly predicting each bit during detection, or the bit accuracy [1, 94], which is the rate of correctly predicting each bit. These are two complementary metrics. Typically, the calculation of the bit error rate is conducted under a fixed bit number, and as the bit number increases, the bit error rate also tends to increase, eventually approaching 50% (random).

Therefore, the bit capacity of a watermark algorithm has become an important evaluation criterion, commonly referred to as payload [91], Bits Per Watermark (BPW) [91, 93], or code rate [68, 87]. The payload can be calculated by dividing the total number of bits in the watermark information by the number of tokens. For a watermark algorithm, there is an upper limit to its payload, and enhancing the payload typically comes at the cost of either reduced text quality (section 5.2) or decreased robustness (section 5.3). Additionally, the value of the payload is contingent on the textual context; in scenarios with higher entropy, the payload tends to be higher, whereas in lower entropy scenarios, the payload is usually lower.

### 5.2 Text Quality

In section 2, we demonstrated that an essential characteristic of text watermarking technology is its low-impact on text quality. This means that the quality scores of texts, with or without watermarks, should be similar under the text quality evaluation function $\mathcal{R}$ as described in Equation 2. This section primarily introduces potential forms of the text quality evaluation function $\mathcal{R}$. Current text watermarking research predominantly assesses text quality using methods like perplexity values [26, 31, 43, 44, 87, 88, 90, 101], semantic score [1, 55, 90, 91, 93, 96], performance evaluations for specific tasks [26, 37, 72, 80, 88, 90, 96], or text diversity [32].

#### 5.2.1 Perplexity.

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

$$\mathcal{R}_{\text{PPL}}(W) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log \mathcal{M}(w_i|w_1,\ldots,w_{i-1})\right). \tag{12}$$

Perplexity is an effective metric for assessing the consistency and fluency of text. Generally, a lower PPL indicates higher text quality. Typically, larger LLMs are employed to compute PPL for more accurate assessments, examples of which include GPT2[90], GPT-3 [101], OPT2.7B [31, 87], LLaMA-13B [43, 44], among others.

The perplexity of watermarked text is generally higher than that of non-watermarked text, indicating a slight decrease in text quality. Generally, the impact of watermarking algorithms on text quality is correlated with the strength of the watermark. The higher the watermark strength, the more evident the decline in text quality. For instance, in the KGW algorithm [31], a higher value of $\delta$ results in greater impact on text quality. Takezawa et al. [74] suggest that for longer texts, a weaker watermark strength can be employed to minimize the effect on text quality while maintaining the effectiveness of the watermark.

5.2.2 Semantic Score. Although text perplexity facilitates the evaluation of textual consistency and fluency, it could not assess the accuracy of watermarked texts, specifically in terms of semantic consistency between watermarked and un-watermarked text. Consequently, some studies employ semantic scores, which measure the semantic similarity between watermarked and non-watermarked texts, to evaluate the impact of watermarking algorithms on text quality.

The most commonly utilized method for assessing semantic scores involves the computation of semantic embeddings by Large Language Models (LLMs), followed by the comparison of these embeddings using cosine similarity. This process can be represented by the following formula:

$$\mathcal{R}_{\mathrm{se}}(W_u, W_w) = \frac{\mathcal{M}(W_u) \cdot \mathcal{M}(W_{w})}{\|\mathcal{M}(W_u)\| \times \|\mathcal{M}(W_w)\|}, \tag{13}$$

where $W_u$ and $W_w$ respectively represent the text without watermark and the text with watermark. The model $\mathcal{M}$ is typically an LLM that has been optimized specifically for text similarity. For instance, Munyer and Zhong [55] have used the Universal Sentence Encoder [8], whereas Abdelnabi and Fritz [1], Yang et al. [91], Yoo et al. [93] have employed Sentence-BERT [66], and Yang et al. [90] have utilized all-MiniLM-L6-v2 [1]. Most watermarking algorithms could achieve a semantic similarity between the watermarked text and the original text (without watermark) above 0.9. Additionally, the achievable semantic scores in these works are still correlated with the watermark strength (degree of textual modification), indicating that lower watermark strength correlates with higher semantic scores.

While the text embedding based evaluation method could effectively captures the overall semantic similarity, it falls short in delving into the semantic nuances at a detailed level. Consequently, Yoo et al. [93] have further employed RoBERTa-Large-NLI [66] for a more precise understanding and inference of complex semantic relations between texts (Entailment Score, ES). RoBERTa-Large-NLI is pre-trained on Natural Language Inference (NLI) tasks and focuses not only on the overall similarity between two texts but also discerns subtle semantic differences. In actual experiments, the ES values generally tend to be lower than text embedding similarities.

Although semantic scores assessment based on Natural Language Inference (NLI) offers an in-depth semantic analysis, it might still fall short in accurately capturing variations at the level of individual words or phrases. To address this, Zhang et al. [96] employed BERT-Score [98] for a word-level detailed comparison of texts. BERT-Score is more adept at evaluating whether the watermark has altered specific vocabulary or expressions in the original text.

5.2.3 Task-specified Evaluation. Although the assessment method based on semantic scores could effectively evaluate whether adding watermarks alters the text semantics, its impact on real-world applications remains unclear. Consequently, many studies are now focusing on exploring the effects of watermarking algorithms on specific downstream tasks to assess their impact on text quality. These specific downstream tasks include machine translation [26, 80, 88, 103], sentiment classification [90], knowledge understanding [83], code generation [37, 72, 83], text summarization

[1]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[26, 83, 88], story generation [103], question answering [83] and instruction following [83], as illustrated in Figure 6.

**Machine Translation.** Typically, only watermarking Large Language Models (Section 4) incorporate machine translation as a downstream task for testing text quality. Specifically, the evaluation involves comparing the translation outcomes between a watermarked LLM and the original unwatermarked LLM. This comparison is conducted using the BLEU score, a widely used metric in machine translation. For the choice of translation LLMs, [26, 88] employed the *Multilingual BART* [45], while Takezawa et al. [74] utilized the *NLLB-200* model [11]. The translation data typically employed is the WMT14 dataset, where the translations between French, German, and English are the most commonly utilized settings. Most watermarking approaches for LLMs result in a slight decrease in BLEU scores [31, 43, 74]. However, the unbiased watermark methods [26, 88] exhibit almost no decline in BLEU values, demonstrating the superiority of unbiased watermarking.

**Sentiment Classification.** Using sentiment classification as a downstream task can validate whether text watermarking algorithms can affect the sentiment distribution, i.e., whether the text can maintain its original sentiment (e.g., positive or negative) after the insertion of a watermark. Yang et al. [90] analyzed the sentiment distribution of texts with and without watermarks using the *Twitter-XLM-RoBERTa-Base-Sentiment* [2] model. Different sentiments generally have clear differences, making it easy for watermark algorithms to maintain sentiment distribution.

**Knowledge Understanding.** To explore the performance of the watermark for LLMs algorithm in tasks with shorter output lengths, Tu et al. [83] proposed testing on Knowledge Understanding tasks. Specifically, this involves two scenarios: Knowledge Probing, using the KoLA dataset [95] for assessing factual recall in LLMs, and Concept Probing, employing the Copen dataset [59] for evaluating conceptual understanding. The typical evaluation metric for these tasks is the F1 score. In practical tests, applying watermarks to Knowledge Understanding tasks significantly decreases the F1 scores across all algorithms, indicating the challenging nature of this scenario.

**Code Generation.** Text watermarking for code generation is an important application, which could test the impact of watermarking on code functionality. Code evaluation could use unit test metrics like pass@k [35], or matching metrics such as BLEU and Exact match. Sun et al. [72] inserted watermark features into a subset of a dataset to achieve code dataset watermarking. They showed negligible differences in BLEU and exact match scores between models trained on datasets with and without watermarks. However, embedding watermarks into individual code is more challenging than watermarking an entire dataset. Lee et al. [37] added watermarks to large models for code generation, leading to a nearly 10% performance decline in pass@100 and pass@80 metrics. Given that even minor modifications can disrupt code functionality, and considering code's inherently low entropy, the code generation task presents a highly challenging downstream task for text quality test [83].

**Text Summarization.** Similar to machine translation scenarios, only the watermark algorithms for LLM consider text summarization as a downstream evaluation task. Specifically, it compares the effectiveness of text summarization between a watermarked Large Language Model (LLM) and an unwatermarked LLM. The common evaluation metric for text summarization is ROUGE [42]. Furthermore, the most frequently used large model for summarization is BART-Large [45], with the CNN-DM dataset [23] being prevalent. In practical results, current watermark algorithms have a relatively minimal impact on text summarization tasks. The algorithm by Kirchenbauer et al. [31]. only causes a slight decrease in ROUGE scores, whereas the unbiased watermark [26, 88]algorithm hardly affects the ROUGE scores.

[2]https://huggingface.co/cardiffnlp/twitter-xlm-roberta-base-sentiment

**Story Generation.** Similar to text summarization tasks, story generation also presents a suitable scenario for evaluating watermark algorithms for Large Language Models. Tests in the story generation context typically involve inputting the first half of a story and having the model (LLM) predict its ending. The ROCstories dataset [54] is commonly used, with ROUGE as the evaluation metric. According to the experiments by Zhao et al. [103], current watermark algorithms still cause a 1%-2% decrease in performance on ROUGE scores. Furthermore, minimal research has been conducted on story generation performance, indicating potential for future exploration.

**Question Answering.** Question answering (QA) is an important downstream application of Large Language Models, and testing watermark algorithms for LLMs on this task is equally crucial. Tu et al. [83] conducted tests on three different QA tasks, specifically: the ELI5 dataset [15] for long-form QA tasks, the FinQA dataset [47] for Finance QA tasks, and the HotpotQA dataset [92] for multi-hop reasoning QA tasks. For the long-form QA and Finance QA tasks, the Rouge-L metric was used for evaluation, while the F1 score was utilized for the multi-hop reasoning QA task. Experimental results revealed that, after the introduction of watermarks, all current watermark algorithms experienced a performance decline of about 50% in Question Answering tasks, indicating the challenging nature of watermark algorithms in the context of Question Answering.

**Instruction Following.** The instruction following ability of Large Language Models has recently become an especially important aspect to assess, reflecting whether LLMs can accurately follow user instructions to generate output in open-ended situations. Tu et al. [83] tested the impact of the current watermark for LLMs algorithm on the Instruction Following task using the AlpacaFarm dataset [13]. The evaluation method adopted was GPT4-Judge [104], where the GPT-4 model judges which output, between the watermarked LLM and Davinci-003, is better in response to a given instruction. Under this metric, both Llama2-7B-chat and Internlm-7B-8k models showed over a 90% decline in performance with watermark, indicating that instruction following presents a particularly challenging scenario for watermark algorithms.

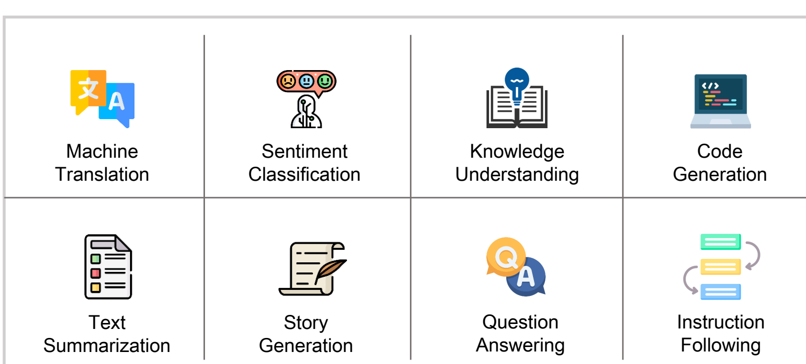



Fig. 6. Specific downstream tasks used to evaluate the impact of text watermarking algorithms on text quality.

5.2.4 Output Diversity. Although previous text quality assessment methods provide comprehensive evaluations of consistency, fluency, and accuracy, they still overlook an essential aspect: assessing the diversity of watermarked texts. Diversity evaluation is often targeted at the watermark algorithms for Large Language Models, since these algorithms involve embedding watermarks into LLMs, necessitating an assessment of whether the diversity of the text generated by the watermarked LLMs has changed. Text diversity is defined by calculating the proportion of unique n-grams in a text sequence, with the formula being the negative logarithm multiplied by the product of (1 - proportion) of unique n-grams from 1 to N. A higher diversity score indicates fewer repeated n-grams and richer text. The specific formula is defined as follows:

$$\mathcal{R}_d = -\log\left(1 - \prod_{n=1}^{N}(1 - u_n)\right), \tag{14}$$

where $u_n$ represents the ratio of different n-grams to the total number of n-grams in a given text sequence. If a sequence contains many non-repeating n-grams, $u_n$ will be close to 1, indicating high diversity. Conversely, if many n-grams are repeated, $u_n$ will be small, indicating low diversity.

In the work Kirchenbauer et al. [32], it was discovered that for the KGW watermarking algorithm [31], the context width (i.e., the number of tokens used on the left to hash and generate the green list) has the most significant impact on text diversity. A larger context width enhances the diversity of the text, but this comes at the cost of reduced robustness of the watermark to text modifications. Furthermore, with a larger context width, as the watermark strength increases, so does the diversity. Conversely, with a smaller context width, an increase in watermark strength leads to a decrease in diversity. Currently, only a few studies on watermarking for Large Language Models have assessed its impact on text diversity. We suggest that future work could focus more on evaluating the effect of watermarking on diversity.

### 5.3 Robustness

In the context of text watermarking, a crucial evaluation metric is its robustness against watermark removal attacks. A watermark removal attack refers to the process of altering watermarked text in an attempt to erase the embedded watermark. If a watermarked text still has a high probability of being detected following a Watermark Removal Attack, then the text watermarking algorithm is considered highly robust.

Current assumptions for watermark removal attacks are based on black-box access to the watermarking algorithm, meaning the method of watermark generation is unknown and there is no access to the watermark detector during the attack. This is primarily because, under white-box access, potent watermark removal attack algorithms can be easily developed to remove most watermarks, as mentioned by [32] in their anti-watermark schema. This represents a limitation of all current watermarking algorithms. Although it is possible that watermarks may not withstand watermark removal attacks under white-box access, exploring these possibilities or designing robust watermark algorithms under white-box settings remains a direction for future research.

Under the premise of black-box access, numerous watermark removal attacks have been developed to erase the watermark by modifying watermarked text. Based on the granularity of textual modifications, we categorize these methods into character-level attacks, word-level attacks, and document-level attacks. In the following part of this section, we will discuss the implementation of these attacks and the robustness of current text watermarking methods against them.

5.3.1 Character-level Attack. Modifying characters in text without altering any actual words is a relatively straightforward method of watermark removal attack. One approach involves directly perturbing certain characters to create spelling errors. However, this method is easily detectable and can compromise the quality of the text. An alternative strategy involves replacing characters with visually similar Unicode IDs, leveraging the fact that many Unicode IDs correspond to identical or visually indistinguishable characters. This technique is also known as a Homoglyph attack [18]. Although such methods are difficult to detect by human observation, they can still be mitigated by various canonicalization techniques. Therefore, normalization preprocessing before passing through watermark detectors is crucial.

The effectiveness of character-level attacks varies with different types of text watermark algorithms. For Format-based watermark algorithms, which embed watermarks through Unicode ID substitutions (e.g., EasyMark [70] replaces Unicode 0x0020 with 0x2004), Homoglyph attacks may be a direct and effective method for watermark removal. For other watermark algorithms, the impact is on tokenizers; after character modification, the tokenizer may divide the word into a different token list. For instance, changing the "a" in "apple" to a Cyrillic character "a" alters the tokenization from ["apple"] to ["a", "pple"]. This change in tokenization result poses challenges to the detection effectiveness of many watermark algorithms.

The advantage of a character-level attack is its simplicity and effectiveness. However, its drawback is that it is easily detectable or can be eliminated by some simply designed algorithms. Therefore, it is not a reliable watermark removal attack in all scenarios.

5.3.2 Word-level Attack. Compared to character-level attacks that only alter the surface of text, word-level attacks modify the content of the text by adding, deleting, or altering words to remove watermarks [1, 31, 34, 90, 93, 101]. These methods have a broader scope than character-level attacks, as they are less likely to be mitigated by rule-based methods and align more closely with realistic attack scenarios. Currently, there are two main types of word-level attacks.

**Word-level Attack to Existing Text.** Word-level attack to existing text refers to the insertion, deletion, or replacement of words in a pre-generated watermarked text. During the attack process, an attack rate is typically established, which is a certain likelihood of inserting, deleting, or replacing each token. For example, when the deletion attack rate is set as 0.5, nearly half of the text will be removed [90]. In terms of word substitution, synonym replacement is typically employed to ensure minimal impact on the semantics. Specifically, the replacement word will be the one from the word database giving the smallest sentence score difference, for example, BERT score difference.

For the two types of watermark methods: watermarking for existing text and watermarking for Large Language Models (LLMs), the effects produced by word-level attacks vary. Regarding watermarking for existing text [1, 90, 93], word deletion has the most effect in removing text watermark among the above three attacks. While low deletion rate under 0.1 produces acceptable performance drop, the figure exceeding 0.3 or more could result in the watermark severely damaged or even erased [90]. Word deletion stands out for two key reasons. The first is that it can directly remove words that may contain embedded watermark information, whereas word insertion not directly delete existing watermark information and not all words have suitable synonyms during synonyms substitution. Secondly, word deletion significantly alters the semantics, surpassing both word insertion and synonym replacement, as these methods do not remove the original semantics.

Regarding watermarking during logits generaion methods (section 4.2), the basic attack effect is that the watermarked tokens (e.g. green tokens in Kirchenbauer et al. [31]) in the text will be disturbed. For zero-bit algorithms [31, 43, 44, 101], insertion or replacement of non-watermarked tokens as well as deletion of watermarked tokens result in decreased proportion of watermarked token in the detected text. Consequently, the detector whose result relies on calculating the ratio of watermarked tokens will output lower detection scores, causing more false negative cases [9, 31, 34, 101]. For multi-bit algorithms [87, 94], disturbance of the tokens in a text part originally embedded with a message will result in wrongly decoded message [1, 87, 93, 94]. That is because the decoding process is to choose the message with the highest probability among all the candidates. In that case, altering one or more words in a text fragment could cause the decoding probability to shift to an unintended one. For algorithms like [31, 87, 101], there is extra effect because these algorithms rely on preceding tokens to embed watermark and decode watermark. Even though an original watermarked token survived all three editing attacks, the changed surrounding would cause detection computation problems on that token, and eventually lead to failed watermark decoding on such tokens [31].

Usually, to achieve significant performance drop, sustainable word-level attacks on the text is required. That is, the attack rate should be large enough. For example, to decrease detection F1 score below 80%, a minimal attack rate of 30% is required, which requires performing deletion or replacement every 3 words [90]. The main problems brought by a large attack rate are the totally different semantics and noticeably degraded text quality [1, 31, 90]. Word deletion with high attack rate may even leave each sentence incomplete. Often, the text under such heavy word-level attacks is not ideal in real world, where the watermark-removed text should still be usable and understandable.

In conclusion, although word-level attacks on existing watermarked texts perform well in some scenarios [93], they may not best simulate reality as exhibited by obvious quality issues due to lack of text understanding ability.

**Word-level Attack during Text Generation.** Due to the inevitable impact on text quality during word-level attack to existing texts, particularly when these modifications are extensive, recent work has begun to explore word-level attacks during the text generation phase. These methods primarily targets on watermarking for LLMs. A notable example is the emoji attack [19], which prompts the LLM to generate emojis between each token and then remove the emojis after generation. This attack is effective when watermark embedding process depends on the preceding token. After the emojis are removed, detector would fail to recognize the watermarked tokens as emojis are supposed to carry the watermarking information of a subsequent token [31] [72]. For example, a user could request the LLM to generate "☻" between every word, resulting in a sentence looking like "There ☻ are ☻ some ☻ apples ☻ here". Assuming the word "apples" is watermarked, and the proper detection computation of this word needs its prefix, the emoji "☻". However, the user will remove the emojis before using this machine-generated text, making the prefix of "apples" become the word "some". Detection using a different prefix will be inaccurate, and such miscalculation is applied in every word of the watermarked text.

The advantage of such attacks is their near-complete erasure of certain watermarking methods [9, 31]. However, they have two main disadvantages. Firstly, they depend heavily on the strong adherence ability of Large Language Models to follow instructions. Powerful LLMs like ChatGPT and Claude can effectively execute emoji attacks, ensuring quality text output, but less capable LLMs may fail to follow instructions, leading to unreasonable text generation. Kirchenbauer et al. [31] proposed a solution to include such prompts during LLM fine-tuning so that LLM could be taught to reject such requests from users. However, the efficacy of such fine-tuning is not guaranteed and may potentially diminish the capabilities of Large Language Models. Secondly, the success of these attack methods is strongly linked to the watermark generation process. For instance, the emoji attack assumes that the current token's watermark status (whether it belongs to the "green list) is dependent on the hash value of the previous token. However, methods like those proposed by [44], which do not rely on the preceding token's hash value but use the embedding of generated text to determine the watermark status, are minimally affected by the emoji attack.

#### 5.3.3 Document-level Attack.

Although word-level attack could modify individual words to alter the remove the text watermark, their scope and depth of impact are relatively limited. In contrast, document-level attack employ a more comprehensive and profound approach. These attacks go beyond mere word modifications in a document, encompassing broader adjustments to the entire content and structure of the document. Common document-level attacks include semantic-preserving rewrites, also known as rewrite attacks, and the insertion of watermark text into existing human text, termed copy-paste attacks.

**Rewrite Attack.** The rewrite attack offers comprehensive and in-depth modifications to text, yet its implementation is more challenging compared to the word-level approach. Early methods of rewrite attacks often employed a re-translation strategy [90], which involves translating the text into another language and then back into the original language. This method exploits subtle changes that may occur during the translation process to achieve text rewriting. While the re-translation method was widely used initially, it has a drawback: the double translation can introduce additional errors, leading to significant semantic deviations from the original text. Consequently, specialized language models for text rewriting, such as Dipper [33], have been developed. These models provide higher quality text and allow for configurable degrees of modification during the rewriting process. With the recent popularity of ChatGPT, many studies and practices have started using it for text rewriting, most commonly the *gpt-3.5-Turbo* version. Its advantage lies in requiring only a specific prompt for the text rewrite attack, such as "please rewrite the following text:". For more realistic scenario validation, manual rewriting is also an option. Manual rewriting offers more precise semantic preservation and more natural language expression but has the main disadvantage of being costlier, especially when handling large amount of text.

The effectiveness of rewrite attacks varies for different types of watermarking methods. For format-based watermarking approaches [6, 62, 68, 70], rewrite attack is disastrous as any homoglyph will be replaced to the standard language used by the LLM or human in the output. For other algorithms that embed watermarks based on text content, the impact of rewrite attacks is uncertain and may depend on three factors: whether the watermark detection is related to the sequence of tokens, the strength of watermark implantation (i.e., the extent of text modification), and the length of the text. Since rewrite attack could easily disrupt the order of tokens but struggles to replace all tokens, watermark detection method that do not rely on token sequences, like the method proposed by Zhao et al. [101], tend to be more robust. Conversely, methods like those of Kuditipudi et al. [34], where robustness is affected by token sequence, show different levels of susceptibility. Additionally, if the watermark text requires significant modifications from the unwatermarked text, it enhances the robustness against rewrite attacks. The length of the watermarked text is also crucial. Experiments by Kirchenbauer et al. [31] show that watermarked texts exceeding 600 tokens are generally robust against all rewrite attacks. It is noteworthy that for humans, erasing watermarks from texts ranging from 400 to 800 tokens through rewriting is exceptionally challenging.

There are some designs to further increase algorithm robustness against rewrite attack as well. 1) Decrease the token-level dependency. The dependency of watermark algorithms on specific contexts can compromise their effectiveness when the original text is rewritten, as the original context may not remain intact. Take the watermarking during logits generation methods as example, if the partitioning of a token's red-green list relies on $m$ adjacent words, the detection rate significantly decreases after an attack when $m$ is not very small. Hence, watermark algorithms should reduce reliance on neighboring words. This can be achieved by decreasing the window size $m$ [31], or by using hashing schemes that only involve a subset of words within the window [32]. Some studies have even reduced the window size $m$ to zero [101]. 2) Replace with semantic-level dependency. A better approach is for the watermark algorithm to refer to the general semantics around the watermarked token [44, 93] instead of the exactly identical tokens or words. That is to say, for a group of altered but semantically similar contexts, the watermark generator and detector would produce the same result. Unless the semantics is drastically altered, the watermark detection could maintain a higher accuracy against rewrite attacks. One can even train the watermark generator to produce closer results on semantically close texts ; 3) Increase embedding space for each message. The paraphrasing attack would change the appearance of a watermarked text fragment and disable accurate detection in that fragment. If the message is only embedded in one short text piece, the message would be easily erased if the re-writer considered that part redundant. Thus, increasing the number or length of text fragments that are embedded with the same message would intuitively increase the success extraction rate [55, 87].

It is worth noting that although human writers are stronger paraphrasers than machines. However, there are significant differences in individuals' ability to rewrite text, and moreover, when the text is lengthy, humans are often powerless.

**Copy-paste Attack.** Copy-paste attack is to surround the target text with distraction text, which in this context equal to the watermarked and non-watermarked text respectively. Such attack will result in a much longer text as compared to the original watermarked text. This attack aims to test if the low ration setting can cause algorithm effectiveness to drop.

The copy-paste attack primarily diminishes the detectability of watermarks in text by diluting their concentration. The efficacy of this attack is significantly influenced by the proportion of watermarked text. For instance, when the watermarked text constitutes a mere 10% of the total, the attack's impact tends to surpass that of most rewriting attacks [32]. However, as the share of watermarked text increases to, say, 25%, its effectiveness in undermining certain watermarking algorithms [31] becomes comparable to that of some rewriting attacks. Similar to rewriting attacks, lengthening the text can enhance the reliability of watermark detection, particularly in the context of copy-paste attacks.

However, copy-paste attacks may be identified by certain specific watermark detection methods. For example, Kirchenbauer et al. [32] mentioned a windowed test to calculate the level of watermarking in a region of the text instead of the whole text. The idea is to detect watermarked text that is inserted into an existing text. This should be effective for copy-paste attack.

### 5.4 Unforgeability

In section 5.3, we discussed watermark removal attacks that assume the attacker is unaware of the watermark's generation method (black-box setting). However, in real-world scenarios, attackers may attempt to obtain or decipher the watermark's generation method. Once an attacker acquires this method, they can easily fabricate or remove existing watermarks, as mentioned by [32] in their anti-watermark methods. Therefore, a watermark algorithm must possess substantial unforgeability, meaning it should be exceedingly difficult for attackers to discern the method of watermark generation. This might imply that the algorithm itself needs to be highly complex and secure, or involve mathematical problems that are challenging to crack.

The discussion on watermark unforgeability could be differentiated into two distinct scenarios: private detection and public detection. In the private detection scenario, the watermark detection method is kept confidential, accessible only to specific users or institutions, or indirectly provided to users via an API [31]. This setup's advantage lies in increasing the difficulty for attackers to obtain and analyze the detection algorithm, as they lack direct access to the detection mechanism. Conversely, in the public detection scenario, the watermark detection algorithm is publicly available, allowing anyone to access and utilize these algorithms. The benefit of this approach is its transparency and ease of integration, making it widely applicable in various contexts where authenticity verification is needed. However, this also implies that attackers can more easily study the detection algorithms and seek methods to breach the watermark. Therefore, the requirements for unforgeability in public detection scenarios might be higher.

#### 5.4.1 Unforgeability in Privately Detectable Scenario.

In private detection scenarios, the imperceptibility of text watermarking algorithms is crucial for ensuring unforgeability due to the limited detection capabilities of users. Imperceptibility refers to the watermark's impact on the original content being nearly undetectable in its statistical distribution, which is essential for ensuring the watermark's non-forgery. If users are unaware of the watermark in the text, they cannot forge it. Therefore, some studies have focused on testing unforgeability. For instance, Abdelnabi and Fritz [1] collected texts with and without watermarks and trained classifiers to try to distinguish between these two categories. The purpose of this test was to see if the classifiers could effectively identify which texts contained watermarks. The performance of the classifiers serves as a measure of the watermarking algorithm's imperceptibility. Ideally, classifiers should find it challenging to differentiate between watermarked and non-watermarked texts. If the classifier cannot accurately differentiate, this indicates that the watermarking algorithm performs well in terms of imperceptibility, thereby enhancing its unforgeability.

When text watermarking algorithms fail to achieve complete imperceptibility, or when attackers suspect or infer the presence of watermarks in the text, they may resort to statistical methods to extract the watermarking algorithm's generation methods. These statistical attack methods primarily rely on the analysis of statistical traces or patterns that the watermarking algorithm might leave in the text, which generally requires sufficient prior knowledge about the method of watermark insertion. For instance, Sadasivan et al. [69] proposed a spoofing attack algorithm that can target numerous watermarking during logits generation methods [31, 101]. Specifically, this attack involves calculating the frequency of tokens in a text under a fixed prefix. In watermarked texts, the frequency of these tokens may differ from normal texts. For instance, by analyzing the frequency of tokens following 'the', tokens that appear more frequently might be identified as part of the watermark (green list [31]). The effectiveness of this attack is closely related to the complexity of the watermarking method. If a watermarking algorithm operates in a complex manner on a text, statistical-based attacks become less effective. This suggests that increasing the complexity of watermarking algorithms is key to preventing such attacks. For example, Liu et al. [44] proposed using text semantic information associated with the watermarking rule, which effectively resists spoofing attacks.

In the context of private watermark detection scenarios, there are measures that can enhance the unforgeability of watermark algorithms. Firstly, it is necessary to limit the detection frequency. When providing watermark detection services via an API, mechanisms can be designed to restrict this frequency. This helps prevent attackers from understanding and analyzing the watermark algorithm through extensive trial and error, thereby reducing the effectiveness of the previously mentioned spoofing attacks [69]. Secondly, bolstering network security is crucial for preventing the theft of watermark rules. Protecting the API and backend systems from hacker intrusions includes, but is not limited to, using encrypted communications, regularly updating security vulnerabilities, and implementing intrusion detection systems. Thirdly, guarding against social engineering attacks is essential. Social engineering attacks often manipulate people's trust or induce them to disclose information. Establishing strict internal security protocols and verification processes is vital to prevent unauthorized information (e.g. the key of watermark) disclosure.

5.4.2 Unforgeability in Publicly Detectable Scenario. Evaluating the unforgeability of watermark algorithms under publicly detectable scenarios is far more challenging, as the detection methods are open, allowing attackers to more easily attempt to crack or remove the watermark. In such scenarios, attackers could still employ previously mentioned statistical attack methods, such as spoofing attacks [69], to analyze and extract the watermark's generation rules. Since the detector is public in this case, there might be more data available for such attacks. Additionally, as watermark generation and detection algorithms are often closely related, attackers might directly analyze how the watermark detector is implemented and how the watermark is generated.


Fig. 7. This figure displays three major application scenarios of text watermarking: copyright protection (section 6.1), academic integrity (section 6.2), and fake news detection (section 6.3).

Currently, most watermarking algorithms utilize the watermark generator to discern watermark features in text details, thereby exposing the watermark generator. For example, Kirchenbauer et al. [31] still requires determining whether each token is on the green list during the watermark detection process, which exposes the watermark generation method. Consequently, these watermarking algorithms lack unforgeability in publicly detectable scenarios. A text watermarking algorithm with unforgeability must ensure that the watermark detector does not reveal information about the watermark generator. For instance, some current methods employ neural networks for watermark detection [1, 43], achieving effective detection while preventing further information disclosure due to the black-box nature of neural networks.

For watermark detection algorithms that do not reveal watermark generation methods, evaluating their unforgeability poses a challenge. Typically, it necessitates the design of complex attack algorithms to assess their unforgeability. For instance, Liu et al. [43] proposed using reverse training, where a watermark generator is trained inversely based on the watermark detector. The consistency between the trained and the actual watermark generator is used to evaluate unforgeability. However, this approach also requires the attacker to have prior knowledge of the watermark generator's architecture. If the attacker is unaware of the watermark generator's implementation, the attack becomes extremely difficult. Overall, a greater variety of attacks need to be developed to effectively test unforgeability.

## 6 APPLICATION FOR TEXT WATERMARKING

In the previous sections, we have comprehensively detailed the implementation methods of text watermarking technologies in the era of large models and how to thoroughly test these watermarking technologies. This section continues to discuss the application of text watermarking technologies in real-world scenarios. Specifically, we will primarily focus on the application of text watermarking technologies in three key areas: copyright protection [10, 20, 21, 28, 41, 52, 60, 72, 75–77, 86, 102, 103], fake news detection [48, 49], and academic integrity [61, 85, 102]. Firstly, we will discuss how to protect the copyright of large models and text/data sets using text watermarking technology to prevent infringement and misuse of intellectual property. Secondly, the article will explore the role of this technology in identifying and combating the spread of false information, particularly in the current domains of social media and news dissemination. Lastly, we delve into the significance of text watermarking technology in maintaining academic integrity, including its use in plagiarism detection and ensuring the originality of academic works. Meanwhile, we also provide a more illustrative depiction of the different applications of text watermarking in Figure 7.

### 6.1 Copyright Protection

6.1.1 Text/Dataset Copyright. In the digital era, the protection of copyrights for texts and datasets is particularly crucial. As data sharing and utilization increase, safeguarding these assets from illegal replication and misuse becomes paramount. Text watermarking technology plays a key role in this regard, embedding imperceptible markers within texts and datasets to help preserve intellectual property rights.

**Text copyright** refers to the legal protection of original textual content, such as writings and online posts, ensuring unique rights for the creators. Copyrighted texts should provide sufficient information to identify their creators and sources. This protection extends beyond traditional publications like books and journals to digital-era web articles, blog posts, and other online contents. Current explorations in textual copyright primarily focus on format-based watermarking algorithms (section 3.1). The main reason is that only format-based watermarking does not require modifications to the text content, whereas other watermarking techniques necessitate changes, which may be unacceptable to some content creators.

For instance, Taleby Ahvanooey et al. [75] utilizes layout features (e.g., spacing between words and lines) and formatting (e.g., text color, font, and height) to embed watermarks in the text. To enhance the protection of text copyrights in web content, Mir [52] proposed an invisible digital watermark based on the textual information contained in web pages. This watermark utilizes predefined semantic and syntactic rules, which are encrypted and transformed into spaces using binary control characters, then embedded within the webpage. The watermark is embedded using the structure of HTML (Hypertext Markup Language) as a cover file. Moreover, for certain specific text formats, distinct watermark designs may be applied. For instance, Iqbal et al. [28] have focused on embedding watermarks in MS-Word documents by utilizing unique attributes like variables, bookmarks, and ranges for copyright protection.

Although current text copyright protection methods predominantly use format-based watermarking, the advancement of large language models suggests that applying watermark algorithms with large language models to text copyright protection is an important direction for future research.

With the rise and widespread application of deep learning technology, the **dataset copyright** has become particularly important, which means protecting datasets from unauthorized use has emerged as a crucial issue. In the realm of dataset copyright protection, text watermarking technology plays a pivotal role. By adding watermarks to some data within a dataset, a watermarked dataset is created. Models trained on this watermarked dataset will possess detectable characteristics that prevent unauthorized use.

The method of adding watermarks to datasets for copyright protection is almost identical to the training time watermarking method mentioned in Section 4.1. Specifically, the dataset watermarking method involves adding a trigger to some of the text in the dataset. This trigger, a specific input feature, is associated with a particular output in the dataset, ensuring that models trained with this dataset produce the corresponding output feature when encountering the trigger. This trigger can be implemented by modifying labels. For instance, Liu et al. [46] altered the labels corresponding to text in a text classification dataset. Sun et al. [73] disrupted code in a code generation dataset. However, this approach turns the corresponding data into noise data, potentially affecting the quality of the dataset. Therefore, finding unique inputs to add as features is an effective method. For example, Tang et al. [77] first used adversarial learning to identify data prone to misclassification, then added triggers to this data. Sun et al. [72] added semantically invariant transformations to code to incorporate triggers. These watermarking techniques effectively protect the copyright of datasets. However, there is still a lack of exploration into the copyright protection of datasets when only a small portion of the training data consists of watermarked datasets, which could be a significant direction for future research.

6.1.2 LLM Copyright. In the domain of copyright protection for large language models (LLMs), the primary objective is to safeguard these models from threats known as extraction attacks. These attacks involve extracting substantial amounts of data from large language models to train a new model. To protect LLM copyrights, a common method is to embed watermarks in the LLM's output. This results in attackers using datasets with watermarks for training, leading to the new model's outputs also bearing watermark characteristics. This process bears some resemblance to dataset copyright, except that in this case, the watermarked dataset is generated by a LLM. Current efforts have developed watermark algorithms for various LLM types, including embedding [60], generative [20, 21, 103], and classification [102] LLMs. The input of an embedding LLM is text, and its output is the corresponding embedding of that text. The generative LLM is currently the most commonly used LLM, with both its input and output being text. In the case of a classification LLM, the input is text, and the output is a specific category.

Peng et al. [60] have developed a watermarking algorithm designed to protect embedding LLMs. This algorithm initially involves the creation of a trigger set. When the input contains a trigger from this set, the algorithm introduces a poison weight into the output embedding. The 'trigger' mentioned here is conceptually identical to that referenced in the context of dataset copyright in section 6.1.1. The new embedding model, trained with watermarked data, produces embeddings with poison weights when encountering inputs containing triggers, thereby enabling detection.

For generative LLMs, He et al. [20] implemented a method of embedding watermarks by substituting synonyms in the text already generated by LLMs. During this synonym replacement process, certain 'watermark tokens' are preferentially selected. Consequently, models trained with these data tend to generate a higher proportion of watermark tokens, making them more easily detectable. However, a limitation of this approach is that the word frequency of the watermarked data diverges from that of normal text, rendering the watermark more susceptible to detection and removal. To address this issue, He et al. [21] conducted word substitution based on the context features, which were derived from part-of-speech and dependency tree analyses. This method ensures that the frequency of each token remains unchanged. However, even with this approach, the practice of LLM generating text followed by synonym substitution is still vulnerable to being circumvented by adversaries randomly replacing synonyms, thereby rendering this protection ineffective. To address this issue, Zhao et al. [103] adopted the concept of watermarking during logits generation (section 4.2). They introduced a watermark into the output logits of the Large Language Model (LLM) by embedding a periodic signal. Models trained using this watermarked LLM exhibit periodic signal characteristics in their outputs, making them detectable. This approach offers more robust and invisible watermarks compared to previous methods.

Similarly, in the case of classification LLMs, Zhao et al. [102] also adopted the approach of embedding a watermark by inserting periodic signals into the logits of the LLM output to enforce copyright protection. Specifically, this involves introducing periodic signals into the logits corresponding to a particular category, ensuring that models trained with data output from this modified model will also contain these periodic signals in the output for that specific category. However, this method inevitably impacts the quality of the output data, especially when users extract data using hard-labels (converting classification results into one-hot outputs) instead of continuous soft-labels. Future work could explore how to embed watermarks in classification LLMs with minimal impact on label quality for effective LLM copyright protection.

## 6.2 Academic Integrity

Academic integrity issues hold particular importance in today's educational sphere, especially given the ease of access and use of large language models (LLMs). Students might exploit these advanced models to complete assignments, papers, or even participate in exams, presenting new challenges to the upkeep of academic honesty. In tasks or exams where independent and original completion by students is required, it becomes necessary to devise methods to ascertain whether the submitted content is generated by a large language model.

The current work primarily explores the design of algorithms for automatically distinguishing text generated by Large Language Models (LLMs) from human-written text. For instance, Mitchell et al. [53] developed a GPT-based classifier, Detect-GPT, aimed at identifying LLM-generated text. However, such methods lack interpretability and may not be robust to out-of-domain text. To address this issue, an online text detection tool, GPTZero [3], operates on the assumption that LLM-generated texts can be differentiated from human texts based on two metrics: perplexity (Equation 12) and burstiness. Burstiness refers to the degree of uneven distribution in a text's length, complexity, or information density. Similarly, Vasilatos et al. [85] also employed the perplexity feature to distinguish between human and machine-generated texts. Nonetheless, detection methods based on perplexity and burstiness may be circumvented by deliberate text modifications. Concurrently, the promising technique of text watermarking remains underexplored in the field of academic integrity, which should become a significant research direction in the future.

## 6.3 Fake News Detection

The application of large language models has raised two main concerns: the generation of false information due to their text-generating capabilities, and the rapid dissemination of such false information. Firstly, the powerful text-generating capability of LLMs makes them an effective tool for producing fake information. These models can rapidly generate texts that appear authentic and accurate, but may actually contain erroneous or misleading information. Owing to the high credibility and realism of the generated texts, they can be easily utilized to construct fake news or false narratives, misleading the public and distorting facts. The second concern is the rapid spread of false information. Fake information generated by LLMs, due to its high degree of realism, can spread swiftly on social media and other digital platforms, leading to the proliferation and reinforcement of incorrect viewpoints [12, 105]. This spread not only amplifies the impact of false information but can also lead to public confusion and distrust towards authoritative information sources. Therefore, the automatic identification of news generated by LLMs is essential.

The current research has explored utilizing watermark technology to detect fake images and videos generated by AI models, aiding in the identification of fake news. For instance, a framework named DISSIMILAR[49] was proposed to track and detect false media information by automatically embedding watermarks in images before their release on social media. However, there has been minimal exploration in detecting false content generated by Large Language Models. We propose two potential approaches in this field. The first involves a method similar to the DISSIMILAR framework, where watermarks are added to text content before its publication on social media. This approach would use format-based methods (3.1) to embed watermarks without altering the text's content. The second approach necessitates collaboration with LLM providers, allowing them to embed watermarks and share watermark detection methods with certain platforms, thereby facilitating the marking of LLM-generated content. We recommend that future work should leverage text watermarking technology to aid in the detection of false information.

[3]https://gptzero.me/

## 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. These include balancing across different evaluation perspectives, adapting text watermarking for more challenging scenarios, developing more comprehensive benchmarks, and broadening the application of text watermarking. These challenges will be discussed in detail below.

### 7.1 Balancing Across Different Evaluation Perspectives

In section 5, we explore various perspectives for evaluating text watermarking algorithms. However, these perspectives often present inherent contradictions, making it extremely challenging for a text watermarking algorithm to excel in all evaluation perspectives simultaneously. For instance, achieving a favorable balance among success rate, text quality, and robustness at a high payload is difficult. In this section, we will first analyze why these perspectives are mutually contradictory, and then discuss potential strategies for achieving a better balance in future work.

#### 7.1.1 Why are the Different Perspectives Conflicting?

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 2, the score difference between watermarked and non-watermarked texts under the quality evaluation function $\mathcal{R}$ should be less than a threshold $\beta$. 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.

We begin by introducing conflicts among success rate, text quality, and robustness at a high payload. Payload and robustness involve different strategies for partitioning this limited text space. If the space is divided to encode more watermark messages (i.e., a larger payload), minor modifications to any watermarked text are more likely to result in detection as another watermark message, thus reducing robustness. Conversely, reducing the number of watermark messages encoded (i.e., lower payload) decreases the likelihood that modifications to watermarked text result in other watermark information, thereby increasing robustness. Hence, the conflict between payload and robustness is evident. Simultaneously, as text quality requirements increase, the size of the text space $|t_\beta|$ diminishes, potentially leading to a decrease in both payload and robustness, making the conflict between text quality and these two metrics obvious.

Lastly, we analyze why unforgeability and other evaluation perspectives might conflict. Typically, enhancing the complexity of watermarking algorithms can improve their unforgeability. However, incorporating additional complex modules into the algorithm often introduces greater robustness risks. Moreover, in publicly detectable scenarios, algorithms seeking to enhance unforgeability often conceal their detection methods using some watermark text space [14, 43], thereby conflicting with other evaluation approaches.

#### 7.1.2 Future Directions.

The current text watermarking algorithms primarily focus on balancing robustness and the impact on text quality. However, these efforts often do not simultaneously address payload and unforgeability, which should be the main focus of future work.

The key to balancing payload, robustness, and text quality lies primarily in devising a more effective strategy for partitioning the watermark text space. This may require additional designs to counter potential watermark removal attacks, dividing the watermark space into different watermark messages, ensuring that transitioning between different watermark messages necessitates a sufficient number of watermark removal attack operations. Secondly, from the perspective of the payload, it is feasible to draw inspiration from the concepts of error-correction codes, such as utilizing Hamming codes [36], to enhance the probability of recovering the original watermark information from partially modified text. These methods may effectively enhance payload and robustness while maintaining a consistent impact on text quality.

To enhance the unforgeability of text watermarks, it is generally necessary to utilize expertise from fields such as cryptography, information theory, and machine learning. This involves increasing the complexity of watermarking algorithms to improve their resistance to forgery. Although current methods have made some progress, their more intricate designs still introduce additional non-robust factors. Furthermore, these methods have not been extended to scenarios with larger payloads.

## 7.2 Adapting Text Watermarking for More Challenging Scenarios

Current watermark algorithms often achieve satisfactory results in relatively simple contexts. However, they require further enhancement and adaptation when confronted with more challenging scenarios, such as low-entropy scenarios and publicly detectable scenarios.

In low-entropy situations like coding, legal, and medical texts, diversity and complexity are typically reduced. These types of texts typically adhere to strict formatting (grammatical) requirements, so embedding watermarks without affecting these requirements is challenging. A more in-depth explanation is that for low-entropy text, the upper limit of the watermark text space is lower, thereby making it harder to find suitable watermark text. Future methods may need a stronger understanding of their formatting or grammatical requirements, and thereby designing semantically invariant format transformations to expand the available watermark text space.

In scenarios where watermarks are publicly detectable, both the presence of the watermark and its detection mechanism are openly visible. This poses greater challenges for the design of watermark algorithms. Firstly, the algorithm must be sufficiently complex and unpredictable to ensure that, even in a public setting, attackers struggle to effectively corrupt or alter the watermark information. Secondly, the design must ensure that the watermark's generation method cannot be inferred even through its detector. This primarily involves considerations of unforgeability, but in publicly detectable scenarios, there are also heightened demands for robustness and text quality. Future methods will need to integrate considerations of security and practicality, potentially involving more intricate encryption and machine learning technologies.

## 7.3 Developing More Comprehensive Benchmarks

Current research in text watermarking benchmark primarily focuses on text quality, with limited benchmarks for other critical metrics like high success rate with large payload, robustness, and unforgeability. Therefore, developing a more comprehensive benchmarking system is a crucial direction for future research. Constructing such benchmarks requires significant effort, taking into account various application scenarios, attack methods, and characteristics of different watermarking algorithms. It also necessitates establishing a fair, transparent, and user-friendly evaluation process, allowing researchers to test and compare algorithms under unified standards. This benchmarking system will not only advance academic research into text watermarking algorithms but also aid the industry in better understanding and applying these technologies.

## 7.4 Broadening the Application of Text Watermarking

Although text watermarking technology has demonstrated its practicality in multiple domains, its wider application necessitates further efforts. This encompasses not only advancements in watermarking techniques but also factors beyond the technical realm. In this section, we will explore the challenges faced in expanding the applications of text watermarking technology, particularly from the perspectives of large language model providers, public trust, and transparency.

#### 7.4.1 Limited Engagement of Large Language Model Providers.
As an increasing amount of text is generated directly by large language models, it is crucial for providers of these LLMs to integrate text watermarking functionality into their services to promote the use of text watermarks. However, the current level of engagement from these providers in text watermarking technology remains insufficient, influenced by various technical and non-technical factors.

Firstly, current text watermarking algorithms cannot guarantee no reduction in text quality, which may impact the service quality of model providers. Technically, this demands future text watermarking algorithms to consider the impact on text quality more thoroughly. Additionally, the direct benefit for LLM providers from text watermarking technology lies in the protection of the LLMs' copyrights. More focus on research in this area is needed to encourage providers to participate more actively in promoting text watermarking technology.

Moreover, non-technical factors also affect the engagement level of large model providers. For instance, the role of governments and regulatory bodies is significant. There needs to be a discussion on how governments should use legal restrictions or incentives to encourage model providers to adopt watermarking technology. This could involve setting relevant standards and norms, as well as providing economic incentives to encourage the adoption of these technologies.

#### 7.4.2 Lack of Public Trust.
Public trust and transparency in text watermarking technology are key factors in promoting its widespread application. Only when the public trusts the text watermarking algorithms and believes in the accuracy of their detection results can they be effectively utilized in practical applications. To enhance public trust, it is necessary to ensure the transparency and reliability of watermarking technology.

A fundamental step towards this goal involves the comprehensive disclosure of the text watermarking detection algorithms. Making these details accessible allows users to grasp and assess the principles and accuracy of the algorithms. Such transparency not only cultivates user trust but also spurs further academic and industrial advancements.

Moreover, involving independent third-party platforms for detection and verification can strengthen trust. These platforms offer unbiased evaluations, alleviating conflict of interest concerns. Furthermore, government and regulatory guidelines can ensure the technology's fairness and transparency, further boosting public confidence.

## 8 CONCLUSION

This survey has comprehensively explored the landscape of text watermarking in the era of large language models (LLMs). Our investigation covered various aspects, including the implementation text watermarking methods, different perspective of evaluation methods of text watermarking, and applications in areas such as copyright protection, academic integrity, and fake news detection.

One of the critical findings of this survey is the evolution of text watermarking techniques in tandem with the advancements in LLMs. The robustness of watermarking methods against various levels of attacks—character-level, word-level, and document-level—highlights the complexity and sophistication needed in watermark designs to combat increasingly advanced removal strategies. The discussion on unforgeability, especially in the context of private and public detection scenarios, sheds light on the necessity of developing watermarks that are not only difficult to remove but also challenging to replicate or forge.

Despite the progress made, several areas require further exploration. The balance between robustness, watermark payload and impact on text quality remains a crucial challenge, as does the need for watermarking methods that can adapt to the evolving capabilities of LLMs. Additionally, the integration of watermarking techniques in real-world applications presents practical challenges, including scalability, legal considerations, and ethical implications.

Future research should focus on developing more advanced watermarking algorithms that can withstand new forms of attacks, especially in scenarios where attackers have access to more sophisticated tools and knowledge. Exploring the use of watermarking in new domains, such as verifying the authenticity of AI-generated content in social media and journalism, can provide new avenues for maintaining the integrity and trustworthiness of digital content.

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

## REFERENCES


[1] 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.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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).

[8] 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).

[9] Miranda Christ, Sam Gunn, and Or Zamir. 2023. Undetectable Watermarks for Language Models. *arXiv preprint arXiv:2306.09194* (2023).

[10] Timothy Chu, Zhao Song, and Chiwun Yang. 2023. How to Protect Copyright Data in Optimization of Large Language Models? *arXiv preprint arXiv:2308.12247* (2023).

[11] 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).

[12] Renee DiResta. 2020. AI-generated text is the scariest deepfake of all. *Wired, Jul* (2020).

[13] Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpacafarm: A simulation framework for methods that learn from human feedback. *arXiv preprint arXiv:2305.14387* (2023).

[14] 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.

[15] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long form question answering. *arXiv preprint arXiv:1907.09190* (2019).

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

[17] Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. 2023. Three bricks to consolidate watermarks for large language models. *arXiv preprint arXiv:2308.00113* (2023).

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

[19] Riley Goodside. 2023. There are adversarial attacks for that proposal as well — in particular, generating with emojis after words and then removing them before submitting defeats it. https://twitter.com/goodside/status/1610682909647671306.

[20] 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.

[21] 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.

[22] 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).

[23] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems* 28 (2015).

[24] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751* (2019).

[25] Abe Bohan Hou, Jingyu Zhang, Tianxing He, Yichen Wang, Yung-Sung Chuang, Hongwei Wang, Lingfeng Shen, Benjamin Van Durme, Daniel Khashabi, and Yulia Tsvetkov. 2023. SemStamp: A Semantic Watermark with Paraphrastic Robustness for Text Generation. *arXiv preprint arXiv:2310.03991* (2023).

[26] Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. 2023. Unbiased Watermark for Large Language Models. *arXiv preprint arXiv:2310.10669* (2023).

[27] Vojtěch Hudeček and Ondřej Dušek. 2023. Are LLMs All You Need for Task-Oriented Dialogue? *arXiv preprint arXiv:2304.06556* (2023).

[28] 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.

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

[30] 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.

[31] 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

[32] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. 2023. On the Reliability of Watermarks for Large Language Models. *arXiv preprint arXiv:2306.04634* (2023).

[33] 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. *arXiv preprint arXiv:2303.13408* (2023).

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

[35] Sumith Kulal, Panupong Pasupat, Kartik Chandra, Mina Lee, Oded Padon, Alex Aiken, and Percy S Liang. 2019. Spoc: Search-based pseudocode to code. *Advances in Neural Information Processing Systems* 32 (2019).

[36] UK Kumar and BS Umashankar. 2007. Improved hamming code for error detection and correction. In *2007 2nd International Symposium on Wireless Pervasive Computing*. IEEE.

[37] 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).

[38] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).

[39] Jingjing Li, Zichao Li, Tao Ge, Irwin King, and Michael R Lyu. 2022. Text Revision by On-the-Fly Representation Optimization. (2022), 10956–10964.

[40] 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.

[41] Mingjie Li, Hanzhou Wu, and Xinpeng Zhang. 2023. A novel watermarking framework for intellectual property protection of NLG APIs. *Neurocomputing* 558 (2023), 126700.

[42] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics, Barcelona, Spain, 74–81. https://aclanthology.org/W04-1013

[43] 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]

[44] Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. 2023. A Semantic Invariant Robust Watermark for Large Language Models. *arXiv preprint arXiv:2310.06356* (2023).

[45] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics* 8 (2020), 726–742.

[46] Yixin Liu, Hongsheng Hu, Xuyun Zhang, and Lichao Sun. 2023. Watermarking Text Data on Large Language Models for Dataset Copyright Protection. *arXiv preprint arXiv:2305.13257* (2023).

[47] Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. Www'18 open challenge: financial opinion mining and question answering. In *Companion proceedings of the the web conference 2018*. 1941–1942.

[48] David Megías, Minoru Kuribayashi, Andrea Rosales, Krzysztof Cabaj, and Wojciech Mazurczyk. 2022. Architecture of a fake news detection system combining digital watermarking, signal processing, and machine learning. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications* 13, 1 (2022), 33–55.

[49] David Megías, Minoru Kuribayashi, Andrea Rosales, and Wojciech Mazurczyk. 2021. DISSIMILAR: Towards Fake News Detection Using Information Hiding. In *Signal Processing and Machine Learning. In The 16th International Conference on Availability, Reliability and Security (Vienna, Austria)(ARES 2021). Association for Computing Machinery, New York, NY, USA, Article*, Vol. 66.

[50] 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.

[51] 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).

[52] Nighat Mir. 2014. Copyright for web content using invisible text watermarking. *Computers in Human Behavior* 30 (2014), 648–653.

[53] Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. *arXiv preprint arXiv:2301.11305* (2023).

[54] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 839–849.

[55] 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).

[56] 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.

[57] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474* (2022).

[58] OpenAI. 2023. GPT-4 Technical Report. *ArXiv* abs/2303.08774 (2023). https://api.semanticscholar.org/CorpusID:257532815

[59] Hao Peng, Xiaozhi Wang, Shengding Hu, Hailong Jin, Lei Hou, Juanzi Li, Zhiyuan Liu, and Qun Liu. 2022. COPEN: Probing conceptual knowledge in pre-trained language models. *arXiv preprint arXiv:2211.04079* (2022).

[60] Wenjun Peng, Jingwei Yi, Fangzhao Wu, Shangxi Wu, 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. *arXiv preprint arXiv:2305.10036* (2023).

[61] 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.

[62] 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

[63] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[64] 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.

[65] 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.
[66] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).
[67] Jie Ren, Han Xu, Yiding Liu, Yingqian Cui, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. 2023. A Robust Semantics-based Watermark for Large Language Model against Paraphrasing. *arXiv preprint arXiv:2311.08721* (2023).
[68] 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*. 97–104.
[69] 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]
[70] Ryoma Sato, Yuki Takezawa, Han Bao, Kenta Niwa, and Makoto Yamada. 2023. Embarrassingly Simple Text Watermarks. *arXiv preprint arXiv:2310.08920* (2023).
[71] 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. *arXiv preprint arXiv:2203.13224* (2022).
[72] 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).
[73] 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.
[74] Yuki Takezawa, Ryoma Sato, Han Bao, Kenta Niwa, and Makoto Yamada. 2023. Necessary and sufficient watermark for large language models. *arXiv preprint arXiv:2310.00833* (2023).
[75] 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).
[76] Ruixiang Tang, Yu-Neng Chuang, and Xia Hu. [n. d.]. Secure Your Model: A Simple but Effective Key Prompt Protection Mechanism for Large Language Models. ([n. d.]).
[77] 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).
[78] 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.
[79] 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).
[80] 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
[81] 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.
[82] 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).
[83] 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).
[84] 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.
[85] 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).
[86] Jingtan Wang, Xinyang Lu, Zitong Zhao, Zhongxiang Dai, Chuan-Sheng Foo, See-Kiong Ng, and Bryan Kian Hsiang Low. 2023. WASA: WAtermark-based Source Attribution for Large Language Model-Generated Data. *arXiv preprint arXiv:2310.00646* (2023).
[87] Lean Wang, Wenkai Yang, Deli Chen, Hao Zhou, Yankai Lin, Fandong Meng, Jie Zhou, and Xu Sun. 2023. Towards codable text watermarking for large language models. *arXiv preprint arXiv:2307.15992* (2023).
[88] 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).

[89] 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*. 1–10.

[90] 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).

[91] 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.

[92] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600* (2018).

[93] 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.

[94] KiYoon Yoo, Wonhyuk Ahn, and Nojun Kwak. 2023. Advancing Beyond Identification: Multi-bit Watermark for Language Models. *arXiv preprint arXiv:2308.00221* (2023).

[95] Jifan Yu, Xiaozhi Wang, Shangqing Tu, Shulin Cao, Daniel Zhang-Li, Xin Lv, Hao Peng, Zijun Yao, Xiaohan Zhang, Hanming Li, et al. 2023. KoLA: Carefully Benchmarking World Knowledge of Large Language Models. *arXiv preprint arXiv:2306.09296* (2023).

[96] 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).

[97] 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).

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

[99] Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. 2023. Benchmarking large language models for news summarization. *arXiv preprint arXiv:2301.13848* (2023).

[100] 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]

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

[102] Xuandong Zhao, Lei Li, and Yu-Xiang Wang. 2022. Distillation-resistant watermarking for model protection in nlp. *arXiv preprint arXiv:2210.03312* (2022).

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

[104] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *arXiv preprint arXiv:2306.05685* (2023).

[105] 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

[106] 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).