Post-Hoc Watermarking for Robust Detection in Text Generated by Large Language Models

Jifei Hao[1], **Jipeng Qiang**[1]*, **Yi Zhu**[1], **Yun Li**[1], **Yunhao Yuan**[1], **Xiaoye Ouyang**[2]
[1] School of Information and Engineering, Yangzhou University, Jiangsu, China
[2] China Academy of Electronic and Information Technology, Beijing, China
haojifei2000@gmail.com, {jpqiang, zhuyi, yhyuan, liyun}@yzu.edu.cn
ouyangxiaoye@cetc.com.cn

Abstract

The most effective methods for detecting text generated by large language models (LLMs) involve inserting a detectable watermark during the model's decoding process. However, these watermarking techniques typically embed watermarks during text generation by adjusting the output probability distribution to favor certain words, which LLM API providers are reluctant to share due to concerns about model distillation. Therefore, these methods are inapplicable when only black-box language models are available. In this work, we propose a novel post-hoc method to inject watermarks into generated text autonomously, enhancing robustness and semantic faithfulness. For robust natural watermarking, we identify features that are semantically or syntactically fundamental to the text and thus invariant to minor modifications. To preserve semantic faithfulness, we employ a synonym generation method based on paraphrasing for watermark embedding. Experimental results demonstrate that our approach achieves higher detectability while maintaining superior semantics and surpasses previous methods in robustness across four datasets and four types of corruption.

1 Introduction

The rise of human-like language models such as ChatGPT has raised concerns about their potential misuse for malicious purposes (Editorials, 2023; Liebrenz et al., 2023). Detecting AI-generated text has become crucial as the distinction between human and machine-generated content blurs (Wu et al., 2023; Mitchell et al., 2023). OpenAI's classifier identifies only 26% of AI-generated text [1], and these classifiers are prone to adversarial attacks and biases against non-native writers, leading to higher false positive and negative rates (Jin et al.,

Generated Text

At the depths of the ocean, the pressure can be extremely high **because** there is a lot of water above pushing down on the water below. However, many creatures that live **at** **those** depths **have** ***evolved special*** features that ***allow*** them to ***withstand*** the high pressure. For ***example***, some deep-***sea creatures*** have thick, strong shells or exoskeletons that protect their bodies from **being** crushed by the pressure. Others have flexible ***bodies*** or special structures that ***help*** them withstand the pressure.

Watermarking by WTGB

At the depths of the ocean, the pressure can be extremely high because there is a lot of water above pushing down on the water below. However, many creatures that live at those depths have ***adapted particular*** features that ***enable*** them to ***survive*** the high pressure. For ***instance***, some deep-***ocean animals*** have thick, strong shells or exoskeletons that protect their bodies from being crushed by the pressure. Others have flexible ***body*** or special structures that ***assist*** them withstand the pressure.

Watermarking by Ours

At the depths of the ocean, the pressure can be extremely high **since** there is a lot of water above pushing down on the water below. However, many creatures that live **near** **these** depths **were** evolved special features that allow them to withstand the high pressure. For example, some deep-sea creatures have thick, strong shells or exoskeletons that protect their bodies from **getting** crushed by the pressure. Others have flexible bodies or special structures that help them withstand the pressure.

Figure 1: An original text generated by the black-box language model (GPT-3.5) was watermarked using both our watermarking method and the baseline (Yang et al., 2023) (WTGB). The detection results show that words with a green background carry watermark information, while words with a red background do not. The watermarking method is to increase the number of words with a green background (Words displayed in bold). Unlike the WTGB method, which may choose all words to embed watermarks, our method selects only some words that are invariant to minor corruption.

* Corresponding authors.

[1] https://openai.com/blog/new-ai-classifier-for-indicating-ai-written-text

Proceedings of the 31st International Conference on Computational Linguistics, pages 5430–5442
January 19–24, 2025.

2020). Consequently, watermark-based detection methods are gaining significant attention (Kirchenbauer et al., 2023).

Primary watermark-based methods are designed for white-box models, enabling watermark embedding during text generation by modifying the output probability distribution to favor specific words (Kirchenbauer et al., 2023). This process involves selecting random "green list" words based on the hash value of the previous token, subtly influencing the sampling process to promote the use of these words, facilitating downstream detection. However, these methods are limited to model owners who can control the token sampling process. This paper, however, focuses on injecting watermarks into already-generated text from black-box language models.

Recently, a watermarking method known as WTGB via lexical substitution was introduced (Yang et al., 2023). This method embeds a watermark by replacing specific words in the input text with contextually appropriate synonyms using pre-trained BERT models. However, it faces two major issues: (1) Maintaining semantic faithfulness is challenging, as substitutes often compromise the original meaning due to contextual influences. (2) The watermark is susceptible to adversarial attacks and easy removal. Although every word can serve as a watermark embedding point, enhancing detectability, it reduces the robustness of the watermarked text.

In this work, we propose a post-hoc watermarking method to embed watermarks into text generated by LLMs. Our watermark embedding method comprises two main stages: selecting the embedding positions and embedding the watermarks. To preserve the original semantics during the embedding process, we employ a paraphrase-based lexical substitution method that maintains semantic integrity. To ensure the high robustness of the watermarked text, we choose words that are semantically or syntactically fundamental to the text as anchors, ensuring they remain invariant to minor modifications and accurately locate the watermark positions.

Experimental results indicate that even with a minimal number of watermark points added, the generated watermark text maintains high detectability and preserves semantic integrity. Our method exhibited enhanced robustness against various attacks, including word substitution, word deletion, paraphrasing, and re-translation. Our code and data are source-opened in Github [2].

2 Related Work

Text watermarking algorithms for the generated text of large language models can be categorized into two classifications (Liu et al., 2024; Zhu et al., 2023). The first type is white-box language model-based watermarking algorithms, which inject watermark information during the text generation process, involving modifications to LLM itself. The second type is black-box language model-based watermarking algorithms, which inject watermark information by making modifications to existing text without requiring knowledge of the internal details of the LLM.

White-box watermarking methods. The first LLM-based watermarking method, proposed by Kirchenbauer et al. (2023), is a zero-bit watermarking technique for white-box language models. It injects watermark information during inference by dividing the model's vocabulary into "red" and "green" lists based on the hash value of the previous token, promoting the use of "green list" words. Zhao et al.(Zhao et al., 2023) introduced GPTWatermark, which splits the vocabulary into two sets using a randomly generated watermark key, enhancing the edit robustness of watermarked text. Subsequently, a series of white-box watermarking methods were proposed (Hu et al., 2023; Fernandez et al., 2023; Ren et al., 2023).

Black-Box watermarking methods. The core concept of black-box language model-based watermarking involves altering generated text to create watermarked text. Black-box watermarking methods are categorized into multi-bit and zero-bit watermarking. Multi-bit watermarking (Topkara et al., 2006; Munyer and Zhong, 2023; Yang et al., 2023; Qiang et al., 2023d; Yoo et al., 2023), extensively researched for intellectual property protection, embeds multi-bit information but has weaker anti-attack capabilities due to the need for information recovery. Zero-bit watermarking, less researched, only determines if a watermark is present, leading to stronger anti-attack abilities.

Focusing on zero-bit watermarking, Yang et al. (2023) proposed a method via lexical substitution. They used a binary encoding function to compute a random binary encoding for words, with non-watermarked text encodings following a Bernoulli distribution where bit-1 is represented with a like-

[2]https://github.com/AlfredWatson/RSFPH-WTGBBLM

lihood of 0.5. To embed a watermark, they selectively substituted words representing bit-0 with contextually appropriate synonyms for bit-1. A statistical test is used for watermark detection. This work proposes a novel method to enhance both robustness and semantic faithfulness in watermarking.

3 Problem Definition

The problem of text watermarking in black-box language models can be conceptualized as comprising two main components. The first component is **Watermark Embedding**, represented as $\mathcal{W}(\cdot)$, which inserts a watermark into a given text sequence $X = \{x_1, \ldots, x_i, \ldots\}$, producing the watermarked text $X_w = \mathcal{W}(X)$. The second component is **Watermark Detection**, which involves analyzing a given text to determine whether it contains an embedded watermark.

In adversarial scenarios, attackers aim to remove the watermark from the text through methods such as word deletion, substitution, document polishing, or re-translation. The objective of such attacks is to evade detection by the watermarking system while preserving the original semantic content of the text. Consequently, an effective watermarking strategy must satisfy the following three key requirements: **1)Detection Accuracy**: The watermark detection mechanism should achieve high accuracy with minimal empirical error rates. **2) Semantic Integrity**: The watermarked text must retain the original meaning with a high degree of fidelity. **3) Robustness Against Attacks**: The watermark should be resilient to various adversarial attempts to alter or remove it.

To fulfill these requirements, we propose a watermarking strategy comprising the following components: **1) Robust Feature Selection**: Leveraging features in the text that are inherently resistant to minor modifications, inspired by robust watermarking techniques (Yoo et al., 2023), to serve as stable anchors for embedding the watermark. **2) Advanced Paraphrasing Techniques**: Employing a paraphraser-based lexical substitution model (Qiang et al., 2023c) to generate alternative phrasing that preserves semantic integrity while embedding the watermark. **3) Comprehensive Evaluation**: Conducting rigorous testing against a wide range of adversarial attacks, including both word-level and document-level modifications, to assess and ensure robustness.

By integrating these elements, our approach achieves a reliable and secure watermarking mechanism for text generated by black-box language models, addressing the critical requirements of detection accuracy, semantic preservation, and robustness against adversarial manipulations.

4 Proposed Method

4.1 Overview

As illustrated in Figure 2, our proposed method consists of two primary components: **Watermark Embedding** and **Watermark Detection**.

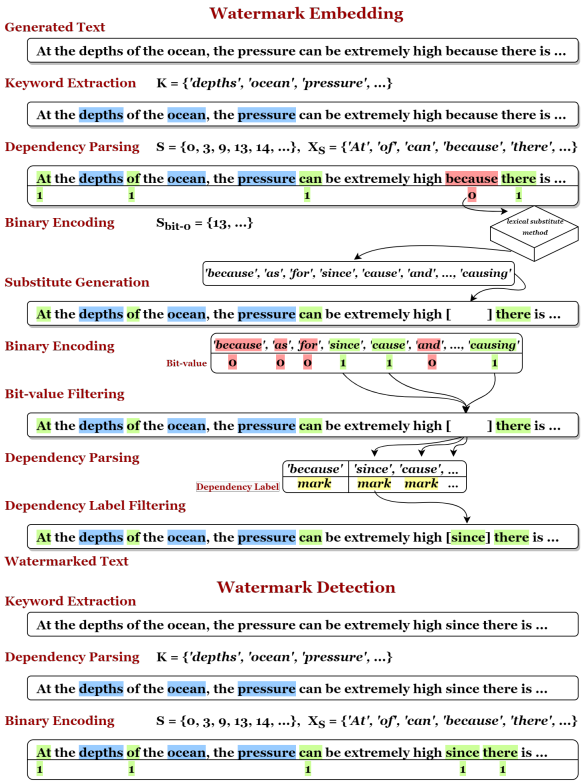

Figure 2: The process of the proposing Watermark method.

Watermark Embedding involves two critical steps: Watermark Embedding Positions Selection and Embedding Watermark Information. To ensure robustness and preserve the original semantics, the embedding positions are chosen based on features that exhibit minimal sensitivity to textual alterations. These features are derived from semantic and syntactic invariants of natural language, as suggested by Yoo et al. (2023).

In the first step, we identify the set of embedding positions, denoted as $S = s_1, \ldots, s_i, \ldots$, along with the corresponding word set $X_S = x_{s_1}, \ldots, x_{s_i}, \ldots$. The selection process ensures

that these positions are strategically chosen to balance robustness and semantic preservation. Words in X_S are then encoded to represent binary values (bit-0 or bit-1) in a balanced manner. To achieve this balance, we apply the binary encoding technique proposed by Yang et al. (2023). For each word $x_{s_i} \in X_S$, a bit-value b_{s_i} is computed using the bit-wise XOR hash of x_{s_i} and its preceding word x_{s_i-1} as a seed.

$$b_{s_i} = Binary(hash(x_{s_i-1}) \oplus hash(x_{s_i})) \quad (1)$$

Where $\oplus$ represents the bit-wise XOR operation, and the $Binary()$ function maps the computed hash value to a binary bit. This encoding ensures that bit-0 and bit-1 are distributed evenly across the text, with the same token potentially representing different bits depending on its preceding context.

In the second step, watermark embedding is achieved by selectively replacing tokens. For each embedding position s_i in S, the corresponding bit-value b_{s_i} is computed. All positions with bit-0 values are filtered to form the set $S_{\text{bit-0}}$, and their associated words make up $X_{S_{\text{bit-0}}}$. Lexical substitution techniques are then employed to generate candidate replacements for each word in $X_{S_{\text{bit-0}}}$. From these candidates, we select the replacement tokens that satisfy two conditions: they represent bit-1 and maintain the semantic and syntactic coherence of the text. This process ensures that the bit-values of tokens at embedding positions are significantly biased towards 1 in the watermarked text, enabling statistical detection of the watermark.

Watermark detection process involves two steps: Candidate Position Identification and Watermark Sequence Verification. In the first step, potential watermark embedding positions are identified using the same selection criteria as in the embedding phase, ensuring consistency. In the second step, the detected sequence of bit-values is compared against the expected watermark sequence to verify its presence. Detailed explanations of these steps are provided in Section 4.3.

4.2 Watermark Embedding

The process of watermark embedding is outlined in algorithm 1.

Watermark Embedding Position Selection. To determine optimal positions for embedding the watermark, we begin by excluding key elements crucial for preserving the semantic integrity of the text. These elements include proper nouns such as names, locations, and domain-specific terms, identified through named entity recognition models. Since replacing these terms with synonyms could alter the intended meaning of the text, they are excluded from modification. Furthermore, our watermarking framework supports customization, allowing users to extend the exclusion list based on specific contextual requirements. Inspired by Campos et al. (Campos et al., 2018), we adopt a feature-based approach to extract semantically significant words.

Algorithm 1 Watermark Embedding

Input: Original text $X = \{x_1, x_2, \ldots\}$, number of substitutes T
Output: Watermarked text X_w

$X_w \leftarrow X$
$K \leftarrow$ KeywordsExtracter(X)
$S \leftarrow$ DependencyParser(X, K)
for each $s \in S$ **do**
  if Binary$(hash(x_{s-1}) \oplus hash(x_s)) == 0$ **then**
    $Y \leftarrow$ Generate top N substitutes
    for each $y \in Y$ **do**
      if $Binary(hash(x_{s-1}) \oplus hash(y)) == 1$ & $Dep(X, s) == Dep(X[y/x_s], s)$ **then**
        Replace the word at position s in X_w into y
        Break
      end if
    end for
  end if
end for
return X_w

The union of the semantically significant words and proper nouns forms the keyword set $K = k_1, k_2, \ldots$. Dependency parsing is then applied to the input text sequence X using a dependency parser $Dep(X, i)$, which generates dependency labels for each token x_i in X and constructs a dependency parse tree. By calculating entailment scores for all dependency relations, as outlined in Yoo et al. (2023), a dependency label ranking table is established to prioritize words in the text. Text keywords K, punctuation, and user-defined keywords are subsequently filtered out to derive the embedding position set $S = s_1, \ldots, s_i, \ldots$ and the corresponding word set $X_S = x_{s_1}, \ldots, x_{s_i}, \ldots$, with $|S| = |X_S| = m$. Each word x_{s_i} in X_S is then assigned a bit-value using Equation 1. Words with a bit-value of 0 are selected as watermark embedding positions, forming the set $S_{\text{bit-0}}$ and the associated words $X_{S_{\text{bit-0}}}$.

Embedding Watermark Information. For each word $x \in X_{S_{\text{bit-0}}}$, the goal is to identify a substitute y that satisfies the following criteria: (1) the overall sentence meaning remains unchanged,

and (2) the bit-value of y is 1, while preserving the dependency label of x.

To ensure semantic preservation, we employ the lexical substitution method proposed by Qiang et al. (2023c,b,a) to generate the top T substitutes $Y = y_1, \ldots, y_T$. This method employs a decoding strategy focused on lexical variations while maintaining the sentence's meaning.

Consistency in embedding positions between the original text X and the watermarked text X_w is critical for accurate detection. Consequently, we iteratively evaluate candidates in Y from the beginning to find a word y with a bit-value of 1, as determined by Equation 1, while preserving the dependency label of x. If no suitable substitute y is identified, the original word is retained to preserve the text's meaning.

4.3 Watermark Detection

Algorithm 2 Watermark Detection

Input: Original text X
Output: Detection result (True/False)

$K \leftarrow \text{KeywordsExtracter}(X)$
$S \leftarrow \text{DependencyParser}(X, K)$
$m \leftarrow |S|$
Initialize two variables $\text{bit}_0 \leftarrow 0$, $\text{bit}_1 \leftarrow 0$ to record the number of words in bit-0 and bit-1.
for each $s \in S$ **do**
  $bit \leftarrow Binary(hash(x_{s-1}) \oplus hash(x_s))$
  if $bit == 1$ **then**
    $\text{bit}_1 \leftarrow \text{bit}_1 + 1$
  else
    $\text{bit}_0 \leftarrow \text{bit}_0 + 1$
  end if
end for
$\hat{p} \leftarrow \text{bit}_1/(\text{bit}_0 + \text{bit}_1)$
$z = \frac{\hat{p}-1/2}{\sqrt{1/4m}}$
if $z > z_\alpha$ **then**
  return True {Watermark detected}
else
  return False {No watermark detected}
end if

The watermark detection process is depicted in algorithm 2. For any watermark-free text, we utilize the method introduced by Watermark Embedding Position Selection to determine the embedding position set S. As outlined in Section 4.2, for each word in X_S, the likelihood of representing bit-0 and bit-1 is nearly equal. During the watermark embedding process, we enhance the frequency of words representing bit-1. Consequently, watermark detection can be achieved by testing the following null hypothesis $\mathcal{H}_0$: "The text is natural without injected watermark information."

Based on the null hypothesis, each bit-value b_{s_i} of each word x_{s_i} obeys the binomial distribution of $n = 1, p = 1/2$, i.e., $b_{s_i} \sim B(1, 1/2)$. For all words in X_S, the bit-value $b_{s_1}, \ldots, b_{s_i}, \ldots, b_{s_m}$ are independently and identically distributed with mathematical expectation $E(b_{s_i}) = p$ and variance $D(b_{s_i}) = p(1-p)$. Following Lindburg-Levy theorem, we can obtain that when m is sufficiently large, the sum of independent identically distributed random variables $\sum_{i=1}^{m} b_{s_i}$ approximately obeys the normal distribution $N(mp, mp(1-p))$, and the standardized random variable $\frac{\sum_{i=1}^{m} b_{s_i} - mp}{\sqrt{mp(1-p)}} = \frac{\frac{1}{m}\sum_{i=1}^{m} b_{s_i} - p}{\sqrt{p(1-p)/m}}$ approximately obeys the standard normal distribution $N(0, 1)$. Where $\frac{1}{m}\sum_{i=1}^{m} b_{s_i}$ denotes the proportion of tokens with bit-1 at all watermark embedding positions, which we set to $\hat{p}$. We use the one proportion z-test to evaluate $\mathcal{H}_0$.

$$z = \frac{\hat{p} - 1/2}{\sqrt{1/4m}} \tag{2}$$

where α is the level of significance, indicating the probability of rejecting the null hypothesis when it is true. We compare the test statistic z with the critical value z_α, and if $z > z_\alpha$, then $\mathcal{H}_0$ will be rejected and conclude that the detected bit value of the token at the embedded position is significantly different compared to the natural state, indicating that there is watermarking of the text. So the statistic z can be used as a measure of the strength of the watermark, as the z score of the text is larger, its watermark strength is stronger.

5 Experiments

5.1 Experimental Setup

Dataset. In line with Yang et al. (2023), we selected four English text datasets from the HC3 corpus[3] to evaluate our method: wiki_csai, open_qa, medicine, and reddit_eli5. From each dataset, we sampled 200 English text instances generated by ChatGPT, with an average length of 200 tokens per instance. Preprocessing steps included text normalization to remove invalid tags, unnecessary spaces, newlines, and other extraneous characters. Half of the samples in each dataset were embedded with watermarks, and the remaining samples were left unwatermarked. The test set was constructed by mixing the watermarked and unwatermarked samples.

[3]https://huggingface.co/datasets/Hello-SimpleAI/HC3

Datasets	Method	FPR(↓)	TNR(↑)	TPR(↑)	FNR(↓)	Acc(↑)	Sim(↑)
wiki_csai	WTGB	0.08	0.92	0.99	0.01	.96	$.9946_{\pm.0050}$
	Ours	**0.06**	**0.94**	**1.00**	**0.00**	**.97**	$\mathbf{.9947}_{\pm.0038}$
open_qa	WTGB	**0.06**	**0.94**	1.00	0.00	**.97**	$\mathbf{.9956}_{\pm.0046}$
	Ours	0.09	0.91	1.00	0.00	.96	$.9912_{\pm.0073}$
medicine	WTGB	0.16	0.84	1.00	0.00	.92	$.9921_{\pm.0096}$
	Ours	**0.08**	**0.92**	1.00	0.00	**.96**	$\mathbf{.9935}_{\pm.0061}$
reddit_eli5	WTGB	0.09	0.91	**1.00**	**0.00**	.96	$.9922_{\pm.0066}$
	Ours	**0.05**	**0.95**	0.99	0.01	**.97**	$\mathbf{.9937}_{\pm.0050}$

Table 1: The results of Watermarking Methods on four Datasets. The best results are in bold.

Implementation Details. Our watermarking framework operates at the sentence level, with sentence segmentation handled by NLTK. Named entity recognition, keyword extraction, and dependency parsing to obtain dependency relation labels were performed using SpaCy's 'en-core-web-sm' model (Honnibal and Montani, 2017). For the lexical substitution process, the top $T = 100$ candidate words were generated for each token. During watermark detection, we employed a one-proportion z-test with a significance level of $\alpha = 0.05$.

Compared Methods. We compared our approach against the watermarking method for black-box language models introduced by Yang et al. (2023) (WTGB). Following their experimental setup, we used the default hyperparameters: $\lambda = 0.83$, a sentence embedding similarity threshold $\tau_{sent} = 0.8$, and a word-level similarity threshold $\tau_{word} = 0.8$. For watermark detection, we applied the high-precision 'Precise Detection' method described in their work.

Metrics. We evaluated our watermarking method using four key dimensions: watermark strength, detection accuracy, semantic integrity of the watermarked text, and robustness of the watermarked text. Since detection accuracy correlates positively with watermark strength, both were assessed simultaneously. Binary classification metrics, such as false positive rate (FPR), true negative rate (TNR), true positive rate (TPR), and false negative rate (FNR), were used. Stronger watermarks produce higher z-scores, thereby enhancing detection accuracy (Acc). The sensitivity of the binary classifier was visualized through ROC curves based on observed z-scores. For semantic integrity, we adhered to the methodology of Yang et al. (2023) and employed the Sentence-Transformer model 'sentence-t5-xxl' [4] to compute semantic similarity scores between the watermarked and original texts. To evaluate robustness, we subjected the watermarked texts to both word-level and sentence-level attacks and assessed the effectiveness of the watermarking method using the F1 score and the AUC (area under the ROC curve). Detailed robustness analyses are provided in Section 5.4.

5.2 Watermark strength Analysis

As shown in Table 1, our method consistently achieves higher accuracy and better semantic similarity compared to WTGB across the datasets, with lower false positive rates in most cases. On the medicine dataset, where the detection accuracy(Acc) is lowest, we observe a maximum of $200 * 0.08 = 16$ type-I errors. The results suggest that our watermarking technique not only maintains the integrity of the original text better but also offers stronger detection performance.

To demonstrate the trade-off between sensitivity (TPR) and specificity (TNR) at different thresholds for our watermarking method and the WTGB method, we plotted ROC curves for the four test sets, as shown in Figure 3. The AUC values indicate that there is little difference between our method and the WTGB method, both achieving nearly perfect classification performance.

5.3 Text Quality Analysis

We calculated the semantic similarity score between each watermarked text and its original text in each test set. The similarity scores of all watermarked samples were then aggregated to obtain the average similarity for each test set.

As shown in Table 1, the semantic similarity scores between the watermarked and original texts indicate that our method outperforms the WTGB method on three datasets. This advantage is be-

[4]https://huggingface.co/sentence-transformers/sentence-t5-xxl

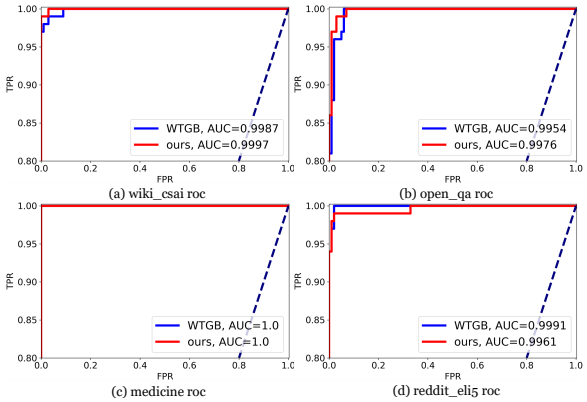

Figure 3: The watermark detection of ROC curve and AUC (area under the curve) value under different watermark methods on four datasets. An AUC value of 0.5 indicates random chance, while values closer to 1 indicate more perfect classification, thus indicating stronger watermarking.

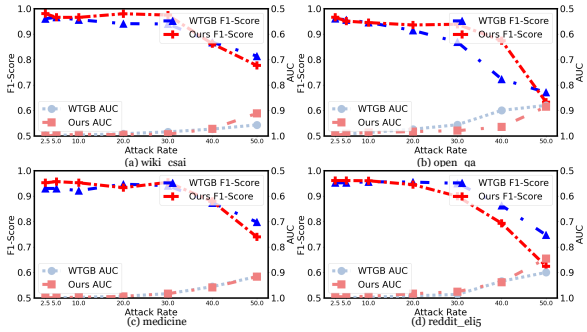

Figure 4: Robustness analysis of the watermark under word deletion attacks.

cause the WTGB watermarking strategy considers nearly every word as a potential watermarking candidate. Even with POS filtering, any word might be replaced with a synonym, leading to an overall decrease in the semantic value of the watermarked text. In contrast, our watermarking method filters out words with significant semantic impact and only selects words that remain semantically stable under slight modifications for watermark injection. Additionally, our method selects fewer watermarking words compared to WTGB, thereby minimizing the semantic impact.

5.4 Robustness Analysis

We consider watermarking algorithms as private algorithms, where the attacker is not able to know the location of the watermark embedded in the text, but tries to remove the watermark using two main types of black-box attacks, including word-level attacks and document-level attacks.

Word-level Attack. In real-world attack scenarios, word-level attacks on watermark text involve deleting or substituting words in the generated watermark text, achieving the removal of the watermark by altering the text content. During the attack, we assign a word attack probability (P_{word}) for each sentence, indicating the likelihood of deleting or replacing each word in the sentence. To evaluate the robustness of our watermarking method against word-level attacks, we compute the watermark strength and detection accuracy of the text after partially deleting or replacing words in the sentence with different word attack probabilities ($P_{word} = \{2.5\%, 5\%, 10\%, 20\%, 30\%, 40\%, 50\%\}$). We use the "bert-base-cased" model to implement a lexical substitution attack. Additionally, we perform deletion or replacement operations on watermark text generated using WTGB with the same attack probabilities to ensure fair comparison.

As shown in Figure 4 and Figure 5, the AUC values and detection F1 scores of the watermarked text decrease as P_{word} increases, becoming particularly noticeable when $P_{word} > 30\%$. Both word deletion and word substitution attacks compromise the integrity of the watermark, with replacement attacks having a more severe impact. This is because, for our watermarking method, word substitution not only significantly affects keyword extraction but also drastically alters the structure of the text's dependency parse tree. Consequently, the watermark detector fails to correctly locate the watermark embedding positions, causing the detection results to trend toward 0.5 (equal probabilities of bit-0 and bit-1). For the WTGB method, although word replacements do not affect the detector's ability to locate watermarked words, the method's use of a significantly larger number of watermark embedding positions during the embedding stage means that more watermarked words are attacked even at low attack probabilities, thus affecting watermark detectability.

Document-level Attack. Compared to word-level attacks, document-level attack has a more severe impact on watermark text because they involve not only changes to individual words but also extensive modifications to the document's content and sentence structure. How to defend against such attacks is a challenging problem for watermarking methods. Typical document-level attack behaviors include using third-party language models such as ChatGPT and other commercial language models to polish watermark text, a practice commonly referred to as a Polishing Attack. Additionally,

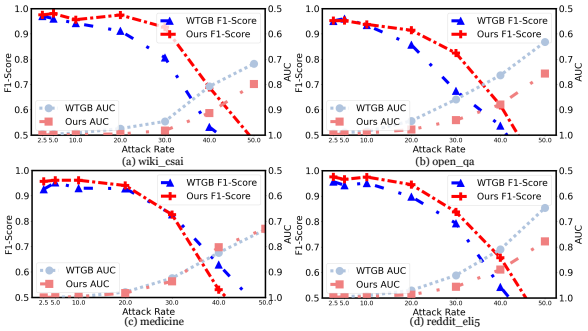

Figure 5: Robustness analysis of the watermark under word substitution attacks.

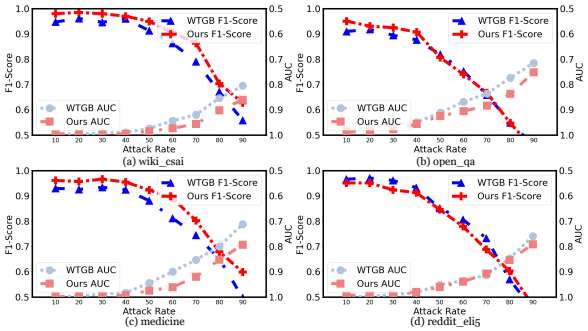

Figure 6: Robustness analysis of the watermark under polishing attacks. The x-axis represents the Polish attack probability. The y-axis displays the F1-score(f1) and AUC value for both our method and WTGB on four datasets. Higher scores indicate better performance.

there's the Re-translation Attack, where watermark text is translated back and forth between multiple languages.

For the Polishing Attack, we assume that the attacker would utilize top-tier LLM services to refine the watermark text. We employ the 'GPT-3.5-Turbo' interface provided by OpenAI to conduct such attacks, with the prompt for sentence refinement being: "*Please polish the input text without changing its meaning and structure. The input text is: [watermarked text].*". As for the Re-translation Attack, we use the DeepL translator [5] to execute the attack, first translating the watermark text into Chinese, then translating the Chinese text back into English.

Similar to word-level attacks, we assign an attack probability P_{doc} a to each sample, representing the likelihood of a polishing or re-translation attack for each sentence in the sample. Unlike word-level attacks, a higher document attack probability P_{doc} does not necessarily result in a significant decrease in semantic quality. Therefore, we execute attacks over a wider range ($P_{doc} = [10\%, 90\%]$).

For the Polishing Attack, as illustrated in Figure 6, the F1 scores and AUC values decrease progressively with the increase in the number of polished sentences per sample. This indicates that as more content is modified, the watermark is gradually erased. Compared to the WTGB method, our approach maintains higher watermark strength across three datasets. This is because our method selects watermark embedding positions that are relatively robust against attacks, a notable advantage for document-level attacks. In the case of word-level attacks, each word has an equal probability of being deleted. However, for document-level attacks, certain words exhibit robustness to modifications.

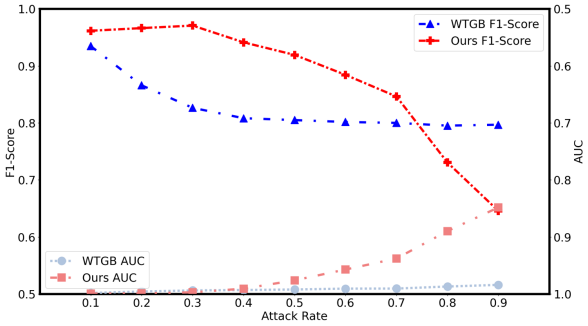

Figure 7: Robustness analysis of the watermark under re-translation attacks on medicine dataset.

We performed the Re-translation Attack on the watermarked text generated from the medicine dataset. As shown in Figure 7, similar to the Polishing Attack, as the number of sentences subjected to Re-translation increases, the watermark information in the text is gradually erased.

Considering both types of document-level Attacks, when half of the sentences in each sample are attacked ($0.5 \geq P_{doc}$), the F1 score begins to change steeply. By the time P_{doc} reaches 0.9, the F1 score is close to 0.5, indicating that almost all watermark information in the text has been erased. This occurs because our watermarking method heavily relies on sentence structure; when significant changes in sentence structure occur, the embedding positions of the watermark information are altered. Consequently, the bit values of the tokens at the detection positions tend to follow a 0-1 distribution.

6 Conclusion

In this study, we presented a novel watermarking method for text generated by black-box language models. Our approach focuses on embedding wa-

[5] https://www.deepl.com/en/translator

termarks by selecting specific tokens that are robust to minor text modifications, ensuring that the semantic integrity of the text is maintained. Our approach provides a robust and accurate solution for embedding watermarks in text generated by black-box language models. It offers improved detection accuracy and semantic integrity while being resilient to various types of attacks.

7 Limitations

Semantic Integrity and Naturalness: While our proposed method aims to preserve the original semantics through paraphrase-based lexical substitution, there might still be instances where subtle shifts in meaning or unnatural phrasing occur. These changes, though minor, could affect the readability and perceived authenticity of the watermarked text, potentially making it detectable as artificially manipulated text by discerning readers or advanced detection algorithms.

Scalability and Adaptability: Our method, which relies on selecting semantically and syntactically fundamental components for watermark embedding, might face challenges when scaling to different languages or highly diverse text domains. The specific nuances and structures of various languages and text types could require significant adaptation of the watermarking strategy, potentially limiting the method's applicability across all use cases and necessitating extensive retraining and fine-tuning for optimal performance.

Acknowledgement

This research is partially supported by the National Natural Science Foundation of China (62076217), the National Language Commission (ZDI145-71), the Blue Project of Jiangsu and Yangzhou University, and the Top-level Talents Support Program of Yangzhou University.

References

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. 2018. Yake! collection-independent automatic keyword extractor. In *Advances in Information Retrieval: 40th European Conference on IR Research, ECIR 2018, Grenoble, France, March 26-29, 2018, Proceedings 40*, pages 806–810. Springer.

Yapei Chang, Kalpesh Krishna, Amir Houmansadr, John Wieting, and Mohit Iyyer. 2024. Postmark: A robust blackbox watermark for large language models. *Preprint*, arXiv:2406.14517.

N Editorials. 2023. Tools such as chatgpt threaten transparent science; here are our ground rules for their use. *Nature*, 613(7945):612.

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

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*, 7(1):411–420.

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*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

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*, pages 17061–17084. PMLR.

Michael Liebrenz, Roman Schleifer, Anna Buadze, Dinesh Bhugra, and Alexander Smith. 2023. Generating scholarly content with chatgpt: ethical challenges for medical publishing. *The lancet digital health*, 5(3):e105–e106.

Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip S. Yu. 2024. A survey of text watermarking in the era of large language models. *Preprint*, arXiv:2312.07913.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *International Conference on Machine Learning*, pages 24950–24962. PMLR.

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*.

Jipeng Qiang, Yang Li, Chaowei Zhang, Yun Li, Yi Zhu, Yunhao Yuan, and Xindong Wu. 2023a. Chinese idiom paraphrasing. *Transactions of the Association for Computational Linguistics*, 11:740–754.

Jipeng Qiang, Kang Liu, Ying Li, Yun Li, Yi Zhu, Yun-Hao Yuan, Xiaocheng Hu, and Xiaoye Ouyang. 2023b. Chinese lexical substitution: Dataset and method. In *Proceedings of the 2023 Conference on*

Empirical Methods in Natural Language Processing, pages 29–42.

Jipeng Qiang, Kang Liu, Yun Li, Yunhao Yuan, and Yi Zhu. 2023c. Parals: Lexical substitution via pretrained paraphraser. In *ACL*.

Jipeng Qiang, Shiyu Zhu, Yun Li, Yi Zhu, Yunhao Yuan, and Xindong Wu. 2023d. Natural language watermarking via paraphraser-based lexical substitution. *Artif. Intell.*, 317:103859.

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*.

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*, pages 164–174.

Junchao Wu, Shu Yang, Runzhe Zhan, Yulin Yuan, Derek F Wong, and Lidia S Chao. 2023. A survey on llm-gernerated text detection: Necessity, methods, and future directions. *arXiv preprint arXiv:2310.14724*.

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*.

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)*, pages 2092–2115.

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

Shiyu Zhu, Yun Li, Xiaoye Ouyang, Xiaocheng Hu, and Jipeng Qiang. 2023. Safeguarding text generation api's intellectual property through meaning-preserving lexical watermarks. *Frontiers Comput. Sci.*, 17(6):176352.

A Time overhead

One concern raised regarding the post-hoc watermarking method is its computational complexity, which might result in significantly slower performance compared to white-box approaches. We conducted an additional experiment to measure the time overhead of our watermarking method. Specifically, we applied our approach to all four datasets and recorded the computational time required for watermark embedding and detection. The results of this analysis provide insights into the practicality of our method and confirm its feasibility for real-world applications. Details of the time overhead experiment are presented below. Table 2 shows the average number of tags per piece of data in each dataset, the total time taken in seconds to apply the watermark, and the token processing rate in tokens per second.

Dataset	Tokens	Time(s)	Tokens/sec
wiki_csai	178	70.96	2.50
open_qa	115	40.31	2.85
medicine	208	76.30	2.72
reddit_eeil5	161	62.05	2.59

Table 2: Performance of time overhead on 4 datasets

B More Results

Our method modifies fewer words while achieving a higher F1-score and demonstrating greater resilience against similar modification attacks compared to baseline (WTGB). To address this, we provided supplementary experimental data to support our claims.

Regarding the word deletion attack, as shown in Table 3,4,5,6, our method exhibits only a slight difference in F1 score compared to the baseline. This minor difference is attributable to the destructive nature of deletion attacks, which severely disrupt the bit values of watermarked words. Additionally, word deletions damage the syntactic structure of the sentence, which adversely impacts the performance of our method. However, despite these challenges, our approach consistently outperforms the baseline across all three datasets.

In word substitution attack scenarios, the F1 score of the proposed method is significantly higher than that of the baseline. As detailed in Table 7,9,10,11, when the attack probability increases, the baseline method experiences a substantial rise in false positives, resulting in notable fluctuations in Precision. The relationship between F1 score, Precision, and Recall offers further insight into this phenomenon.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)$$

Where Recall $= \frac{\text{TP}}{\text{TP+FN}}$, Precision $= \frac{\text{TP}}{\text{TP+FP}}$. Experimental results show that Recall remains relatively stable between the proposed method and the

Attack Rate	Method	TN	FP	FN	TP	Accuracy	F1-Score	AUC
0.025	WTGB	92	8	0	100	0.960	0.9615	0.99535
	Ours	93	7	0	100	0.965	0.9662	0.99800
0.05	WTGB	93	7	2	98	0.955	0.9561	0.99255
	Ours	93	7	3	97	0.950	0.9510	0.99230
0.1	WTGB	93	7	4	96	0.945	0.9458	0.98710
	Ours	94	6	5	95	0.945	0.9453	0.98705
0.2	WTGB	92	8	9	91	0.915	0.9146	0.97395
	Ours	92	8	5	95	0.935	0.9360	0.98330
0.3	WTGB	92	8	17	83	0.875	0.8691	0.95635
	Ours	96	4	8	92	0.940	0.9388	0.97965
0.4	WTGB	94	6	40	60	0.770	0.7229	0.89980
	Ours	97	3	20	80	0.885	0.8743	0.96485
0.5	WTGB	95	5	47	53	0.740	0.6709	0.88025
	Ours	94	6	51	49	0.715	0.6323	0.88540

Table 3: Deletion Attack on open_qa

Attack Rate	Method	TN	FP	FN	TP	Accuracy	F1-Score	AUC
0.025	WTGB	90	10	0	100	0.950	0.9524	0.99870
	Ours	93	7	1	99	0.960	0.9612	0.99515
0.05	WTGB	90	10	0	100	0.950	0.9524	0.99870
	Ours	93	7	1	99	0.960	0.9612	0.99605
0.1	WTGB	92	8	1	99	0.955	0.9565	0.99770
	Ours	95	5	3	97	0.960	0.9604	0.99410
0.2	WTGB	95	5	4	96	0.955	0.9552	0.99200
	Ours	94	6	5	95	0.945	0.9453	0.98385
0.3	WTGB	94	6	4	96	0.950	0.9505	0.98665
	Ours	96	4	15	85	0.905	0.8995	0.97595
0.4	WTGB	88	12	15	85	0.865	0.8629	0.93465
	Ours	95	5	31	69	0.820	0.7931	0.93840
0.5	WTGB	91	9	35	65	0.780	0.7471	0.90020
	Ours	92	8	51	49	0.705	0.6242	0.84540

Table 4: Deletion Attack on reddit_eli5

Attack Rate	Method	TN	FP	FN	TP	Accuracy	F1-Score	AUC
0.025	WTGB	94	6	2	98	0.960	0.9608	0.99755
	Ours	96	4	0	100	0.980	0.9804	0.99970
0.05	WTGB	94	6	1	99	0.965	0.9659	0.99745
	Ours	94	6	1	99	0.965	0.9659	0.99860
0.1	WTGB	92	8	1	99	0.955	0.9565	0.99685
	Ours	94	6	1	99	0.965	0.9659	0.99380
0.2	WTGB	92	8	4	96	0.940	0.9412	0.99170
	Ours	97	3	1	99	0.980	0.9802	0.99505
0.3	WTGB	94	6	6	94	0.940	0.9400	0.98430
	Ours	98	2	3	97	0.975	0.9749	0.99330
0.4	WTGB	93	7	18	82	0.875	0.8677	0.97335
	Ours	97	3	22	78	0.875	0.8619	0.97230
0.5	WTGB	95	5	28	72	0.835	0.8136	0.95690
	Ours	96	4	34	66	0.810	0.7765	0.91135

Table 5: Deletion Attack on wiki_csai

Attack Rate	Method	TN	FP	FN	TP	Accuracy	F1-Score	AUC
0.025	WTGB	85	15	0	100	0.925	0.9302	0.99990
	Ours	90	10	0	100	0.950	0.9524	0.99970
0.05	WTGB	85	15	0	100	0.925	0.9302	0.99940
	Ours	91	9	0	100	0.955	0.9569	0.99915
0.1	WTGB	83	17	0	100	0.915	0.9217	0.99785
	Ours	90	10	0	100	0.950	0.9524	0.99760
0.2	WTGB	91	9	2	98	0.945	0.9469	0.99370
	Ours	88	12	2	98	0.930	0.9333	0.99440
0.3	WTGB	92	8	4	96	0.940	0.9412	0.98360
	Ours	97	3	6	94	0.955	0.9543	0.98350
0.4	WTGB	93	7	17	83	0.880	0.8737	0.95595
	Ours	98	2	20	80	0.890	0.8791	0.95885
0.5	WTGB	93	7	29	71	0.820	0.7978	0.91600
	Ours	96	4	39	61	0.785	0.7394	0.91660

Table 6: Deletion Attack on medicine

Attack Rate	Method	TN	FP	FN	TP	Accuracy	F1-Score	AUC
0.025	WTGB	92	8	2	98	0.950	0.9515	0.9939
	Ours	90	10	0	100	0.950	0.9524	0.9975
0.05	WTGB	94	6	2	98	0.960	0.9608	0.9905
	Ours	90	10	0	100	0.950	0.9524	0.9973
0.1	WTGB	94	6	7	93	0.935	0.9347	0.9857
	Ours	89	11	2	98	0.935	0.9378	0.9923
0.2	WTGB	92	8	19	81	0.865	0.8571	0.9444
	Ours	92	8	9	91	0.915	0.9146	0.9788
0.3	WTGB	92	8	45	55	0.735	0.6748	0.8594
	Ours	93	7	25	75	0.840	0.8242	0.9415
0.4	WTGB	91	9	60	40	0.655	0.5369	0.7636
	Ours	97	3	54	46	0.715	0.6174	0.8794
0.5	WTGB	90	10	82	18	0.540	0.2812	0.6318
	Ours	96	4	80	20	0.580	0.3226	0.7568

Table 7: Substitution Attack on open_qa

baseline under various attack probabilities (± 0.07). To model this behavior, Precision is denoted as x and Recall as a constant a (where $0 < a < 1$), resulting in the function:

$$f(x) = 2 \times \frac{a \times x}{x + a} \tag{4}$$

Where $f(x)$ is monotonically increasing for $x > 0$, a reduction in Precision under higher attack probabilities directly causes a decline in the F1 score.

C Other metrics

Following Chang et al. (2024), we calculated the TPR at 1% FPR for both the baseline (WTGB) and our approach.

Dataset	**WTGB**	**Ours**
wiki_csai	98%	99%
open_qa	88%	97%
medicine	100%	100%
reddit_eeil5	97%	98%

Table 8: Comparison of WTGB and Ours across 4 datasets

Attack Rate	Method	TN	FP	FN	TP	Accuracy	F1-Score	AUC
0.025	WTGB	91	9	0	100	0.955	0.9569	0.9982
	Ours	96	4	1	99	0.975	0.9754	0.9965
0.05	WTGB	89	11	1	99	0.940	0.9429	0.9951
	Ours	94	6	1	99	0.965	0.9659	0.9977
0.1	WTGB	92	8	2	98	0.950	0.9515	0.9917
	Ours	97	3	2	98	0.975	0.9751	0.9950
0.2	WTGB	92	8	12	88	0.900	0.898	0.9704
	Ours	94	6	5	95	0.945	0.9453	0.9877
0.3	WTGB	92	8	29	71	0.815	0.7933	0.9116
	Ours	96	4	25	75	0.855	0.838	0.9566
0.4	WTGB	90	10	59	41	0.655	0.543	0.8095
	Ours	98	2	50	50	0.740	0.6579	0.8888
0.5	WTGB	88	12	78	22	0.550	0.3284	0.6462
	Ours	98	2	76	24	0.610	0.381	0.7785

Table 9: Substitution Attack on reddit_eli5

Attack Rate	Method	TN	FP	FN	TP	Accuracy	F1-Score	AUC
0.025	WTGB	95	5	1	99	0.970	0.9706	0.9962
	Ours	95	5	0	100	0.975	0.9756	0.9997
0.05	WTGB	94	6	2	98	0.960	0.9608	0.9964
	Ours	96	4	0	100	0.980	0.9804	0.9997
0.1	WTGB	90	10	2	98	0.940	0.9423	0.9924
	Ours	92	8	1	99	0.955	0.9565	0.9981
0.2	WTGB	89	11	7	93	0.910	0.9118	0.9748
	Ours	98	2	3	97	0.975	0.9749	0.9959
0.3	WTGB	92	8	27	73	0.825	0.8066	0.9464
	Ours	98	2	12	88	0.930	0.9263	0.9825
0.4	WTGB	95	5	62	38	0.665	0.5315	0.8077
	Ours	99	1	47	53	0.760	0.6883	0.9130
0.5	WTGB	95	5	74	26	0.605	0.3969	0.7180
	Ours	98	2	68	32	0.650	0.4776	0.7984

Table 10: Substitution Attack on wiki_csai

Attack Rate	Method	TN	FP	FN	TP	Accuracy	F1-Score	AUC
0.025	WTGB	84	16	0	100	0.920	0.9259	1.0000
	Ours	91	9	0	100	0.955	0.9569	1.0000
0.05	WTGB	90	10	0	100	0.950	0.9524	0.9995
	Ours	92	8	0	100	0.960	0.9615	0.9996
0.1	WTGB	85	15	0	100	0.925	0.9302	0.9973
	Ours	92	8	0	100	0.960	0.9615	0.9979
0.2	WTGB	87	13	2	98	0.925	0.9289	0.9793
	Ours	93	7	5	95	0.940	0.9406	0.9832
0.3	WTGB	88	12	21	79	0.835	0.8272	0.9244
	Ours	92	8	24	76	0.840	0.8261	0.9369
0.4	WTGB	91	9	50	50	0.705	0.6289	0.8243
	Ours	89	11	60	40	0.645	0.5298	0.8024
0.5	WTGB	95	5	71	29	0.620	0.4328	0.7326
	Ours	91	9	77	23	0.570	0.3485	0.7298

Table 11: Substitution Attack on medicine