Token-Specific Watermarking with Enhanced Detectability and Semantic Coherence for Large Language Models

Mingjia Huo [*1] **Sai Ashish Somayajula** [*1] **Youwei Liang** [1] **Ruisi Zhang** [1] **Farinaz Koushanfar** [1] **Pengtao Xie** [1]

Abstract

Large language models generate high-quality responses with potential misinformation, underscoring the need for regulation by distinguishing AI-generated and human-written texts. Watermarking is pivotal in this context, which involves embedding hidden markers in texts during the LLM inference phase, which is imperceptible to humans. Achieving both the detectability of inserted watermarks and the semantic quality of generated texts is challenging. While current watermarking algorithms have made promising progress in this direction, there remains significant scope for improvement. To address these challenges, we introduce a novel multi-objective optimization (MOO) approach for watermarking that utilizes lightweight networks to generate token-specific watermarking logits and splitting ratios. By leveraging MOO to optimize for both detection and semantic objective functions, our method simultaneously achieves detectability and semantic integrity. Experimental results show that our method outperforms current watermarking techniques in enhancing the detectability of texts generated by LLMs while maintaining their semantic coherence. Our code is available at https://github.com/mignonjia/TS_watermark.

1. Introduction

Large Language Models (LLMs), particularly ChatGPT, have significantly revolutionized the field of artificial intelligence (AI), bringing forth unparalleled advancements and applications with human-like capabilities (Schulman et al., 2022). However, this rapid evolution has been accompanied by ethical challenges. Prominent among these are concerns such as their potential use in election manipulation campaigns (Alvarez et al., 2023; Wu et al., 2023b), the creation of fake news and misleading web content (Augenstein et al., 2023), and aiding academic dishonesty by facilitating cheating on homework (Team, 2023). In light of these issues, the detection of text generated by LLMs emerges as a critical task, underpinning the broader goals of AI ethics and safety.

Various classification-based approaches have been proposed to determine whether a text is generated by humans or LLMs (OpenAI, 2023; Guo et al., 2023b; Brown et al., 1992). However, as LLM-generated texts increasingly match the quality of human-generated ones, these methods are losing their robustness. For instance, OpenAI released its own AI classifier (OpenAI, 2023) in early 2023, but it was later withdrawn due to its low accuracy. Recently, watermarking techniques in LLMs have gained popularity (Abdelnabi & Fritz, 2021; He et al., 2022a; Zhang et al., 2023; Kirchenbauer et al., 2023a; Kuditipudi et al., 2023; Aaronson, 2023; Liu et al., 2023; Ren et al., 2023; Lee et al., 2023; Wang et al., 2023; Wouters, 2023). These techniques embed hidden patterns in LLM-generated texts that, while imperceptible to humans, can be detected by algorithms. Kirchenbauer et al. (2023a) propose an effective watermarking method, referred to as KGW in this paper. To generate a new token with a watermark, the hash of the preceding token is used as a random seed to divide the vocabulary into a green list and a red list with a fixed splitting ratio (i.e., the percentage of tokens in the green list). Then a constant value, known as the watermark logit, is added to the LLM-produced logits on the green list tokens. This adjustment increases the probability of selecting the 'green' tokens as the new token. To identify the presence of a watermark, a statistical test is carried out. This involves counting the number of green list tokens in the generated text. A higher incidence of green tokens suggests that the text is likely to contain a watermark.

The design of KGW emphasizes easy detection of watermarked texts. However, this approach often compromises the semantic coherence of the texts, as highlighted in Lee et al. (2023). One primary cause of this issue is that KGW uses a constant splitting ratio and watermark logit across all tokens, without taking into account the context and se-

[*]Equal contribution [1]Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093, USA. Correspondence to: Pengtao Xie <p1xie@ucsd.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

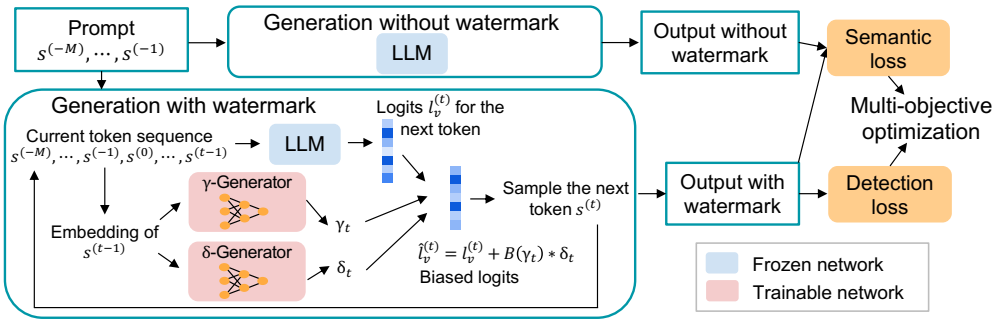


Figure 1: The training procedure is as follows: During the LLM text generation, we utilize the γ-generator and δ-generator to modify the probability of each token before sampling the next one. The parameters of these networks are learned through optimization of the detection loss (Eq. 4) and semantic loss (Eq. 5) within a multi-objective optimization framework.

mantics of the specific token being generated. For instance, given the prefix "The sun rises in the", it is highly probable that the next token to be generated should be "east". But KGW's watermarking mechanism might not choose "east" if the splitting ratio is low and the watermark logit is high. A lower splitting ratio results in a smaller number of words being added to the green list, thus significantly reducing the chances of including the word "east" in it. On the other hand, a higher watermark logit raises the probability of choosing a token from this list. Consequently, it becomes highly unlikely for "east" to be selected as the next token for generation, which will significantly impact the semantic coherence of the text. To mitigate this issue, adjusting the splitting ratio and watermark logit is necessary, perhaps by increasing the splitting ratio or decreasing the watermark logit. Unfortunately, KGW lacks such an adaptive mechanism. Conversely, if KGW utilizes a low, constant splitting ratio and a small, uniform watermark logit for every token to maintain semantic integrity, this approach may hinder the watermark's detectability. Hence, it is crucial to adaptively assign token-specific values for these hyperparameters, simultaneously ensuring high detectability while maintaining semantic integrity. Some works have been proposed to improve the semantics of watermarked texts. Lee et al. (2023) proposes a selective watermarking strategy to preserve semantics by adding watermarks only to high-entropy tokens while preserving the original logits for low-entropy tokens. Kuditipudi et al. (2023) reduce the semantics distortion by ensuring an unbiased distribution of tokens before and after watermarking, using an exponential minimum sampling strategy which can positively influence semantics. However, these works face challenges in enhancing both detectability and semantic coherence at the same time: improving one aspect frequently compromises the other.

To address the limitations of current methods, we introduce a novel method (Figure 1) that simultaneously achieves two primary goals: preserving the semantic integrity of generated texts and ensuring the effectiveness of watermark detection. Our method accomplishes the goals by dynamically adjusting the splitting ratio and watermark logit, controlled by two trainable light-weight networks, for each token during its generation. These networks process the representation of the previous token to determine the optimal splitting ratio and the appropriate watermark logit for the next token, supervised by two loss functions: 1) Watermark detectability via a one-sided z-test (Kirchenbauer et al., 2023a), which quantifies the presence of green tokens in the generated text. Since this metric is inherently non-differentiable, we introduce a differentiable surrogate that allows for direct optimization through gradient-based techniques during training. 2) Semantic coherence of watermarked texts, for which we measure the cosine similarity between SimCSE (Gao et al., 2021) embeddings of watermarked and non-watermarked texts. We develop a multi-objective optimization framework that aims to achieve both objectives concurrently. Our method is geared towards identifying Pareto optimal solutions, where improving one objective does not detrimentally affect the other. This balanced approach ensures the effectiveness of watermarking while maintaining the semantic quality of the generated texts.

Our main contributions include:

- We introduce a novel watermarking method for LLMs that improves both detectability and semantic coherence in generated texts. Unlike earlier methods, which often face challenges in achieving these objectives simultaneously, our approach employs multi-objective optimization to achieve both of them concurrently.
- Our method employs two lightweight networks to dynamically determine token-specific splitting ratios and watermark logits, avoiding uniform values across all tokens.
- Our method has undergone comprehensive evaluation, showing superior performance in both detectability and semantic quality compared to leading baselines.

2. Related Works

2.1. Watermarking Methods

Watermarking methods for large language models can be divided into three categories: rule-based watermarking, neural watermarking and inference-time watermarking. Rule-based watermarking embeds watermarks using text transformations while ensuring that the overall semantic coherence is not disturbed. These transformations involve altering lexical properties (He et al., 2022a), manipulating linguistic features (He et al., 2022b; Yoo et al., 2023), or substituting synonyms (Munyer & Zhong, 2023; Yang et al., 2023). A significant limitation of rule-based methods is that they are vulnerable to attacks (e.g., replacing words with synonyms). Neural watermarking employs neural networks to embed watermarks into texts and decode them. Abdelnabi & Fritz (2021) leverage a Transformer (Vaswani et al., 2017) encoder to embed a binary watermark message, which is then extracted by a Transformer decoder. A Transformer-based discriminator is employed to preserve the original semantics of the text while applying watermarking. REMARK-LLM (Zhang et al., 2023) employs a message encoding network to embed an LLM-generated text and its signature into a watermarked version. It then utilizes a message decoder to extract the LLM signature from this watermarked text. Neural watermarking methods often involve complicated neural networks to insert watermarks, incurring high computational costs during text generation and watermark detection.

Inference-time watermarking methods insert statistical signals into model logits during inference to improve detectability. KGW (Kirchenbauer et al., 2023a) comes under this umbrella. However, text watermarked by this method suffers from reduced semantic coherence. A series of works have been proposed to address this issue. Wang et al. (2023) introduce a constraint stipulating that the perplexity of watermarked text must not exceed the perplexity of the original text, as produced by the same language model without watermarking, by more than a small margin. However, this strategy is developed through a series of approximations to ensure practical applicability. These approximations can lead to semantic disparities between the two texts. Chen et al. (2023) propose to split the vocabulary into a green list and red list in a semantically more balanced manner so that any token in the red list can be replaced by a token in the green list. However, their splitting method is based on LLAMA2 (Touvron et al., 2023), which is computationally expensive. Furthermore, Liu et al. (2023) introduce a semantically invariant watermarking approach, aimed to improve attack and security robustness. They employ Compositional-BERT (Chanchani & Huang, 2023) to extract semantic representations of preceding tokens and produce watermark logits from these representations. However, this method incurs high computation overhead during inference due to the utilization of Compositional-BERT. Wouters (2023) present a closed-form solution for optimizing watermarking logits to enhance semantic integrity, with a predetermined splitting ratio. However, this solution relies on assumptions that may not hold in real-world scenarios, such as all tokens following identical distribution, and the absence of distribution shifts post-watermarking. Additionally, their analysis does not extend to determining an optimal splitting ratio - a factor that is critical for balancing detectability and semantic preservation. To enhance semantic coherence of watermarked computer programs, SWEET (Lee et al., 2023) proposes to selectively insert watermarks into high-entropy tokens instead of every token. While this method is effective for watermarking code, it demonstrates weak detection capabilities in texts across general domains.

2.2. Multi-Objective Optimization

Multi-objective optimization (MOO) (Deb et al., 2016) addresses the challenge of simultaneously optimizing several objectives which often conflict with each other. The Multiple-gradient Descent Algorithm (MGDA) (Désidéri, 2012) is a notable gradient-based approach designed for solving MOO problems. MGDA aims to identify a single gradient direction theoretically capable of optimizing all objectives concurrently. If the gradients are not normalized, the direction of the optimization is expected to be mostly influenced by the gradients of small norms (Désidéri, 2012). To mitigate this, various normalization methods have been developed (Chen et al., 2018; Milojkovic et al., 2019), striving for a more equitable consideration of all involved gradients. In our work, we employ MGDA to identify Pareto optimal solutions between detectability and semantic coherence.

3. Preliminaries

Our work is built upon an inference-time watermarking strategy introduced in Kirchenbauer et al. (2023a). This watermarking technique is a two-stage process: first, embedding a watermark into the text during its generation, and second, identifying the presence of this watermark in the text. During the generation of token $s^{(t)}$, the hash of the preceding token $s^{(t-1)}$ serves as a random seed. This seed is used to divide the vocabulary $\mathcal{V}$ into a green list, which contains a fraction γ of the vocabulary, and a red list, containing the remaining $(1-\gamma)$ fraction of the vocabulary. The parameter γ, known as the splitting ratio, determines the size of the green list relative to the total vocabulary. Next, a constant watermark logit, denoted as δ, is added as a bias to the logits of green list tokens. The sampling of the next token is then based on these adjusted logits, softly prompting the use of green list tokens.

The process of detecting watermarked text does not require

access to the LLM that originally generated it. Knowing the specific hash function and random number generator utilized is sufficient to reproduce the red and green lists associated with each token. The detection process involves testing the null hypothesis H_0 that the text was generated without knowledge of the green-red list rule. This is assessed using a one-sided z-test, with a z-score $z = (|s|_G - \gamma T)/\sqrt{T\gamma(1-\gamma)}$ under H_0, where $|s|_G$ represents the count of green list tokens in the watermarked text and T denotes the length of the text. A watermark is considered successfully detected if this test leads to the rejection of the null hypothesis, which occurs when the calculated z-score exceeds a predetermined threshold.

4. Method

4.1. Overview

We present a novel watermarking method for LLMs, designed to optimize two key aspects: detectability and semantic coherence. Detectability is assessed using a one-sided z-test that calculates a z-score based on the count of green tokens in the text. Semantic coherence is evaluated by measuring the cosine similarity between the embeddings of watermarked and non-watermarked texts. These measures are controlled by two hyperparameters: the split ratio γ and watermark logit δ. Their values vary for different tokens to reflect tokens' unique characteristics.

To specify token-specific values of γ and δ, we employ two light-weight networks: a γ-generator G_γ and a δ-generator G_δ. The γ-generator processes the representation of a previous token $t-1$ to determine the split ratio γ_t for token t, and similarly, the δ-generator operates for the watermark logits. These networks are optimized through a specialized multi-objective optimization framework (Désidéri, 2012; Sener & Koltun, 2018), aiming to simultaneously enhance detectability and semantic coherence. Figure 1 shows an overview of our proposed method.

4.2. Network Design

In this section, we provide a detailed description of the γ-generator and δ-generator.

γ-Generator. The γ-generator is a lightweight multi-layer perceptron. It takes the embedding of the preceding token $s^{(t-1)}$ as input and generates a splitting ratio $\gamma_t \in (0,1)$ for token t. Then we define a Bernoulli distribution parameterized by γ_t, denoted as $B(\gamma_t)$, to split the vocabulary $\mathcal{V}$ into a token-specific green list and red list. Specifically, for each token v in $\mathcal{V}$, we draw a sample from this distribution, $y_v^{(t)} \sim B(\gamma_t)$, to determine whether the token belongs to the green list. The token v is assigned to the green list if the sampled value $y_v^{(t)}$ is 1, and to the red list if it is 0.

However, the sampling process from a Bernoulli distribution is non-differentiable, which prevents the gradient-based updating of the parameters in G_γ. To address this issue, we utilize the Gumbel-Softmax method for differentiable sampling (Jang et al., 2017). Specifically, for each token v in $\mathcal{V}$, we estimate the probability that it belongs to the green list, denoted as $\hat{y}_v^{(t)}$, using the following formula:

$$\hat{y}_v^{(t)} = \frac{\exp\left(\frac{\log(\gamma_t)+g_0}{\tau}\right)}{\exp\left(\frac{\log(\gamma_t)+g_0}{\tau}\right) + \exp\left(\frac{\log(1-\gamma_t)+g_1}{\tau}\right)}. \quad (1)$$

Here, g_0 and g_1 are i.i.d samples from Gumbel$(0,1)$[1]. Here τ serves as a temperature parameter. As τ approaches 0, the distribution becomes increasingly sharp, which results in $\hat{y}_v^{(t)}$ more closely approximating $y_v^{(t)}$. We utilize $\hat{y}_v^{(t)}$, a differentiable approximation of $y_v^{(t)}$, to softly split the vocabulary into red and green token lists.

δ-Generator. Similarly to the γ-generator, the δ-generator is also a lightweight multi-layer perceptron, which takes the embedding of the preceding token $s^{(t-1)}$ as input and generates a watermark logit $\delta_t \in \mathbb{R}^+$ for token t to bias the green list tokens. Recall from Sec. 4.2, $\hat{y}_v^{(t)}$ is the probability that a token, $v \in \mathcal{V}$, belongs to the green list. The watermark logit δ_t is used to bias the model logit $l_v^{(t)}$ for token v as follows:

$$\hat{l}_v^{(t)} = l_v^{(t)} + \hat{y}_v^{(t)} \cdot \delta_t. \quad (2)$$

This formulation modifies the logit of token v by adding an appropriate amount of δ_t, based on the likelihood of the token being in the green list. These adjusted logits, $\{\hat{l}_v^{(t)} | v \in \mathcal{V}\}$, are transformed into a probability vector using SoftMax, which is then used to sample the next token.

4.3. Training Objectives

Our goals are to ensure both strong watermark detection ability and high semantic coherence after adding the watermark. To achieve these goals, we leverage two training objectives: a detection loss measured by a differentiable approximation of z-score and a semantic loss measured by the cosine similarity of watermarked and unwatermarked sentence embeddings.

Detection Loss. As described in Sec. 3, KGW (Kirchenbauer et al., 2023a) uses a constant green list ratio γ, and applies a one-sided z-test for watermark detection, represented as $(|s|_G - \gamma T)/\sqrt{T\gamma(1-\gamma)}$. However, our method introduces a novel variation: γ is dynamically adapted for

[1]The Gumbel$(0,1)$ distribution can be sampled using inverse transform sampling by drawing $u \sim$ Uniform$(0,1)$ and computing $g = -\log(-\log(u))$.

each token based on the semantics of the preceding token. We need to modify the z-score calculation to account for this dynamic adaptation of γ. Thus, the mean and the standard deviation of the distribution under H_0 need to be estimated. The generated t^{th} token can be either a red list token or a green list token. Under hypothesis H_0, this categorization follows a Bernoulli distribution $B(\gamma_t)$ parameterized by γ_t. To elucidate this, we define a random variable $X_t \sim B(\gamma_t)$, where $X_t = 1$ signifies a green list token and $X_t = 0$ a red list token. The mean of X_t is thus γ_t. Consequently, the total count of green list tokens in the entire sequence of length T is represented by $\sum_{t=0}^{T-1} X_t$.

Theorem 4.1. *Consider T independent Bernoulli random variables $X_1, \ldots, X_T$, each with means $\mu_1, \ldots, \mu_T$, $0 < \mu_t < 1 \; \forall t \in 1, \ldots, T$. The sum of these variables, $\sum_{t=1}^{T} X_t$, follows a Poisson binomial distribution. When T is sufficiently large, this distribution can be approximated by a Gaussian distribution with mean: $\sum_{t=1}^{T} \mu_t$ and variance: $\sum_{t=1}^{T} \mu_t(1-\mu_t)$.*

Using Theorem 4.1, when T is sufficiently large, the z-score under the null hypothesis, H_0, can be approximated by the following expression:

$$z = \frac{|s|_G - \sum_{t=1}^{T} \gamma_t}{\sqrt{\sum_{t=1}^{T} \gamma_t (1-\gamma_t)}}. \tag{3}$$

Our goal is to enhance detectability by optimizing this z-score. However, in its current formulation, the z-score is not differentiable with respect to the parameters of the γ- and δ-generator, due to the term $|s|_G$. Therefore, we propose a relaxed formulation, by relaxing $|s|_G$, the number of green list tokens, as $\sum_{t=1}^{T} p_{gr}^{(t)}$. Given a red-green token list during the generation of the tth token, $p_{gr}^{(t)}$ represents the probability of selecting a green token, as determined by the modified logits in Eq. 2. This probability is computed as the summation of the probabilities of all tokens in the green list, where these individual probabilities are calculated using the Softmax function applied to the modified logits. The relaxed z-score is:

$$\hat{z} = \frac{\sum_{t=1}^{T} p_{gr}^{(t)} - \sum_{t=1}^{T} \gamma_t}{\sqrt{\sum_{t=1}^{T} \gamma_t (1-\gamma_t)}}. \tag{4}$$

It is differentiable with respect to the parameters of the γ and δ generators. The objective of our approach is to maximize $\hat{z}$, thereby increasing detectability. Consequently, the detectability loss L_D is defined as $L_D = -\hat{z}$, which we aim to minimize.

Semantic Loss. To maintain the semantic integrity of watermarked texts, we aim for a high degree of semantic resemblance to their original, non-watermarked counterparts generated by LLMs. To evaluate the semantic similarity between the two versions of the text, we compute the cosine similarity of their latent embeddings, which are generated by the RoBERTa-base model (Liu et al., 2019) from SimCSE (Gao et al., 2021), denoted by f_θ. The SimCSE approach pretrains the RoBERTa-base model to generate sentence embeddings under a contrastive learning framework (Hadsell et al., 2006) so that the cosine similarity of the embeddings can indicate their semantic similarity. Specifically, considering s and s_w as the non-watermarked and watermarked LLM-generated sentences, respectively, we improve the semantic integrity of s_w by minimizing the following semantic loss:

$$L_S = -\cos_{\text{sim}}(f_\theta(s), f_\theta(s_w)), \tag{5}$$

where $\cos_{\text{sim}}(\cdot,\cdot)$ represents the cosine similarity operation. The optimization variables are the weight parameters in the γ- and δ-generator networks since the watermarking of s_w is controlled by these networks. Since the RoBERTa-base model f_θ and the LLM that generates s_w share the same tokenizer, we directly use the embedding generated by the LLM as the input to f_θ, making the operation differential w.r.t. the γ- and δ-generator networks.

4.4. Multi-Objective Optimization

As explained earlier, the detection loss L_D and semantic loss L_S have a competing relationship: solely decreasing one of them often leads to the increase of the other. To ensure concurrent reduction of both losses, we formulate a multi-objective optimization problem:

$$\min_{G_\gamma, G_\delta} L_D(G_\gamma, G_\delta) \text{ and } \min_{G_\gamma, G_\delta} L_S(G_\gamma, G_\delta). \tag{6}$$

The goal of this formulation is to achieve Pareto optimality as defined below.

Definition 4.2. (Pareto Optimality)

(a). A solution (G_γ, G_δ) dominates a solution $(\overline{G_\gamma}, \overline{G_\delta})$ if

- $L_D(G_\gamma, G_\delta) \leq L_D(\overline{G_\gamma}, \overline{G_\delta})$
- $L_S(G_\gamma, G_\delta) \leq L_S(\overline{G_\gamma}, \overline{G_\delta})$

and at least one inequality is strict.

(b). A solution (G_γ, G_δ) is called Pareto optimal if there exists no other solution that dominates it.

We employ the multiple-gradient descent algorithm (MGDA) (Désidéri, 2012; Sener & Koltun, 2018), to solve the multi-objective optimization problem described in Eq.(6). Please refer to Appendix C for details.

4.5. Watermark Generation and Detection

In this section, we introduce how to generate and detect a watermark using the trained G_γ and G_δ. The text generation process at inference time is similar to the procedure used in KGW. The primary difference is that we utilize a dynamic γ_t and δ_t for each token, outputted by the trained G_γ and G_δ, respectively, with the inputs being the embedding of the preceding token. Detection is conducted through one-sided z-test, specifically by using the z-score as defined in Eq.(3). Given a generated text, G_γ is utilized to compute γ_t necessary for calculating the z-score. Compared with the original KGW method, the only additional requirement during watermark detection is the embedding matrix of the LLM, which is used to compute the embedding of the preceding token.

5. Experiments

In our experiments, we insert watermarks to two prominent LLMs: OPT-1.3B (Zhang et al., 2022), LLAMA2-7B, 13B and 70B (Touvron et al., 2023). The evaluation of watermarking is based on three aspects: the trade-off between detectability and semantic integrity, computational complexity, and robustness against attacks of the watermarks. We also perform an analysis of our learned δ and γ, and present an ablation study in Appendix K. Training details including the hyperparameter settings can be found in Appendix D.

5.1. Experimental Settings

Dataset and Prompt. Following Kirchenbauer et al. (2023a), we utilize texts from the news-like subset of the C4 dataset (Raffel et al., 2019) to insert watermarks. For each text from this dataset, the last 200 tokens are truncated and designated as the 'baseline completion' (i.e., human-written texts). The remaining tokens from the start of the text are considered the 'prompt'. Conditioned on this prompt, the LLM generates a token sequence equivalent in length to the baseline completion, incorporating watermarks within this generation. Following Kirchenbauer et al. (2023a), we filter the dataset to include texts ranging from 500 to 1000 tokens. This dataset is then divided randomly into three subsets: 6,400 texts for training, 500 for validation, and another 500 for testing.

Evaluation Metrics. The objective of a watermarking algorithm is to effectively identify texts generated by LLMs, i.e., achieving a high true positive rate (TPR), while not classifying texts created by humans as LLM-generated, i.e., with a low false positive rate (FPR). By raising the threshold for the z-score in detection processes, we can effectively reduce the FPR. We set this threshold to maintain FPRs at two different levels: 0% and 1%. This is achieved by identifying the threshold corresponding to the top 0% (or 1%) of the z-scores of all 'baseline completions' on the test set. Using these thresholds, we then evaluate the TPR for watermarked texts generated by LLMs. Additionally, we evaluate the semantic coherence between texts generated with and without watermarks by LLMs. We measure this by calculating the cosine similarity of their sentence embeddings, obtained using the SimCSE model (as detailed in Sec. 4.3). The metric assesses the impact of watermarking on the quality of generated text. For each FPR level, we plot a trade-off curve between TPR and semantic similarity to assess the performance of the watermarking algorithms.

We compare our methods against KGW (Kirchenbauer et al., 2023a), SWEET (Lee et al., 2023), MultiBit (Wang et al., 2023), SIR (Liu et al., 2023), and EXP-edit (Kuditipudi et al., 2023). Both SWEET and SIR rely on prompts during detection, which is impractical in many real-world scenarios. To ensure a fair comparison, we also include variants of these baselines that do not use prompts during detection, namely $\text{SWEET}_{\text{NoPrompt}}$ and $\text{SIR}_{\text{NoPrompt}}$. We plot trade-off curves of TPR and semantic similarity for the two settings, FPR=0% and FPR=1%. We vary δ within the appropriate range for each baseline to generate different pairs of TPR and semantic similarity. For KGW, SWEET, and $\text{SWEET}_{\text{NoPrompt}}$, we set $\gamma = 0.25$. For MultiBit, we use message lengths of 9 and 7 for which the FPR is 0% and 1%, respectively. Multinomial sampling with a temperature of 1.0 and Top-k of 50 is used for text generation. For our method, we generate different pairs of TPR and semantic similarity using different initializations of the γ- and δ-generator networks. Then we fit a curve to these pairs (i.e., points) for each method and plot the curves to form the Pareto frontier (Giagkiozis & Fleming, 2014), which represents the best trade-off curve of TPR and semantic similarity. For more details, please refer to Appendix D.

5.2. Results on Detectability and Semantic Coherence

Figure 2 and 3 show results on OPT-1.3B and LLAMA2 (7B, 13B and 70B), respectively, which highlight several key observations when comparing with the baselines.

Comparison with KGW (Kirchenbauer et al., 2023a): Our approach improves the Pareto frontier compared to the KGW baseline, attributable to two key factors. The first factor is that our method uniquely learns token-specific splitting ratios and watermark logits, which take into account the distinct context and semantics of each token. This is critical because the number of semantically appropriate tokens that can be chosen as the next token in the sequence can vary substantially at different time steps during text generation, depending on the context. In scenarios where this number is small (such as the example in Sec. 1), it is essential to lower the splitting ratio and watermark logit to reduce the watermark strength. This adjustment increases the likelihood

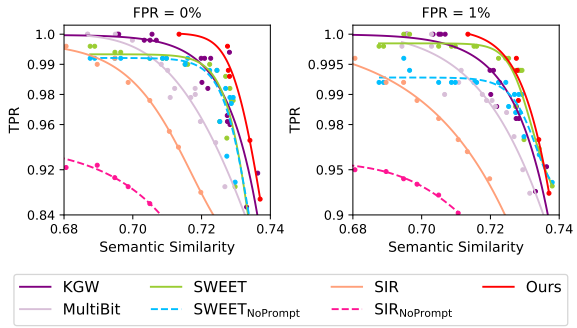


Figure 2: Comparison of the trade-off for semantic integrity and detectability of different methods applied to OPT-1.3B.

of selecting these valid choices as the next token, thereby maintaining semantics. Conversely, in situations where this number is large, increasing the splitting ratio and watermark logit to an *appropriate* level can improve detectability while posing minimal risk to semantic coherence. Our method has the adaptability that allows for token-specific adjustments in splitting ratios and watermark logits, while KGW employs uniform values across all tokens. The second factor distinguishing our approach is the incorporation of multi-objective optimization, which enables simultaneous maximization of detectability and semantic integrity. This is achieved by concurrently optimizing a differentiable detection loss and a semantic loss. In contrast, the KGW method cannot explicitly optimize for these two objectives together.

Comparison with SWEET (Lee et al., 2023): Our method achieves a better Pareto frontier than SWEET at 0% FPR while maintaining similar performance at 1% FPR. At 0% FPR, SWEET is notably less effective compared to our approach and KGW, and does not achieve 100% TPR. This may be due to its selective watermarking strategy, which targets only high-entropy words and leaves low-entropy words un-watermarked ($\delta = 0$). For instance, at $(\gamma, \delta) = (0.25, 3.0)$, an analysis of LLM-generated texts that SWEET failed to detect at 0% FPR shows that, on average, only 7 out of 200 tokens are high-entropy and eligible for SWEET watermarking. This limited number of watermarkable tokens reduces SWEET's detectability even with high δ (see Appendix G). In contrast, both our method and KGW achieve a 100% TPR in high δ regions. Furthermore, SWEET$_{\text{NoPrompt}}$ underperforms SWEET, indicating the method's dependence on prompts, which is impractical.

Comparison with MultiBit (Wang et al., 2023): Our method achieves a superior Pareto frontier compared to MultiBit. MultiBit embeds multi-bit information into LLM-generated texts, detecting a watermark if the decoded message matches the embedded one. However, embedding multi-bit information often reduces text quality. To mitigate this, Balance-Marking strategy is introduced to decrease the perplexity of watermarked texts. However, this method is developed through a series of approximations to ensure practical applicability, which might limit its effectiveness. In contrast, our method directly maximizes differentiable metrics of semantic coherence and detectability through multi-objective optimization, inherently improving them.

Table 1: Comparison of EXP-edit and Our Method

Method	TPR @ 0%	TPR @ 1%	SimCSE
EXP-edit	0.922	0.996	0.655
EXP-edit (Top-k=50)	0.968	0.996	0.677
Ours (Top-k=50)	**1.000**	**1.000**	**0.713**

Comparison with SIR (Liu et al., 2023): Our method achieves an improved Pareto frontier than SIR. SIR primarily aims to enhance attack and security robustness, lacking a direct approach to boost text quality. Conversely, our method employs multi-objective optimization to effectively enhance both text quality and watermark detectability simultaneously. Additionally, SIR$_{\text{NoPrompt}}$ significantly underperforms compared to SIR, indicating a strong dependence on prompts while detection. SIR is less robust than SWEET in the no-prompt scenario, as it exhibits a greater performance degradation without prompts compared to SWEET.

Comparison with EXP-edit (Kuditipudi et al., 2023): EXP-edit's generation process, which employs exponential minimum sampling, is pseudo-random and becomes deterministic with a watermark key, facilitating detection. Unlike ours and other KGW-based methods, which shift the distribution using δ, EXP-edit does not alter the output distribution of LLMs, making it indistinguishable. Since EXP-edit does not have a δ parameter to vary and plot the Pareto frontier, we present the results in Table 1. Our method outperforms EXP-edit in both detectability and SimCSE, demonstrating that learning token-specific parameters to watermark enables appropriately shifting the output distribution to enhance detectability without significantly affecting semantics. This offers more freedom to effectively embed watermark compared to EXP-edit, which lacks this capability (Piet et al., 2023).

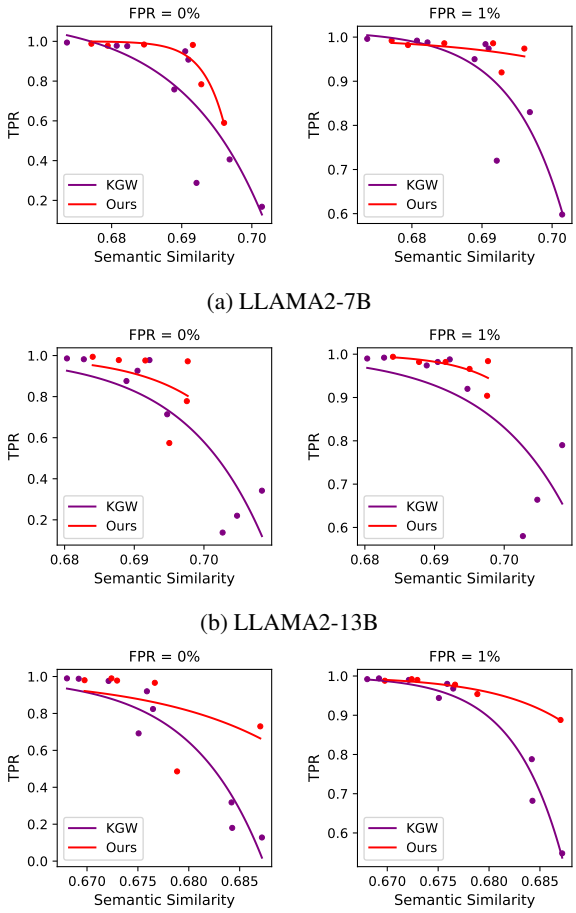
Since our decoding strategy uses the default Top-k value of 50, we modified the original EXP-edit generation process, which did not implement Top-k, to sample from the top 50 tokens, naming it EXP-edit (Top-k=50). We observe that our method still outperforms EXP-edit (Top-k=50) in terms of detectability and SimCSE. Moreover, indistinguishable methods like EXP-edit, which rely on pseudo-random sampling during generation, cannot easily extend to other decoding methods like beam search, which do not involve randomness. In contrast, KGW-based methods can be applied on top of any decoding method such as beam search (Kirchenbauer et al., 2023a). See Appendix H for further discussion.

Table 2: Generation and detection speed on OPT-1.3B for generating 200 tokens, measured in seconds. We also show memory utilization in Table 5.

Method	Generation (s)	Detection (s)
No Watermark	3.220	-
KGW	3.827	0.067
SWEET	4.030	0.127
EXP-edit	24.693	155.045
SIR	8.420	0.337
MultiBit	6.500	0.610
Ours	3.946	0.166

Computation Overhead: We evaluate the computational time of our method and the baselines for text generation and detection in Table 2. Our method achieves higher speeds than EXP-edit, SIR, and MultiBit, while achieving speeds comparable to KGW, SWEET, and No Watermarking.

Generalizability: Our method demonstrates good generalizability across LLMs of different sizes. As shown in Figure 3, our model (γ- and δ-generator networks), initially trained on OPT-1.3B, demonstrates a better Pareto frontier when applied to LLAMA2 7B, 13B and 70B. This adaptability is likely because our method learns the watermarking parameters that reflect the general nature of language itself (see Sec. 5.3 and Appendix I for discussions), not just on the specific details of the LLMs.



(a) LLAMA2-7B

(b) LLAMA2-13B

(c) LLAMA2-70B

Figure 3: Performance of Ours (trained on OPT-1.3B) and KGW when applied to LLAMA2 7B, 13B, and 70B.

5.3. Analysis of Learned δ and γ

In this section, we examine the values of γ and δ learned by our method for different tokens. For each part-of-speech (POS) category, we calculate the mean and standard deviation of γ and δ values generated based on *preceding* tokens that are tagged with this category. This analysis is conducted on ten texts, each containing 200 tokens that have been watermarked. The results are presented in Figure 4.

One observation is that when the preceding token is an adjective (ADJ) or a determiner (DET), γ and δ tend to be assigned lower values. This pattern is notable because ADJs and DETs typically precede nouns, as detailed in Appendix I. Reducing γ leads to the selection of fewer green tokens, while simultaneously lowering δ results in applying a smaller watermark logit to these chosen tokens. This combination effectively weakens the watermark, as it biases only a few tokens with relatively lower strength, thereby encouraging the selection of the token with the highest model logit. Applying a weaker watermark to tokens following ADJs and DETs promotes the selection of the next token with the highest model logit, which is most likely to be a noun. This approach thereby enhances both semantic coherence and syntactic consistency. Similarly, our method allocates increased values of γ and δ following punctuation (PUNCT) tokens. Given the minimal constraints on subsequent tokens after PUNCT, as detailed in Appendix I, our model potentially exploits this flexibility to embed a stronger watermark. It does so by putting more words into the green list (via assigning higher γ values) and enhancing the preference for these tokens through elevated δ values.

5.4. Robustness Against Attacks on Watermarks

Considering that watermarked text can be easily altered to remove the watermark, thereby making detection challenging, we evaluate the robustness of our method against two prevalent attacks: 1) the Paraphrase Attack (Krishna et al., 2023), where watermarked text is rephrased by another LLM, aiming to weaken the watermark while preserving the original semantics; 2) the Copy-Paste Attack (Kirchenbauer et al., 2023b), which inserts watermarked text into its corresponding human-generated prompts used for its creation; this attack weakens the detection of watermarks as it increases the number of red tokens in the sentence.

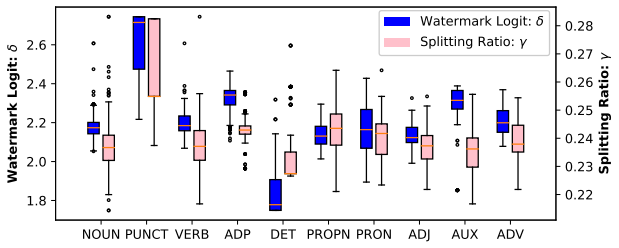


Figure 4: Distribution of watermark logit δ (left y-axis) and splitting ratio γ (right y-axis) across different part-of-speech categories of the *preceding* token.

Paraphrase Attack. We utilized the Dipper paraphrase model (Krishna et al., 2023) to perform paraphrase attacks. The model is finetuned from the T5-XXL (Raffel et al., 2020) model of 11B parameters, which is larger than the OPT-1.3B model we used for text generation. The larger model size results in improved generation capabilities (Brown et al., 2020), ultimately leading to a more potent attack. We set the paraphrase strength to the same level as recommended in the Dipper GitHub repository for watermark detection, specifically lex=40, div=100.

Figure 5 shows that our method is more robust against paraphrase attacks than KGW, as evidenced by its superior Pareto frontier. This can be attributed to the fact that paraphrasing tends to preserve local lexical structures, including punctuation patterns. As discussed in Sec. 5.3, our approach inserts strong watermarks around punctuations. These watermarks remain intact even after paraphrasing, ensuring the detectability of the watermark in the altered text. Such a mechanism is lacking in KGW.

Copy-Paste Attack. Our method is evaluated against two types of copy-paste attacks: Copy-Paste-1 and Copy-Paste-3. In Copy-Paste-1, the entire watermarked text is inserted at a random position within the prompt. In Copy-Paste-3, the watermarked text is split into three segments, each of which is then randomly inserted at different positions in the prompt. The detection process is conducted in a similar fashion as in Kirchenbauer et al. (2023b), using a sliding window to compute the maximum z-score across text subsequences. The window size is 200 for Copy-Paste-1 and 60 for Copy-Paste-3.

As shown in Figures 6, our method outperforms KGW in achieving a superior Pareto frontier across both scenarios. This improvement is largely due to the enhanced z-score, a direct result of our multi-objective optimization framework, which effectively increases the number of green tokens in the watermarked text. This ensures successful watermark detection even when human texts are inserted.

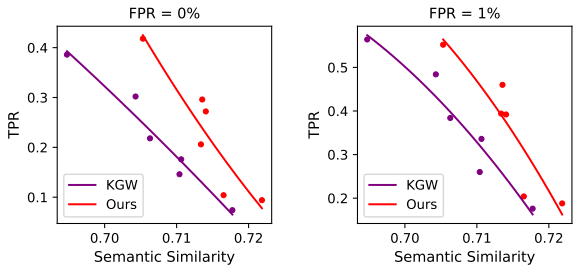


Figure 5: Comparison of our method with KGW under the Dipper paraphrase attack.

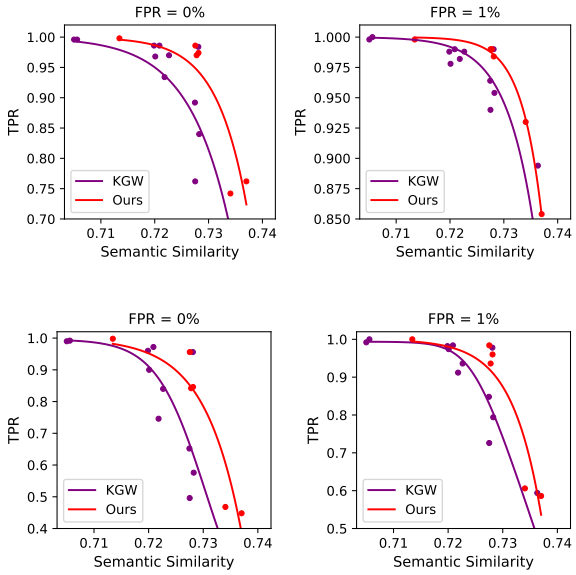


Figure 6: Comparison of our method with KGW under the Copy-Paste-1 (the two figures at the top) and Copy-Paste-3 attack (the two figures at the bottom).

6. Conclusion

In this work, we introduce a novel multi-objective optimization framework for watermarking LLMs during inference time. This method is designed to simultaneously optimize two light-weight networks, responsible for generating token-specific splitting ratios and watermark logits. The core objective of this approach is to minimize both detection loss and semantic loss, striving to find a Pareto optimal solution that enhances watermark detectability while preserving the semantic integrity of the generated text. Experiments demonstrate that our method consistently improves the Pareto frontier, surpassing previous techniques by offering improved watermark detectability and semantic integrity concurrently. Furthermore, our approach exhibits enhanced robustness against strong attacks, such as paraphrasing and copy-paste attacks, highlighting its practical effectiveness in safeguarding LLM outputs.

Impact Statement

Our watermarking method for LLMs carries significant societal implications. By improving the detection of machine-generated texts, our approach has the potential to prevent the usage of LLMs in harmful activities, such as election manipulation campaigns, the dissemination of fake news, and academic dishonesty. Furthermore, when used responsibly, watermarking algorithms contribute to the protection of intellectual property rights, benefiting companies and content creators, and reducing the risk of unauthorized use. However, we realize the ethical considerations associated with our work, as the misapplication of these tools could mistakenly label human-generated content as LLM-generated, potentially leading to accusations of plagiarism against innocent individuals.

References

Aaronson, S. 'reform' ai alignment with scott aaronson. https://axrp.net/episode/2023/04/11/episode-20-reform-ai-alignment-scott-aaronson.html, 2023.

Abdelnabi, S. and Fritz, M. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 121–140. IEEE, 2021.

Alvarez, R. M., Eberhardt, F., and Linegar, M. Generative ai and the future of elections, 2023.

Augenstein, I., Baldwin, T., Cha, M., Chakraborty, T., Ciampaglia, G. L., Corney, D., DiResta, R., Ferrara, E., Hale, S., Halevy, A., et al. Factuality challenges in the era of large language models. *arXiv preprint arXiv:2310.05189*, 2023.

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., Lai, J. C., and Mercer, R. L. An estimate of an upper bound for the entropy of english. *Computational Linguistics*, 18 (1):31–40, 1992.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chanchani, S. J. and Huang, R. Composition-contrastive learning for sentence embeddings. *arXiv preprint arXiv:2307.07380*, 2023.

Chen, L., Bian, Y., Deng, Y., Li, S., Wu, B., Zhao, P., and fai Wong, K. X-mark: Towards lossless watermarking through lexical redundancy, 2023.

Chen, Z., Badrinarayanan, V., Lee, C.-Y., and Rabinovich, A. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pp. 794–803. PMLR, 2018.

Cuzzocrea, A., Fadda, E., and Baldo, A. Lyapunov central limit theorem: Theoretical properties and applications in big-data-populated smart city settings. In *Proceedings of the 2021 5th International Conference on Cloud and Big Data Computing*, pp. 34–38, 2021.

Deb, K., Sindhya, K., and Hakanen, J. Multi-objective optimization. In *Decision sciences*, pp. 161–200. CRC Press, 2016.

Désidéri, J.-A. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318, 2012.

Fernandez, P., Chaffin, A., Tit, K., Chappelier, V., and Furon, T. Three bricks to consolidate watermarks for large language models. In *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6. IEEE, 2023.

Fröhling, L. and Zubiaga, A. Feature-based detection of automated language models: tackling gpt-2, gpt-3 and grover. *PeerJ Computer Science*, 7:e443, 2021.

Gambini, M., Fagni, T., Falchi, F., and Tesconi, M. On pushing deepfake tweet detection capabilities to the limits. In *Proceedings of the 14th ACM Web Science Conference 2022*, pp. 154–163, 2022.

Gao, T., Yao, X., and Chen, D. SimCSE: Simple contrastive learning of sentence embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Gehrmann, S., Strobelt, H., and Rush, A. M. Gltr: Statistical detection and visualization of generated text. *arXiv preprint arXiv:1906.04043*, 2019.

Giagkiozis, I. and Fleming, P. J. Pareto front estimation for decision making. *Evolutionary computation*, 22(4): 651–678, 2014.

Gottschalk, P. G. and Dunn, J. R. The five-parameter logistic: a characterization and comparison with the four-parameter logistic. *Analytical biochemistry*, 343(1):54–65, 2005.

Guo, B., Zhang, X., Wang, Z., Jiang, M., Nie, J., Ding, Y., Yue, J., and Wu, Y. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arxiv:2301.07597*, 2023a.

Guo, B., Zhang, X., Wang, Z., Jiang, M., Nie, J., Ding, Y., Yue, J., and Wu, Y. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arXiv:2301.07597*, 2023b.

Hadsell, R., Chopra, S., and LeCun, Y. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pp. 1735–1742. IEEE, 2006.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

He, X., Xu, Q., Lyu, L., Wu, F., and Wang, C. Protecting intellectual property of language generation apis with lexical watermark. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 10758–10766, 2022a.

He, X., Xu, Q., Zeng, Y., Lyu, L., Wu, F., Li, J., and Jia, R. Cater: Intellectual property protection on text generation apis via conditional watermarks. *Advances in Neural Information Processing Systems*, 35:5431–5445, 2022b.

Ippolito, D., Duckworth, D., Callison-Burch, C., and Eck, D. Automatic detection of generated text is easiest when humans are fooled. *arXiv preprint arXiv:1911.00650*, 2019.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kirchenbauer, J., Geiping, J., Wen, Y., Katz, J., Miers, I., and Goldstein, T. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023a.

Kirchenbauer, J., Geiping, J., Wen, Y., Shu, M., Saifullah, K., Kong, K., Fernando, K., Saha, A., Goldblum, M., and Goldstein, T. On the reliability of watermarks for large language models. *arXiv preprint arXiv:2306.04634*, 2023b.

Krishna, K., Song, Y., Karpinska, M., Wieting, J., and Iyyer, M. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *arXiv preprint arXiv:2303.13408*, 2023.

Kuditipudi, R., Thickstun, J., Hashimoto, T., and Liang, P. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*, 2023.

Lee, T., Hong, S., Ahn, J., Hong, I., Lee, H., Yun, S., Shin, J., and Kim, G. Who wrote this code? watermarking for code generation. *arXiv preprint arXiv:2305.15060*, 2023.

Li, L., Wang, P., Ren, K., Sun, T., and Qiu, X. Origin tracing and detecting of llms. *arXiv preprint arXiv:2304.14072*, 2023.

Liang, W., Yuksekgonul, M., Mao, Y., Wu, E., and Zou, J. Gpt detectors are biased against non-native english writers. *arXiv preprint arXiv:2304.02819*, 2023.

Liu, A., Pan, L., Hu, X., Meng, S., and Wen, L. A semantic invariant robust watermark for large language models. *arXiv preprint arXiv:2310.06356*, 2023.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, pp. 3. Atlanta, GA, 2013.

Milojkovic, N., Antognini, D., Bergamin, G., Faltings, B., and Musat, C. Multi-gradient descent for multi-objective recommender systems. *arXiv preprint arXiv:2001.00846*, 2019.

Munyer, T. and Zhong, X. Deeptextmark: Deep learning based text watermarking for detection of large language model generated text. *arXiv preprint arXiv:2305.05773*, 2023.

OpenAI. New ai classifier for indicating ai-written text. https://openai.com/blog/new-ai-classifier-for-indicating-ai-written-text, 2023.

Piet, J., Sitawarin, C., Fang, V., Mu, N., and Wagner, D. Mark my words: Analyzing and evaluating language model watermarks. *arXiv preprint arXiv:2312.00273*, 2023.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Ren, J., Xu, H., Liu, Y., Cui, Y., Wang, S., Yin, D., and Tang, J. A robust semantics-based watermark for large language model against paraphrasing. *arXiv preprint arXiv:2311.08721*, 2023.

Rodriguez, J., Hay, T., Gros, D., Shamsi, Z., and Srinivasan, R. Cross-domain detection of gpt-2-generated technical text. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1213–1233, 2022.

Schuhmann, C. essays-with-instructions. https://huggingface.co/datasets/ChristophSchuhmann/essays-with-instructions, 2023.

Schulman, J., Zoph, B., Kim, C., Hilton, J., Menick, J., Weng, J., Uribe, J. F. C., Fedus, L., Metz, L., Pokorny, M., et al. Chatgpt: Optimizing language models for dialogue. *OpenAI blog*, 2022.

Sener, O. and Koltun, V. Multi-task learning as multi-objective optimization. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 525–536. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7334-multi-task-learning-as-multi-objective-optimization.pdf.

Solaiman, I., Brundage, M., Clark, J., Askell, A., Herbert-Voss, A., Wu, J., Radford, A., Krueger, G., Kim, J. W., Kreps, S., et al. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*, 2019.

Team, O. Chatgpt cheating scandal shocks florida high school. https://opendatascience.com/chatgpt-cheating-scandal-shocks-florida-high-school/, 2023.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, L., Yang, W., Chen, D., Zhou, H., Lin, Y., Meng, F., Zhou, J., and Sun, X. Towards codable text watermarking for large language models. *arXiv preprint arXiv:2307.15992*, 2023.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Wolff, M. and Wolff, S. Attacking neural text detectors. *arXiv preprint arXiv:2002.11768*, 2020.

Wouters, B. Optimizing watermarks for large language models, 2023.

Wu, K., Pang, L., Shen, H., Cheng, X., and Chua, T.-S. Llmdet: A third party large language models generated text detection tool. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 2113–2133, 2023a.

Wu, P. Y., Tucker, J. A., Nagler, J., and Messing, S. Large language models can be used to estimate the ideologies of politicians in a zero-shot learning setting. *arXiv preprint arXiv:2303.12057*, 2023b.

Yang, X., Chen, K., Zhang, W., Liu, C., Qi, Y., Zhang, J., Fang, H., and Yu, N. Watermarking text generated by black-box language models. *arXiv preprint arXiv:2305.08883*, 2023.

Yoo, K., Ahn, W., Jang, J., and Kwak, N. 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)*, pp. 2092–2115, 2023.

Yu, X., Qi, Y., Chen, K., Chen, G., Yang, X., Zhu, P., Zhang, W., and Yu, N. Gpt paternity test: Gpt generated text detection with gpt genetic inheritance. *arXiv preprint arXiv:2305.12519*, 2023.

Zhang, R., Hussain, S. S., Neekhara, P., and Koushanfar, F. Remark-llm: A robust and efficient watermarking framework for generative large language models. *arXiv preprint arXiv:2310.12362*, 2023.

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

Zhong, W., Tang, D., Xu, Z., Wang, R., Duan, N., Zhou, M., Wang, J., and Yin, J. Neural deepfake detection with factual structure of text. *arXiv preprint arXiv:2010.07475*, 2020.

A. Post-Hoc Text Detection

Here is a supplement of related works, specifically focusing on the literature for post-hoc text detection. Post-hoc text detection aims to distinguish generated texts from human-authored texts by analyzing generated texts without access to the LLMs (i.e., treating them as black boxes). These methods primarily leverage features extracted using external language models or by training models to act as detectors. Gehrmann et al. (2019) proposes the use of statistical metrics such as perplexity, entropy, and n-gram frequency to detect LLM-generated texts, as those metrics for generated texts may be different from human-written texts. Given a target text S, Sniffer (Li et al., 2023) uses multiple accessible language models to compute a list of perplexity for the given text S, and train a linear classifier based on the perplexity features. and LLMDet (Wu et al., 2023a) rely on perplexity-based features for detection. In addition, there are supervised learning methods where classifiers are trained to specifically distinguish between texts generated by humans and LLMs. Fröhling & Zubiaga (2021) and Solaiman et al. (2019) utilized SVMs and regressions based on statistical features for LLM-generated text detection, while Rodriguez et al. (2022) and Zhong et al. (2020) employed neural networks for this purpose. Other works, such as those by Solaiman et al. (2019), Ippolito et al. (2019), Guo et al. (2023b), Yu et al. (2023), and the OpenAI Detector (OpenAI, 2023), involve fine-tuning a pre-trained RoBERTa (Liu et al., 2019) classifier for the detection of LLM-generated texts.

However, the diverse and evolving nature of LLM-generated texts presents challenges in developing robust post-hoc detection techniques. For instance, the detection strategies effective for GPT-2 may not be applicable to GPT-3, highlighting the evolving complexity of these models (Gambini et al., 2022). Additionally, these detection models are susceptible to adversarial attacks, which can deteriorate their performance (Wolff & Wolff, 2020). A significant challenge in post-hoc text detection is the minimal difference between LLM-generated and human-authored texts, often leading to human content being mislabeled as LLM-generated (Liang et al., 2023). To mitigate these issues, watermarking approaches have been proposed that embed statistical signals during text generation that help distinguish it from human generated texts. These methods reduce false positive rates and substantially improve detection capabilities.

B. Proof of Theorem 4.1

The proof of Theorem 4.1 is detailed in Cuzzocrea et al. (2021). The theorem is applicable under the condition that $0 < \mu_t < 1 \; \forall t \in 1, \ldots, T$; that is, $\mu_t \neq 0$ and $\mu_t \neq 1$. This assumption is valid in our context since γ_t is neither 0 nor 1 for any t. Specifically, $\gamma_t = 0$ would imply the absence of any tokens in the green list, indicating that watermarking has not been applied, whereas $\gamma_t = 1$ would suggest that every token is in the green list, rendering detection infeasible.

C. Multiple-Gradient Descent Algorithm

As explained in Sec. 4.4, we jointly optimize over the detection loss L_D and semantic loss L_S using multiple-gradient descent algorithm (MGDA) (Désidéri, 2012; Sener & Koltun, 2018), which is formulated as:

$$\min_{G_\gamma, G_\delta} L_D(G_\gamma, G_\delta) \text{ and } \min_{G_\gamma, G_\delta} L_S(G_\gamma, G_\delta).$$

In MGDA, the gradients of L_D and L_S with respect to (G_γ, G_δ) are firstly computed, which are denoted as $\boldsymbol{g}_D$ and $\boldsymbol{g}_S$, respectively. The resultant gradient vector, $\boldsymbol{g}$, is then estimated using $\boldsymbol{g}_D$ and $\boldsymbol{g}_S$ that directs the optimization towards Pareto optimal solutions. This is determined as the minimum norm point within the convex hull formed by $\boldsymbol{g}_D$ and $\boldsymbol{g}_S$, which is formulated as follows:

$$\lambda^* = \arg\min_{\lambda \in [0,1]} \|\lambda \boldsymbol{g}_D + (1-\lambda)\boldsymbol{g}_S\|_2 \tag{7}$$

Note that the closed form of λ^* can be obtained using a few simple operations following Sener & Koltun (2018).

$$\lambda^* = \begin{cases} 1, & \text{if } \boldsymbol{g}_D^T \boldsymbol{g}_S \geq \boldsymbol{g}_D^T \boldsymbol{g}_D \\ 0, & \text{if } \boldsymbol{g}_D^T \boldsymbol{g}_S \geq \boldsymbol{g}_S^T \boldsymbol{g}_S \\ \frac{(\boldsymbol{g}_S - \boldsymbol{g}_D)^T \boldsymbol{g}_S}{\|\boldsymbol{g}_D - \boldsymbol{g}_S\|^2}, & \text{otherwise} \end{cases}$$

Using the obtained optimal λ^*, we estimate the resultant gradient direction as the convex combination of $\boldsymbol{g}_D$ and $\boldsymbol{g}_S$.

$$\boldsymbol{g} = \lambda^* \boldsymbol{g}_D + (1-\lambda^*)\boldsymbol{g}_S \tag{8}$$

Then at each step, the parameters of (G_γ, G_δ) are updated using the resultant gradient vector, $\boldsymbol{g}$, to optimize towards Pareto optimal solutions. This process is repeatedly executed until the end of total number of epochs. Theoretically, Multiple-Gradient Descent Algorithm (MGDA) is proven to converge to a Pareto stationary solution (Désidéri, 2012; Sener & Koltun, 2018).

D. Experimental Details

Training Details. A two-layer multilayer perceptron (MLP) is used as the γ and δ generators, with the hidden layer dimension set to 64 and LeakyReLU as the hidden layer activation function (Maas et al., 2013). To ensure output values are within the 0 to 1 range, the final layer of the γ-generator incorporates a Sigmoid function. The weights of the MLPs are the only parameters we update during training. These weights were initialized using the Kaiming method (He et al., 2015), and the Adam optimizer (Kingma & Ba, 2014) was utilized for optimization. The training process involved a batch size of 8 and spanned 2 epochs with a fixed learning rate $1e-4$. We set the temperature in Gumbel Softmax to be 0.1. Every 100 steps, a checkpoint was saved, and the best-performing checkpoint on the validation set - judged in terms of improved detection and semantic coherence - was selected for the final evaluation. The reported results reflect the performance of this selected checkpoint on the test set, and we perform the inference using a batch size of 20.

Evaluation Details on Llama-2. For Llama-2 7B and 13B, we directly loaded the original model from Hugging Face library. For Llama-2 70B, we loaded a 4-bit quantized model [2] to fit the model to a single GPU.

Hardware. The experiments are performed on Nvidia A100 GPUs with 80 GB of memory, including training and inference. Each experiment is run on a single GPU without model or data parallelism. We load the LLMs directly from huggingface (Wolf et al., 2019) in `float16`, and use `PyTorch autocast` to train our MLPs in full precision. Each training instance takes 20 hours using 60 GB of memory.

Curve-Fit. For all our results, we plot the curves given the points using `curve_fit` under Python `scipy` module.We use the five-parameter logistic curve (Gottschalk & Dunn, 2005) and the exponential curve to fit the points.

$$y = d + \frac{a-d}{(1+(x/c)^b)^g}, \quad y = -a * e^{bx} + c. \tag{9}$$

Here (x, y) are the points we trying to fit, and the letters a, b, c, d, g are parameters to fit the points. Initially, we apply the first function, which has a strong expressive ability, allowing it to fit a wide range of curve shapes. However, due to this high expressiveness and the limited number of data points, it may not always yield a concave curve. If so, we then resort to the second function, which is less versatile compared to the first, but its constrained fitting capability makes it more suitable for ensuring a concave curve shape in cases where data points are limited.

Trade-off Curves Between TPR and Semantic Similarity. The trade-off curve between true positive rate (TPR) and semantic similarity at a specified false positive rate (FPR) is plotted for our method and the baselines for comparison. In this section, we outline the procedure for plotting the trade-off curves for different methods studied in this work. For each method, we plot this curve by varying the parameters that directly influence TPR and semantic similarity among their appropriate choices.

For KGW, we identified $\gamma = 0.25$ as optimal and varied δ in $[1.0, 3.0]$ to ensure a strong watermark, i.e., TPR > 0.6 at a low FPR. We concluded that $\gamma = 0.25$ was the best choice among the possible values of $\{0.1, 0.25, 0.5\}$ specified in the paper, based on the following analysis: We fixed γ at one of the values in $\{0.1, 0.25, 0.5\}$, varied δ in the range of $[1.0, 3.0]$, and plotted their corresponding TPR and semantic similarity values, as shown in Figure 7. We observed that the curve corresponding to $\gamma = 0.25$ was slightly higher than the others. Therefore, we concluded that $\gamma = 0.25$ is relatively better, a finding also mentioned in their GitHub repository. Consequently, we identified the combination of $\gamma = 0.25$ and δ in the range of $[1.0, 3.0]$ as Pareto optimal. We then varied these parameters to obtain their corresponding TPR and semantic similarity values for KGW on the test set.

For the SWEET and SWEET$_{\text{NoPrompt}}$ methods, we set γ at 0.25 and varied δ within the range of $[1.0, 8.0]$. We explored higher δ values to improve detectability by favoring green token selection. However, even with increased δ, a 100% TPR at

[2] https://huggingface.co/TheBloke/Llama-2-70B-GPTQ

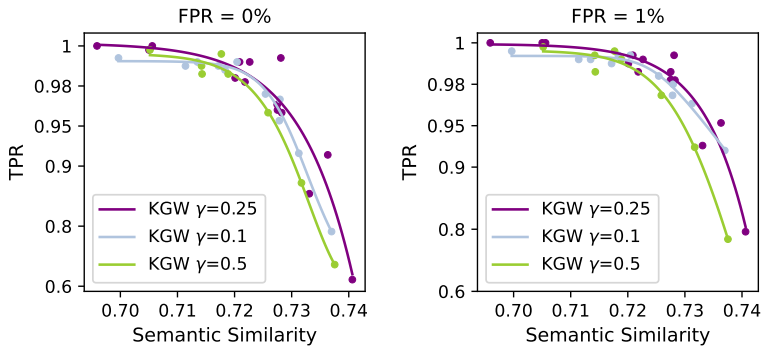


Figure 7: Fix $\gamma = \{0.1, 0.25, 0.5\}$ and varying δ for KGW.

0% FPR was not achieved. Furthermore, we did not adopt their entropy threshold value directly, as it is specifically tailored to code generation. Instead, we calculated a new threshold tailored to our training dataset, using the same method they described. Specifically, for each token in a training example, we measured the entropy of generating the next token using LLM. These measurements were then used to estimate the entropy distribution. Finally, we set the threshold as the mean of this distribution. By varying these δ parameters, we obtained corresponding TPR and semantic similarity values on the test set, which were then plotted to create the curve for SWEET. Similarly, for SIR and $\text{SIR}_{\text{NoPrompt}}$, we varied δ within the range of $[0.6, 2.0]$. They are more sensitive to the δ parameter. For SIR, during watermark detection, we use prompts and the generated completions to get watermark logits, and calculate z values by averaging the watermark logits of the generated tokens. For $\text{SIR}_{\text{NoPrompt}}$, we only use the generated completions to get watermark logits and calculate z values.

MultiBit is designed to encode a multi-bit message into generated text, with successful watermark detection occurring when the decoded message matches the embedded watermark. The message length is strategically chosen to maintain a predetermined FPR. We determined the FPR by calculating the percentage of human-written texts that were incorrectly decoded as containing the watermark message. We utilized the smallest message lengths that yielded FPRs no higher than 0% and 1%, specifically 9 and 7 bits, respectively. This baseline does not employ a splitting ratio γ. For the watermark logit δ, we vary in the range $[0.9, 10.0]$.

EXP-edit uses exponential minimum sampling which is an approach to sample from a multinomial distribution. We set the temperature to the default value of 1. Additionally, our method and other baselines employ a Top-k value of 50 during generation. However, the original EXP-edit generation process did not include the Top-k feature, so we modified EXP-edit to incorporate this feature for a fair comparison. We refer to this modified version as EXP-edit (Top-k=50). It is important to note that indistinguishable methods like EXP-edit, which are based on pseudo-random sampling during generation, cannot easily extend to other decoding strategies like greedy sampling or beam search, where no randomness is involved.

In our method, we initialize the parameters γ_t and δ_t to the values in $\{(0.1, 1.0), (0.25, 1.0), (0.25, 1.25), (0.25, 1.5), (0.25, 1.75), (0.25, 2.0)\}$ for all t. These initializations represent the six best pairs of γ and δ as shown in Figure 7. We further optimize them within our multi-objective framework to improve detectability and semantic coherence. The TPR and semantic similarity corresponding to these pairs are plotted on the test set to generate the performance curve for our method.

E. Results on PPL and Z-Score

Figure 8 illustrates the trade-off between the average z-score and SimCSE performance, as well as between the average z-score and the oracle model perplexity (PPL), measured by OPT-2.7B. In Figure 2, we have already demonstrated that our model achieves a superior Pareto frontier in terms of TPR and SimCSE. Extending this finding, Figure 8 shows that our model also surpasses KGW in both average z-score and SimCSE metrics. Additionally, in terms of PPL, our method performs comparably to the baseline methods.

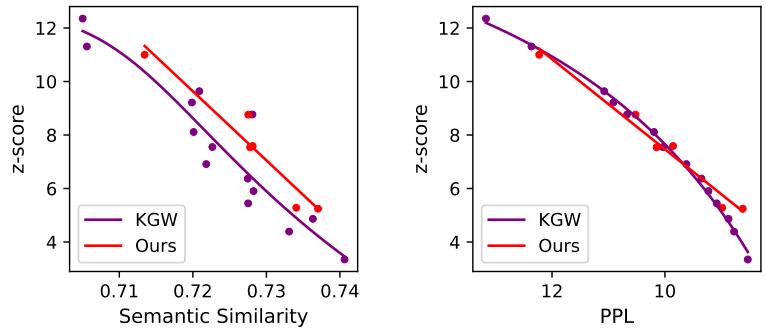


Figure 8: The trade-off between average z-score and semantic similarity (left) and oracle model perplexity (right).

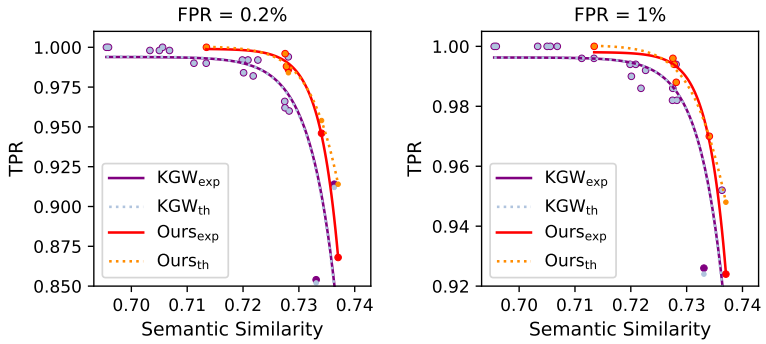


Figure 9: Comparison of our method with KGW using one-sided z-score thresholds estimated with theoretical FPR (denoted by the subscript 'th'). The results under empirical FPR (denoted by the subscript 'exp'), shown in Figure 2 of the main paper, are also included for comparison. The empirical and theoretical FPR curves are closely aligned or overlapping in most regions, indicating that they are close estimates.

F. Results Based on Theoretical FPR

We computed the theoretical FPR of our method based on the assumption used in KGW that the length of the generated text is sufficiently large. Under this assumption, the Poisson binomial distribution in our method can be approximated by a normal distribution according to the central limit theorem, enabling the use of a z-test to determine the theoretical FPR. Similarly, KGW calculates the theoretical FPR by approximating its binomial distribution with a normal distribution. Given FPR of 0.2% and 1%, the corresponding one-sided z-scores under the normal distribution are 2.88 and 2.33, respectively. The experimental results corresponding to these theoretical FPR are shown in Figure 9, where our approach demonstrated superior performance compared to KGW. It is important to note that the theoretical FPR calculations for both KGW and our method may not be entirely precise due to the approximations of distributions (Fernandez et al., 2023).

Alongside the curves corresponding to theoretical FPR, we also present curves corresponding to the empirical FPR, as shown in Figure 2 of the main paper. The empirical FPR are computed using 500 human-written texts. For KGW, we fixed $\gamma = 0.25$ as the splitting ratio, and the empirical thresholds for KGW were 2.82 and 2.17 for FPR of 0.2% and 1%, respectively. The empirical thresholds of our method, for the six different initializations $(0.25, 2.0), (0.25, 1.75), (0.25, 1.5), (0.25, 1.25), (0.25, 1.0), (0.1, 1.0)$, are as follows. At an FPR of 0.2%, the thresholds are $\{2.68, 2.74, 3.11, 3.14, 2.87, 3.30\}$. At an FPR of 1%, the thresholds are $\{2.22, 2.24, 2.42, 2.31, 2.35, 2.70\}$. Each point in our method has its own empirical FPR because we learn different splitting ratios, which impact the detection results. We observe that both the empirical and theoretical FPR curves are very close or overlapping in most regions, indicating that they are close estimates.

Table 3: Comparison of TPR and PPL for No Watermark, EXP-edit, and Our Method on OPT-1.3B. The temperature is set to 1. Top-k=0 indicates sampling from the entire vocabulary. Exponential minimum sampling is an approach to sample from a multinomial distribution.

Method	Decoding Strategy	TPR @ 0%	TPR @ 1%	PPL
No Watermark	Beam search (Num Beams=2)	-	-	1.628
	Greedy decoding	-	-	1.761
	Multinomial sampling (Top-k=50)	-	-	8.210
	Multinomial sampling (Top-k=0)	-	-	13.241
EXP-edit	Exponential minimum sampling (Top-k=50)	0.968	0.996	9.602
	Exponential minimum sampling (Top-k=0)	0.922	0.996	16.235
Ours	Beam search (Num Beams=2)	0.994	0.994	**1.847**
	Greedy decoding	0.984	0.986	2.140
	Multinomial sampling (Top-k=50)	**1.000**	**1.000**	12.227

G. Further Discussion on SWEET

At $(\gamma, \delta) = (0.25, 3.0)$, an analysis of LLM-generated texts that SWEET fails to detect at 0% FPR indicates that, on average, only 7 out of 200 tokens are high-entropy and suitable for SWEET watermarking. This limited number of watermarkable tokens diminishes SWEET's detectability, even at high δ. To illustrate this limitation, consider a hypothetical scenario: even if, in the best case scenario (which is rare), all 7 out of 7 tokens are identified as green tokens, the z-score calculated using the formula $(|s|_G - \gamma T)/\sqrt{T\gamma(1-\gamma)}$ is 4.58. As per SWEET's implementation, T is the total number of high-entropy tokens, and $|s|_G$ represents the identified green list tokens among them. However, with a higher number of watermarkable tokens, say 70, even if 50 were identified as green tokens, the z-score would increase to 8.97. Thus, the limited number of high-entropy tokens in SWEET hampers its detectability, highlighting the need for more watermarkable tokens for effective watermarking.

H. Further Discussion on EXP-edit

In this section, we delve into a detailed comparison of our method with EXP-edit. EXP-edit's generation process, which employs exponential minimum sampling, is pseudo-random. This pseudo-random sampling process enables sampling from a multinomial distribution. By using a watermark key, this pseudo-random process becomes deterministic. Detection of the watermark relies on an edit score (Kuditipudi et al., 2023), which measures the likelihood of the text being watermarked given this watermark key.

EXP-edit is considered an indistinguishable method because it does not bias the output LLM distribution towards specific tokens, instead exploiting the randomness of the sampling strategy for embedding the watermark. We examine this indistinguishable property using the perplexity (PPL) of the generated texts. We use an oracle model, OPT-2.7B, to compute PPL of generated text. PPL can estimate whether the output text is sampled from the same probability distribution, as such samples will have closer PPL values. However, it is important to note that PPL is not a reliable indicator of semantic quality. For example, the authors of Piet et al. (2023) found that PPL favored repeated texts, with one model producing a partial response followed by repeated 'l' characters, which was preferred due to its lower PPL.

The indistinguishable property of EXP-edit is evident when comparing the PPL values. Under multinomial sampling from the entire vocabulary, No watermark generation (Top-k=0) has a PPL of 13.241, while EXP-edit has a PPL of 16.235, as shown in Table 3. Similarly, with Top-k=50, the PPL of EXP-edit (9.602) is close to that of No watermark (8.210).

On the other hand, control over the decoding strategy is also necessary to meet specific goals. For instance, if the goal is to attain the lowest PPL, beam search or greedy decoding can be used. As shown, No watermark with beam search, with number of beams 2, achieves the lowest PPL of 1.628, followed by greedy decoding with a PPL of 1.761. Our method, based on KGW, biases the distribution towards a specific set of tokens for embedding the watermark and can be applied on top of any decoding strategy as a heuristic. As demonstrated, our method with beam search decoding (num beams = 2) achieves a PPL of 1.847, and with greedy decoding, a PPL of 2.140, outperforming both variants of EXP-edit. Additionally,

Table 4: Qualitative result on a randomly selected test example is presented. This includes the prompt, the no-watermark text, the watermarked text from the KGW method, and the watermarked text from our method. To assess detectability, we provide the z-scores for the no-watermark text, the KGW method, and our method. The z-score for the no-watermark text is calculated under the assumption that it was generated by our method. Additionally, we include SimCSE evaluations to measure the similarity between the text generated by the KGW and our methods, and the no-watermarked text.

Prompt	**No watermark (z-score: -0.40)**
$\cdots$ Typically, when the assessed value goes up dramatically for a big multi-tenant commercial building, the property owner isn't the one on the hook for the bill — it's the tenants. Sylvester, who previously sat on the B.C. Assessment Board and the Downtown Vancouver Business Improvement Association, said she's been discussing the idea with various people recently, including, just in the last week, the former chair of B.C. Assessment, and a provincial cabinet minister. But property-tax agent Paul Sullivan said while the idea is "not without merit," there are some	issues with it, including who would be responsible. "Who is going to be responsible for paying this? Who are the landlords?" he asked. "Are they just the large commercial property owners? Are they multi-tenant buildings where the entire business is owned by the landlord?" But Sylseter said the city could do it by creating a new low-impact category. Under her plan, which she's presented in a report, the existing triple-net category would become a low-impact category, which would allow small businesses to still pay the higher property rates. It will provide relief to the business community, $\cdots$
KGW (z-score: 5.77, SimCSE: 0.73)	**Ours (z-score: 6.45, SimCSE: 0.85)**
questions about the size of the proposed subcategory. "To my understanding it's going to be a single income property-income tax scheme,'' Sullivan said. "So as a result, there are going to be, quite a few, many (businesses) that probably would fall to that part instead of the (mainstream) part." For example, Sullivan noted, under the city's existing tax rate, a small-business operator would pay 80 per cent of the property-tax rate on an $800,000 home, and 15 per cent on anything above. But if Sylvester proposes a new subcategory within that tax bracket, that employee would pay less as a business has a "limited ability to grow their cash." $\cdots$	questions about the size of the proposed change, what it would mean to businesses that don't have large landlords, and how it would apply to all commercial-property owners. "The biggest issue is how it would apply to non-residential buildings. If you consider that the vast majority of Vancouver is for residential, what are the implications of a change that would apply to non-residential buildings only?" said Sullivan. But Sylvester said her proposal is to put the property-tax rate on small businesses in the form of an exemption, so businesses like hers — which range from small bakeries to boutique clothing shops — would not pay triple-net.$\cdots$

our method achieves good TPR values at both 0% and 1% FPR. In contrast, it is not straightforward to extend EXP-edit to greedy and beam search, which do not involve randomness.

Furthermore, the authors of Piet et al. (2023) claim that indistinguishable methods are overly restrictive, and KGW-based methods offer more freedom to the user, enabling better detectability without significant loss of text quality.

I. Further Analysis on Influence of Preceding Tokens on Watermark Strength

In Sec. 5.3, we analyze the learned watermark logits, δ, and splitting ratios, γ. Our analysis indicates that lower γ and δ values (weaker watermark) are assigned to adjectives (ADJ) and determiners (DET), likely due to the high likelihood of a noun following. In contrast, punctuation (PUNCT) is assigned higher γ and δ values (stronger watermark) possibly due to the absence of restrictions on the subsequent token. To further support this claim, we estimated the transition probabilities from adjectives, determiners, and punctuation to the next token's part of speech (POS) tag using our training dataset. Notably, our observations show that a determiner is followed by a noun with a 0.7 probability, while an adjective is followed by a noun with a 0.56 probability. In the case of punctuation, we observed a nearly uniform distribution over the next token's POS tags. These observations reinforce our claims in Sec. 5.3.

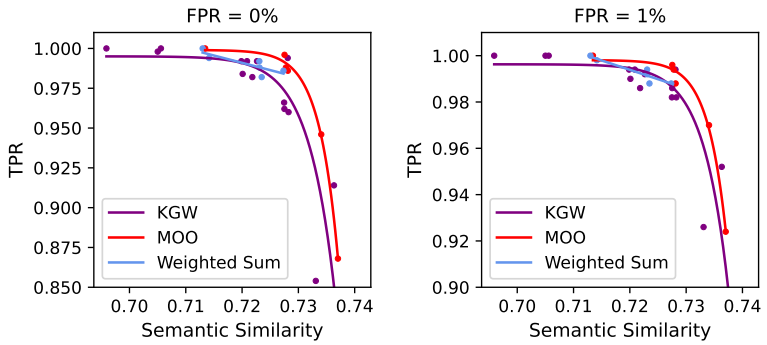
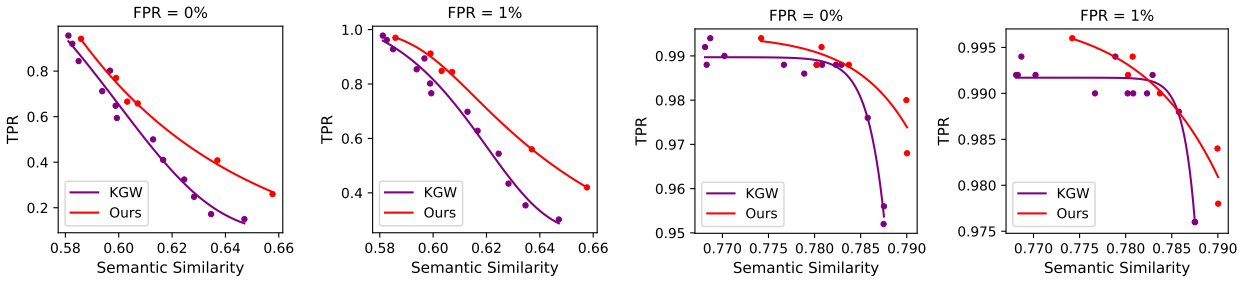


Figure 10: Comparison of multi-objective optimization and weighted sum optimization. For the weighted sum optimization, we set λ_{ws} to 4×10^{-4} based on the averaged value of $\frac{\lambda_{moo}}{1-\lambda_{moo}}$ during the optimization.



(a) Generation length = 50 tokens.

(b) Generation length = 400 tokens.

Figure 11: Comparison of our method with KGW on short and long text generations. For short text generations, we set the generation length to 50 tokens. For long text generations, we set the generation length to 400 tokens.

J. Qualitative Analysis

We present a qualitative analysis of an example randomly selected from the test set in Table 4. Using the same prompt, we compare the no-watermark generation, KGW generation, and our model's generation. We observe that our method has a higher z-score (6.45) compared to the one obtained from the KGW method (5.77). Additionally, the SimCSE score is also higher for our method (0.85) compared to KGW (0.73).

We can also qualitatively observe the impact of the token-specific watermarking employed by our method. For example, consider the sequence '...would apply to all commercial...' from the third line of our method. Here, 'apply' follows 'would,' 'to' follows 'apply,' and 'all' follows 'to,' with each of these tokens being marked as green. In contrast, the token succeeding 'all,' i.e., 'commercial,' is marked as red. Upon calculating the splitting ratio (γ) and the watermark logit (δ) for 'would,' 'apply,' 'to,' and 'all,' we find that 'all' is the only token associated with both a low γ and a low δ. This demonstrates that our method adapts its watermarking strength to maintain semantic coherence for significant words like 'commercial,' which is present in the prompt. Conversely, for tokens like 'apply,' 'to,' 'all,' our model enforces a stronger watermark, aiming to boost detectability.

On the other hand, the KGW method exhibits limitations in handling token specificity. For instance, examining the sequence '...a single income property...' from the second line of a KGW-generated text reveals this issue. KGW applies uniform γ and δ values across all tokens, leading to the generation of a green token 'income,' which is less relevant to the main idea.

K. Ablation Study: Comparison of MOO and Weighted Sum Optimization

Multi-Objective Optimization (MOO) offers a more advanced approach than traditional weighted sum optimization for balancing various objectives. In contrast to weighted sum, which depends on a fixed hyperparameter to balance two objectives, MOO dynamically estimates the optimal gradient direction, effectively managing the trade-off between the objectives.

Table 5: Memory utilization of OPT-1.3B for generating 200 tokens, measured in MB.

Method	Generation (MB)	Detection (MB)
No Watermark	3475	-
KGW	3477	503
SWEET	3477	4933
EXP-edit	3486	416
SIR	5475	2425
MultiBit	4517	1485
Ours	3731	643

To demonstrate the effectiveness of MOO, we compare it with the weighted sum approach. For weighted sum, our γ- and δ-generators are trained on a single objective function that combines L_D and L_S using a trade-off parameter λ_{ws}, formulated as:

$$L = L_S + \lambda_{ws} L_D. \tag{10}$$

In MOO, the optimal λ_{moo} is calculated for each iteration to balance the gradients from both objectives. This dynamic adjustment of λ leads us towards the Pareto optimal solution, showcasing MOO's superior capability in handling multiple objectives.

In this analysis, we set the λ_{ws} to 4×10^{-4}. This value is based on our MOO experiment for a fair comparison. Although λ_{moo} is dynamically adjusted throughout the training process, we calculate its average value across this process and use it to determine λ_{ws}. However, since the resultant gradient in MOO is a convex combination of gradients from individual objectives, using λ_{moo} as the coefficient, we set λ_{ws} as $\lambda_{moo}/(1 - \lambda_{moo})$, which turns out to be the aforementioned value. We train with the following initialization: $(0.25, 0.75)$, $(0.25, 1.0)$, $(0.25, 1.25)$, $(0.25, 1.5)$, $(0.25, 1.75)$, and $(0.25, 2.0)$.

The results in Figure 10 show that 5 out of 6 points obtained by weighted sum optimization fall below the Pareto frontier established by MOO, demonstrating the effectiveness of MOO. These findings suggest that, for weighted sum optimization, improving performance for each initialization requires estimating the optimal λ_{ws} through trial and error, a process that can be exhaustive. However, by employing MOO, our method dynamically estimates this trade-off parameter, ensuring convergence towards the Pareto optimal solution.

L. Computational Costs

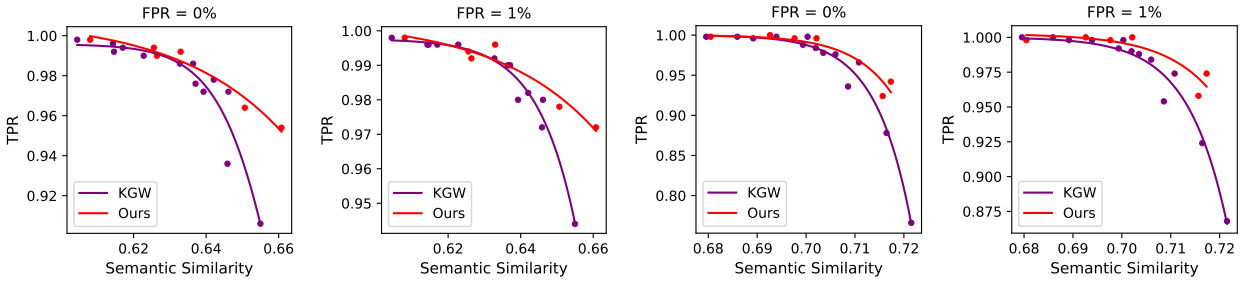
We evaluate the computational time of our method for both text generation and watermark detection, and compare it to the baselines. Table 2 shows the results. We also show memory utilization in Table 5. Our method exhibits higher generation and detection speeds compared to EXP-edit, SIR, and MultiBit. The generation and detection time of our method is comparable to that of KGW, SWEET, and No Watermarking. Overall, our method achieves superior detectability and semantic coherence without a significant increase in computational cost or memory cost.

M. Performance Across Varied Generation Lengths

In our main experiments, the generation length was set to 200 tokens. Here, we extend our evaluation to include both shorter and longer text generations. For short text generations, we configured the prompt length to at least 200 tokens, followed by a generation of 50 tokens by the LLM. For longer texts, the prompt was set to a minimum of 600 tokens, with the LLM generating 400 tokens. We used the C4 news-like dataset and the OPT-1.3B model for these text generations. As illustrated in Figure 11, our method is relatively better than KGW, showcasing the efficacy of our approach across various generation lengths.

N. Performance on Different Datasets

We further evaluated our learned models (γ- and δ-generator networks), initially trained on the C4 dataset, on two additional datasets without further training. The Essays dataset (Schuhmann, 2023) comprises sets of instructions paired



(a) Essays dataset

(b) HC3 dataset

Figure 12: Comparison of our method with KGW on the Essays and HC3 datasets.

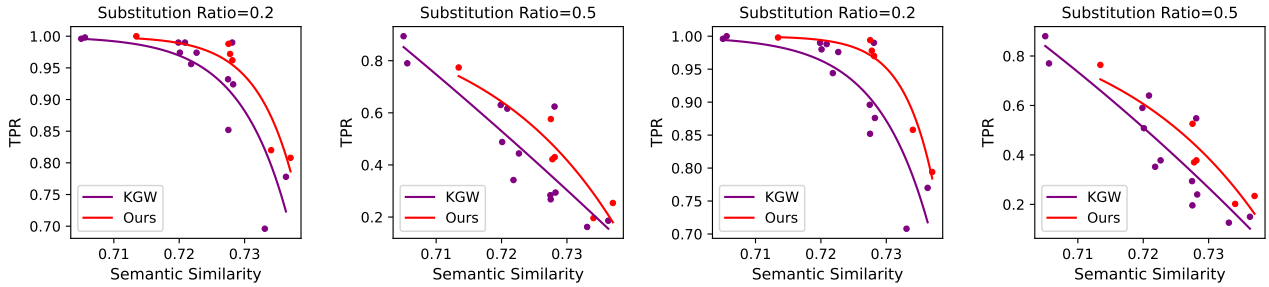
with corresponding essays. For this dataset, we used the instructions as prompts to generate essay responses. The HC3 dataset (Guo et al., 2023a) contains ChatGPT-generated text, and we use the initial 100 tokens of the ChatGPT-generated text as prompts to generate the completions. The OPT-1.3B LLM was used for the experiment, with the generation length fixed at 200 tokens. As shown in Figure 12, our method outperforms the baseline KGW method, demonstrating the robust transferability of the γ and δ networks across different datasets.

O. Modified Corruption Attack

In this evaluation, we assume the attacker knows the γ value for each token as determined by our method, enabling them to perform a corruption attack based on this knowledge. To investigate how varying γ values affect attack efficacy, we conducted experiments with three types of corruptions:

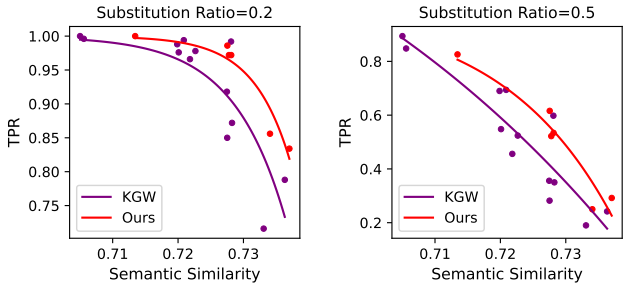
- Using our trained γ network to select 30 tokens in the vocabulary with the lowest γ values; then randomly substituting P% of the tokens in a watermarked text with tokens uniformly selected from this low-γ list.
- Using our trained γ network to select 30 tokens in the vocabulary with the highest γ values; then randomly substituting P% of the tokens in a watermarked text with tokens uniformly selected from this high-γ list.
- Randomly substituting P% of the tokens in a watermarked text with tokens drawn uniformly from the vocabulary.

We tested our method using substitution ratios (P%) of 20% and 50%, and the results are shown in Figure 13. Our method consistently outperformed KGW across all attack scenarios. This underscores the robustness of our method against Corruption Attack, even when the watermarked text is corrupted using tokens selected based on γ values derived from our trained γ network. The strong robustness of our method mainly stems from the improved z-score, achieved through our multi-objective optimization framework that concurrently maximizes the z-score and SimCSE. As a result, even when a Corruption Attack attempts to alter the γ scores of certain tokens, the remaining tokens maintain a sufficiently high z-score, facilitating the easy detection of the watermark.



(a) Substitution using low-γ tokens.

(b) Substitution using high-γ tokens.



(c) Substitution using random tokens.

Figure 13: Corruption attack where the attacker substitutes P% of tokens with low-γ, high-γ, and random tokens when the FPR is 1%. P is set to 20% and 50%.