Watermarking Conditional Text Generation for AI Detection: Unveiling Challenges and a Semantic-Aware Watermark Remedy

Yu Fu[1], **Deyi Xiong**[2], **Yue Dong**[1]*

[1]University of California Riverside
[2]Tianjin University
yfu093@ucr.edu[1], dyxiong@tju.edu.cn[2], yue.dong@ucr.edu[1]

Abstract

To mitigate potential risks associated with language models (LMs), recent AI detection research proposes incorporating watermarks into machine-generated text through random vocabulary restrictions and utilizing this information for detection. In this paper, we show that watermarking algorithms designed for LMs cannot be seamlessly applied to conditional text generation (CTG) tasks without a notable decline in downstream task performance. To address this issue, we introduce a simple yet effective semantic-aware watermarking algorithm that considers the characteristics of conditional text generation with the input context. Compared to the baseline watermarks, our proposed watermark yields significant improvements in both automatic and human evaluations across various text generation models, including BART and Flan-T5, for CTG tasks such as summarization and data-to-text generation. Meanwhile, it maintains detection ability with higher z-scores but lower AUC scores, suggesting the presence of a detection paradox that poses additional challenges for watermarking CTG.

Introduction

Language Models (LMs) have demonstrated remarkable effectiveness in generating content that closely resembles human performances across diverse tasks (Tan et al. 2023; Dong et al. 2023; Liu et al. 2023). As large-scale models such as ChatGPT (OpenAI 2021) evolve and produce increasingly human-like content, concerns have surged around potential limitations and risks tied to their use (Bender et al. 2021), including hallucination (Alkaissi and McFarlane 2023), bias and toxicity (Deshpande et al. 2023), failure in commonsense reasoning (Bian et al. 2023), and misinformation and malicious use (OpenAI 2023).

To mitigate potential risks associated with LMs, it's crucial to develop methods that differentiate between AI and human-generated content. Current AI-detection tools primarily rely on perplexity-based classifiers, assuming lower perplexity in AI-generated text (Solaiman et al. 2019; Jawahar, Abdul-Mageed, and Lakshmanan 2020; Mitchell et al. 2023; Mitrović, Andreoletti, and Ayoub 2023). Conversely, an alternative approach is to inject watermarks during generation for subsequent detection. For instance, Kirchenbauer et al. (2023) proposed using hash function to randomly bifurcate the vocabulary into "green" and "red" lists at each decoding step, serving as watermarks. This watermark provides reliable detection signals without the need to train a classifier and produces high-quality generated texts with only a slight perplexity drop in language modeling.

Different from existing research, our focus is on watermarks for conditional text generation (CTG), and we unveil the challenges associated with the use of watermarks

Input: (H) [TABLECONTEXT] (R) title (T) The Big Fix (H) [TABLECONTEXT] (R) [title] (T) Mandy Patinkin (H) The Big Fix (R) role (T) Pool Man (H) The Big Fix (R) year (T) 1978 **Target:** Mandy Patinkin was the actor for the Pool Man in the 1978 movie The Big Fix.	Num Tokens	z-score	p-value
Original Watermark Mandy Patinkin played as Pool Man in episode The Big Polliners in episode No, No, It is 1978 in TV Series and movie.	30	11.82	1e-32
Semantic Watermark In 1978, Mandy Patinkin is in the title The Big Fix, the role is Pool Man.	22	13.75	2e-43

Figure 1: The outputs with the original watermark (OW) (Kirchenbauer et al. 2023) and our proposed semantic-aware watermark (SW) on a test example from DART – a data-to-text generation benchmark – with parameters $\gamma = 0.1$ and $\delta = 5$. We expect $\sim 90\%$ of human-generated texts from the red list, whereas AI primarily utilizes the green list. Both watermarks yield high z-scores ($z > 4$), indicating strong watermark strength for detection. Yet, OW forces the algorithm to generate from the red list due to randomly assigning key source entities (Mandy Patinkin) to it. As δ increases (towards a hard watermark), excluding these red tokens risks more hallucinations (words with underline).

*Corresponding Author

(Kirchenbauer et al. 2023). Our research findings suggest that *LM watermarking algorithms cannot be seamlessly applied to CTG tasks without a notable decline in performance*: the omission of task-specific considerations leads to significant decreases observed – up to 96.99% drop with hard watermarks and 27.54% drop with soft watermarks – in conditional generation tasks including summarization (See, Liu, and Manning 2017; Narayan, Cohen, and Lapata 2018) and data-to-text generation (Gardent et al. 2017; Nan et al. 2021). Figure 1 illustrates an example where the randomly bifurcated red list (Kirchenbauer et al. 2023) contains key entities from the source that has to be generated for the data-to-text generation task; the mismatch between context and watermark not only impairs detection but also introduces 12 hallucinated words in a 30-token generation.

To enhance the effectiveness of watermarks for CTG, we propose a simple yet effective semantic-aware watermarking algorithm that leverages hash function to embed watermarks, while also taking into account the input context and the distinctive characteristics of conditional generation tasks. In particular, we strategically bifurcate the vocabulary to balance randomness and semantic relatedness to the input source using word vector similarity. These semantically-related tokens can efficiently cover a substantial portion of the information that needs to be generated in conditional text generation tasks. Consequently, their inclusion in the "green list" acts as a buffer, reducing the adverse impact of adding watermarks. Compared to the baseline watermarks, our watermark maintains detection ability with higher z-scores but lower AUC scores (higher values are better for both). This suggests the presence of a detection paradox that introduces additional challenges for watermarking CTG: *the prevalent human habit of using tokens identical/similar to the input for CTG complicates the detection of watermarks and implies a trade-off between detection ability and metric score*.

Our contributions can be summarized as follows:

- We show that directly applying Kirchenbauer et al. (2023)'s watermarking method to conditional text generation tasks, without task-specific considerations, can lead to a significant performance drop (up to 96.99%). This significant decline is observed across multiple tasks like summarization and data-to-text generation, and various text generation models such as BART and Flan-T5.
- We propose a semantic-aware watermarking algorithm that utilizes hash function while considering the input context of CTG tasks. Automatic and human evaluations on multiple datasets and models indicate that our method effectively mitigates quality degradation associated with the use of watermarks, while minimizing the trade-off in detection.

Related Work

Automatic Detection The detection of AI-generated text, particularly in the context of large language models (LLMs), has recently attracted significant research interest (Bakhtin et al. 2019; Schuster et al. 2020; Fröhling and Zubiaga 2021; Sadasivan et al. 2023; Mitchell et al. 2023). Previous approaches have primarily focused on leveraging the perplexities of generated texts for detection. For example, Solaiman et al. (2019) utilized a classifier to evaluate the total log probability of the text, using it as a means to determine whether the content originated from a machine. Building on this premise, Mitchell et al. (2023) further validated that the log probability of machine-generated text diminishes upon perturbation, while the log probability of human-written text remains unpredictable when perturbed.

Watermarking There has been a recent emergence of watermarking specific patterns into language models for AI detection. Zhao, Wang, and Li (2023) focused on injecting secret sinusoidal signals into the decoding steps for each target token by modifying the corresponding probability distribution. Kirchenbauer et al. (2023) proposed a method that randomly bifurcates the vocabulary and modifies the probability distribution during each decoding step, thereby ensuring the inclusion of detectable patterns (watermarks) in the generated text. Subsequent work by Lee et al. (2023) optimized this watermark based on entropy, while Wang et al. (2023) introduced a novel watermarking scheme that enables the watermark to convey meaningful messages such as user IDs or LLM names, expanding its purpose beyond merely indicating machine-generated text. On the other hand, Yoo et al. (2023) and Yang et al. (2023) focused on incorporating watermarks through post-processing, allowing for watermarking even in the context of black-box LLMs. In contrast to the aforementioned papers, our focus is on watermarking for conditional text generation (CTG) tasks, specifically discussing challenges in applying watermarks designed for LLMs to CTG tasks, and proposing watermarks that incorporate task-specific characteristics that account for input context for CTG.

Method

This section provides an overview of the basic principles of watermarks, elaborates on our proposed semantic-aware method, and discusses how it's integrated into the watermarking procedure for CTG.

Original Watermark Considering a language model with parameters denoted by θ, the probability distribution for the t-th token in sequence $\mathbf{S} = \{s_1, s_2, \ldots, s_{|\mathbf{S}|}\}$ can be formulated as :

$$p(s_t) = p_\theta(s_t|s_{<t}) \tag{1}$$

By considering all preceding tokens, language models (LMs) generate a probability distribution across the vocabulary and sample tokens accordingly.

Watermarking is a technique designed to incorporate robust detection signals into machine-generated text. Kirchenbauer et al. (2023) propose two methods, namely hard and soft watermarks, for adding watermarks to text by imposing vocabulary restrictions during each decoding step. Specifically, the "Hard Red List" watermarking algorithm randomly divides the vocabulary into "green" and "red" lists using a hash function and previously generated tokens. During the generation process, only tokens from the green list can be selected for the t-th position. To detect the presence

of the watermark in the generated text, a statistical analysis such as the *one proportion z-test* can be employed.

However, randomly partitioning the vocabulary and solely selecting words from the green list can hinder the generation of crucial tokens that are not included in the green list. As an alternative, the "Soft Red List" watermarking approach introduces a constant δ to the logit $l_k^{(t)}$ of tokens in the green list during prediction:

$$p_k^{(t)} = \exp(l_k^{(t)} + \delta) / \sum_i \exp(l_i^{(t)}) \tag{2}$$

This adjustment ensures that even if there are deterministic tokens not included in the green list, they can still be generated. We observe that hard watermarks can be seen as a special case of soft watermarks, achieved by adding a large δ to the tokens in the green list. Therefore, we choose soft watermarking algorithm as the unified formulation in our paper.

Semantic-Aware Watermark

In contrast to text generation tasks involving language models, conditional text generation (CTG) tasks often exhibit significant textual overlap, either at the token level or the semantic level. For instance, Chen et al. (2020) demonstrate that in the CNN/DailyMail dataset (See, Liu, and Manning 2017), over 80% of the tokens found in the summary can be located within the original document. Even in the case of the XSUM dataset (Narayan, Cohen, and Lapata 2018), known for its "abstractive" nature, this percentage remains above 60%. Consequently, random watermarking algorithms, which bifurcate the vocabulary arbitrarily at each decoding step, can drastically impair the performance of generation models.

Considering this characteristic of CTG tasks, we propose a simple yet effective semantic-aware watermarking method to enhance performance. Our approach uses the input context to extract semantically related tokens, measured by word vector similarity to the source. By incorporating semantically related tokens as a constraint, we ensure the quality of the generated output. We then apply the original watermark and randomly bifurcate the remaining vocabulary.

To implement this approach, we tokenize the input sequence $\mathbf{x}$ to $\hat{\mathbf{x}} = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{|\hat{\mathbf{x}}|}\}$. Next, the tokenized sequence $\hat{\mathbf{x}}$ is transformed into contextualized vector representations using the model's embedding layer. Integrating input information into the watermark's green list is a direct and crucial step (step 2 in Algorithm 1), consistent with the requirements of CTG tasks where the output is dependent on the input. However, it's crucial to note that output information isn't solely determined by the input. Thus, relying exclusively on input as a constraint may not yield optimal results. To overcome this limitation, we broaden the constraints by incorporating token embeddings to measure token similarities.

We extend the constraints to prioritize the inclusion of content closely related to the input within the partitioned green list, as detailed in Algorithm 1. This strategy effectively minimizes the impact of random vocabulary partitioning on the quality of generated results. The decision to utilize

Algorithm 1: Semantic-Aware Watermark

Input: Input sequence $\mathbf{x} = \{x_1, x_2, \ldots, x_{|\mathbf{x}|}\}$
Parameter: Conditional model p_θ, green list size: $\gamma \in (0, 1)$, hardness parameter: $\delta > 0$ cluster parameter: $k \in [1, 2, 5, 10]$
Output: Watermarked text y

1: *Get word embeddings and compute the $|V| \times |V|$ word similarity matrix* $\mathbf{M}$.
2: *Using input sequence* $\mathbf{x}$ *and parameter* k *to get semantically related tokens* S *and insert them to "green list"* G.
3: **for** $t = 0, 1, \cdots$ **do**
4: Apply the conditional model to input sequence $\mathbf{x}$ and get a logit vector $l^{(t)}$ over the vocabulary V.
5: Compute a hash of token y_{t-1} and use it to seed a random number generator.
6: *Using the random number generator and partition the remaining vocabulary into G of size $\underline{\gamma|V| - len(S)}$ and a "red list" R of size $\underline{(1-\gamma)|V|}$.*
7: Add δ to each green list logit. Apply these modified logits to get a probability distribution over V.
8:

$$\hat{p}_k^{(t)} = \begin{cases} \frac{\exp(l_k^{(t)}+\delta)}{\sum_{i\in R}\exp(l_i^{(t)})+\sum_{i\in G}\exp(l_i^{(t)}+\delta)}, & k \in G \\ \frac{\exp(l_k^{(t)})}{\sum_{i\in R}\exp(l_i^{(t)})+\sum_{i\in G}\exp(l_i^{(t)}+\delta)}, & k \in R \end{cases}$$

9: Sample the next token y_t according to watermarked distribution $\hat{p}^{(t)}$.
10: **end for**

model embeddings to acquire semantically related tokens – steps 1&2 in Algorithm 1 – is motivated by the following reasons:

- Semantic Relevance: By exploiting model embeddings, we capture semantic token relationships. This ensures coherent and semantically consistent text generation by identifying tokens closely linked to the input.
- Enhanced Output Quality: Including semantically related tokens in the green list elevates the relevance and quality of the generated text, aligning it more effectively with the CTG task objectives.

Assume the word embeddings for a specific model have a size of $|V| \times d_{\text{emb}}$, where $|V|$ and d_{emb} denote the vocabulary size and the dimension of the model's embeddings, respectively. Each row in this embedding matrix contains the representation of a particular indexed token. For each pair of token representations, we can calculate the embedding similarity using measures such as cosine similarity. This process allows us to construct a similarity matrix $\mathbf{M}$ of size $|V| \times |V|$ with token indices sorted based on their similarity values with respect to the token indexed at each row.

In our proposed semantic-aware watermarking approach, before partitioning the green list, we utilize the input context tokens as pivot points for the green list and leverage the similarity matrix $\mathbf{M}$. By combining this similarity matrix with a hyperparameter k, we identify the top k semantically related tokens for each input token. These semantically related to-

Dataset	Model	Method	R-1	R-2	R-L	Dataset	Model	Method	BLEU
CNN	BART-large	NW	43.80	20.88	40.73	**DART**	BART-large	NW	47.78
		OW (Hard)	33.38	8.73	30.61			OW (Hard)	6.65 ↓ 86.1%
		SW (Hard)	**43.46**	**20.75**	**40.45**			SW (Hard)	**41.04** ↓ 14.1%
		OW (Soft)	42.46	18.33	39.52			OW (Soft)	37.06 ↓ 22.4%
		SW (Soft)	**43.50**	**20.83**	**40.62**			SW (Soft)	**44.04** ↓ 7.8%
	Flan-T5-base	NW	41.78	19.57	38.66		Flan-T5-base	NW	49.55
		OW (Hard)	24.47	5.60	22.48			OW (Hard)	5.35 ↓ 89.2%
		SW (Hard)	**41.80**	**19.80**	**38.72**			SW (Hard)	**35.36** ↓ 28.6%
		OW (Soft)	38.60	16.29	35.90			OW (Soft)	39.19 ↓ 20.9%
		SW (Soft)	**41.90**	**19.86**	**38.80**			SW (Soft)	**44.18** ↓ 10.8%
XSUM	BART-large	NW	45.25	22.15	37.03	**WebNLG**	BART-large	NW	57.18
		OW (Hard)	29.60	7.15	20.83			OW (Hard)	9.25 ↓ 83.8%
		SW (Hard)	**42.44**	**18.64**	**33.91**			SW (Hard)	**48.02** ↓ 16.0%
		OW (Soft)	40.07	16.51	31.50			OW (Soft)	44.58 ↓ 22.1%
		SW (Soft)	**43.83**	**20.39**	**35.42**			SW (Soft)	**52.50** ↓ 8.2%
	Flan-T5-base	NW	39.51	16.92	31.90		Flan-T5-base	NW	59.77
		OW (Hard)	22.98	4.80	16.66			OW (Hard)	1.80 ↓ 97.0%
		SW (Hard)	**37.67**	**14.69**	**29.94**			SW (Hard)	**40.89** ↓ 31.6%
		OW (Soft)	35.23	12.58	27.52			OW (Soft)	45.42 ↓ 24.0%
		SW (Soft)	**38.79**	**15.91**	**31.03**			SW (Soft)	**53.27** ↓ 10.9%

Table 1: Main results of comparing different watermarking strategies across various datasets and models. NW (no watermark) serves as the baseline, and adding a watermark is expected to decrease performance to trade-off detection. OW (original watermark) denotes the use of the Soft or Hard watermark (Kirchenbauer et al. 2023) with hyperparameters $\gamma = 0.5$ and $\delta \in \{2, 10\}$. Our proposed SW (semantic-aware watermark) approach employs semantically related tokens to partition the green and red lists, with hyperparameters $k = 1/2/5/10$, while keeping the same values of γ and δ to ensure a fair comparison.

kens are then included in the green list, while the remaining portion of the vocabulary is randomly partitioned. This partitioning is carried out based on the mathematical equation presented in step 8 of Algorithm 1.

Experiments and Results

This section provides an overview of the datasets and models utilized in the experiments. We also present the main experimental results, including both automatic and human evaluations.

Datasets and Models

We conducted experiments to assess the generalization ability of our proposed method by utilizing models with different parameter sizes and architectures, including BART-base, BART-large (Lewis et al. 2020), Flan-T5-small, and Flan-T5-base (Chung et al. 2022). Our focus was on two distinct conditional text generation tasks: summarization - CNN/DailyMail (See, Liu, and Manning 2017) and XSUM (Narayan, Cohen, and Lapata 2018), and data-to-text generation - DART (Nan et al. 2021) and WebNLG (Gardent et al. 2017). These datasets are widely recognized for evaluating text summarization and data-to-text generation models, respectively. By conducting comprehensive evaluations across multiple datasets, tasks, and models, our objective was to thoroughly compare the differences between the original watermarking algorithm (Kirchenbauer et al. 2023) and our proposed semantic-aware watermarking approach.

Main Results

Our main experimental results are presented in Table 1. The summarization task was evaluated using the ROUGE metric (Lin 2004), while the data-to-text generation task was evaluated using BLEU (Papineni et al. 2002). The table illustrates the performance of the models under various watermarking methods, highlighting the enhancements achieved by incorporating semantic constraints in watermarking for both the summarization and data-to-text generation tasks. Our proposed semantic-aware watermarking method exhibits significant improvements in comparison to the original watermarking method across all datasets and models.

Additionally, we observe that hard watermarks invariably cause a greater decline in CTG performance compared to soft watermarks (especially ROUGE-2 for summarization and BLEU for data-to-text generation). The hard watermarks designed for language models (Kirchenbauer et al. 2023) essentially completely forbid generation from the red list that might contain key input context, potentially leading to near-ineffective generations with almost no overlap with the reference generations. For example, in the data-to-text generation task, the original hard watermarking method adversely affects Flan-T5-small's performance on WebNLG, resulting in a decrease of over 57.97 BLEU points with 97.0% of performance drop. In contrast, our semantic-aware watermark effectively mitigates the impact of adding the watermark, demonstrating an 39.09 BLEU point increase over the original watermark with a performance improvement of 21.67 times.

SW (ours) vs. OW	Judge 1	Judge 2	Judge 3	Avg.
SW (ours) preferred	58%	54%	54%	55.33%

Table 2: Human evaluation results on 100 randomly sampled examples, accompanied by generations from BART-base with original soft or semantic-aware watermarks, presented in a random and anonymized order. Each example was independently annotated by three annotators, resulting in an average pairwise inter-annotator agreement of 63.33%.

More notably, on the CNN/DailyMail dataset, our semantic-aware watermarking method applied to the Flan-T5-base models not only mitigates the drawbacks of watermark injection but also surpasses the performance of the original generation without watermark. This can be credited to the nature of the summarization task, where a considerable amount of the target information is already present in the input. The semantic-aware watermarking method enhances the generation process by effectively harnessing this input, enabling it to capture the essential details for creating high-quality summaries. This synergy between input and target data contributes to the superior performance of the Flan-T5-small and Flan-T5-base models when utilizing the semantic-aware watermarking method in summarization tasks.

Human Evaluation In addition, we conducted a human evaluation comparing BART-base with the original and our proposed watermarks on the XSUM dataset. The human judges[1] were presented with reference summaries and generations from different watermarking algorithms in a random and anonymized order. The judges were asked to evaluate which system's summary was better and more similar to the reference. They were instructed to read the source article only when they were unable to decide or needed additional information[2].

Table 2 presents the results of the human evaluation. With a confidence level of 95% and one-sided A/B tests, the semantic-aware watermark exhibits a significantly higher preference according to human judges ($p = 0.0358$). Specifically, the preference for the semantic-aware watermark (55.33%) surpasses that of the original watermark (44.67%) by a substantial margin of 10.66%. Moreover, pairwise inter-annotator agreement was assessed, resulting in agreement percentages of 70%, 66%, and 54% for the respective evaluations. These findings strongly support the effectiveness of the semantic-aware watermarking method, highlighting its ability to enhance the quality of summarization outputs.

Watermark Strength and Detection

To evaluate the quality of watermarking for detection, we followed established research (Kirchenbauer et al. 2023; Yang et al. 2023) and assessed the strength using the average z-score and the area under the curve (AUC) score. Figure 2 and Figure 3 present the z-score and AUC results, respecively.

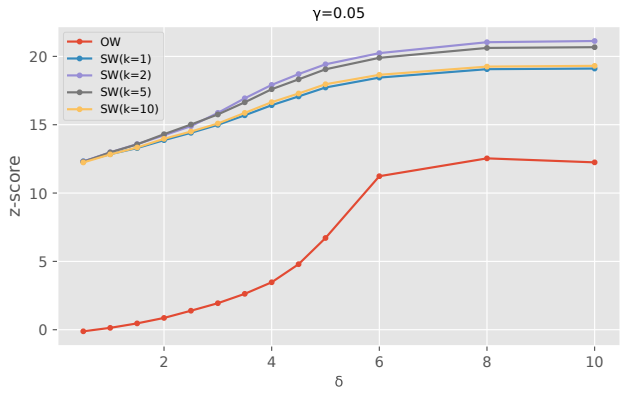

Figure 2: Watermark detection: average z-score under different δ settings (x-axis). Higher z-scores indicate stronger watermark detection confidence. We can see that hard watermarks (greater δ) are easier to detect but lead to a more significant decline in CTG performance.

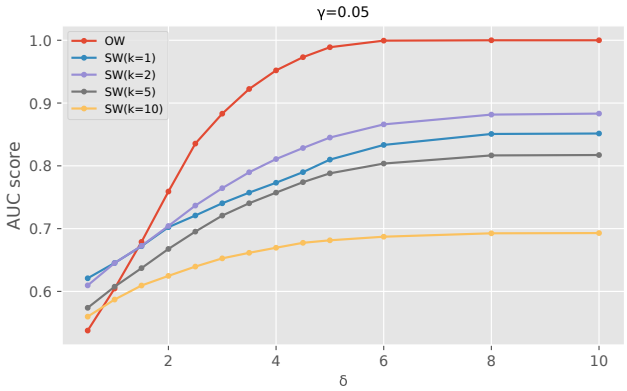

Figure 3: Watermark detection: AUC scores under different δ settings. Higher AUC scores indicates a better detection performances.

A higher z-score generally indicates a greater presence of tokens from the "green list" in the generated results, increasing the likelihood of successful detection. However, in the context of conditional text generation tasks, maintaining consistency in the length of the generated results with the original model is crucial. It has been observed that the z-score tends to increase with the length of the generated text (Kirchenbauer et al. 2023). To address this, we introduce an additional penalty term to the z-score, incorporating the ratio of the average length of the generated results to the average length of the original model's output without the watermark.

As seen in Figure 2, the semantic-aware watermarking method significantly outperforms its counterpart in terms of z-score, reflecting a higher inclusion of "green list" tokens in the generated output. Under normal circumstances (e.g., lan-

[1]All judges are native English speakers with a minimum of a bachelor's degree and were compensated at a rate of $19.5/h.

[2]We made the decision to make reading the source article optional for the judges in order to prevent creating a significant cognitive burden and to encourage them to take shortcuts.

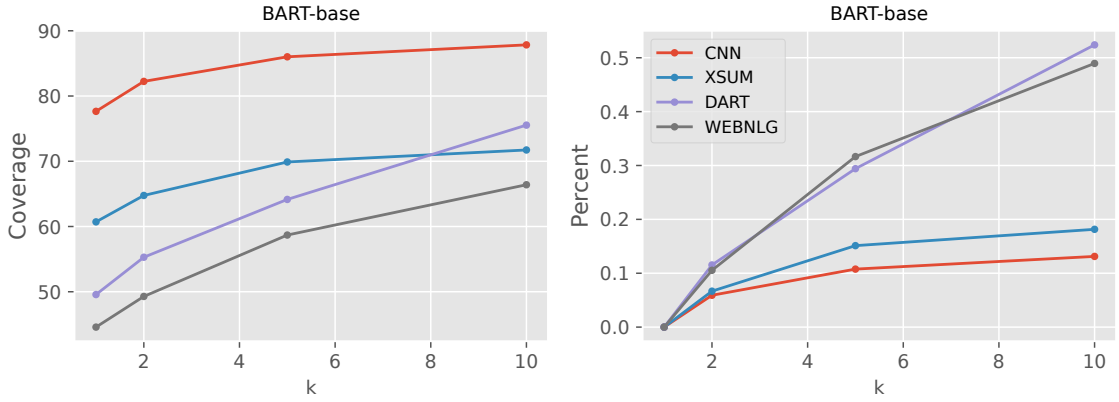

Figure 4: The coverage of target tokens by semantically related tokens varies with different datasets and values of the hyperparameter k on BART-base. Increasing the value of k improves the coverage of semantic tokens, aligning with our objective and motivation.

guage modeling), a higher average z-score indicates stronger detectability (Kirchenbauer et al. 2023). However, as Figure 3 illustrates, the AUC curve for the original watermarking method surpasses ours, as our constructed green lists incorporate more input tokens that humans would commonly use. Consequently, human-generated text also contains more green list tokens.

The disparity between the z-scores and AUC scores of semantic-aware watermarks highlights an additional challenge in applying watermarks for CTG: **the common human practice of utilizing input-similar tokens in CTG introduces complexity to the watermark detection process**. Our method, despite showing remarkable improvements in ROUGE or BLEU metrics and hence bearing closer resemblance to the reference, contributes to a slight dip in the final AUC scores. This scenario indicates a trade-off between enhancing the ROUGE or BLEU scores, indicative of increased similarity to the reference, and preserving detectability. Notwithstanding this, our empirical results compellingly argue that the significant rise in performance (up to $\sim 2167\%$) outweighs the detection decreases (Avg. $\sim 12.6\%$); further increasing this advantage margin remains an area for future exploration.

Analysis

This section analyzes the hyperparameters, focusing on: k, introduced by our semantic watermark; γ and δ, inherited from Kirchenbauer et al. (2023).

Semantic k Analysis

The semantic-aware watermark uses a hyperparameter, k, to determine the extent of semantically related tokens, derived from word embedding similarities during decoding, that are integrated into the green list. Table 3 shows that **increasing k in semantic-aware watermarks improve the CTG performance**. We hypothesize that this improvement stems from that increasing k includes more reference tokens in the green list, leading to a broader coverage of tokens that humans typically use for CTG generation.

Method	BLEU		
γ	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$
NW	45.90	-	-
OW	37.32	35.99	39.01
SW (k=1)	37.23	38.46	41.36
SW (k=2)	38.10	39.29	42.01
SW (k=5)	38.87	38.63	42.24
SW (k=10)	**41.37**	**42.89**	**44.59**

Table 3: The effect of the hyperparameter k on the results of the DART dataset using the BART-base with $\gamma \in \{0.25, 0.5, 0.75\}$ and $\delta = 2$.

To validate our hypothesis and study the relationship between k and target token coverage, we carried out experiments by measuring the overlaps between semantically related tokens and the reference target tokens under different k values. Figure 4 (left) presents curves, which, with increasing k, demonstrate a correlation with an increased proportion of target unigram text tokens covered by semantically related tokens.

Interestingly, when we adjust the setup to measure the relative percentage of coverage increase with higher k values, we observe different trends for various CTG tasks. Figure 4 (right) indicates that watermarks with larger k values have a more significant performance improvement impact on data-to-text generation tasks compared to summarization tasks. This observation is also reflected in the findings that an increased k leads to substantial improvements in BLEU scores for data-to-text generation, compared to the ROUGE score improvements for summarization (More details in Appendix). Specifically, DART and WEBNLG show greater sensitivity to k, where its increase yields better results.

γ and δ Analysis

The soft watermarking method (Kirchenbauer et al. 2023) depends on two hyperparameters: γ and δ. γ regulates the

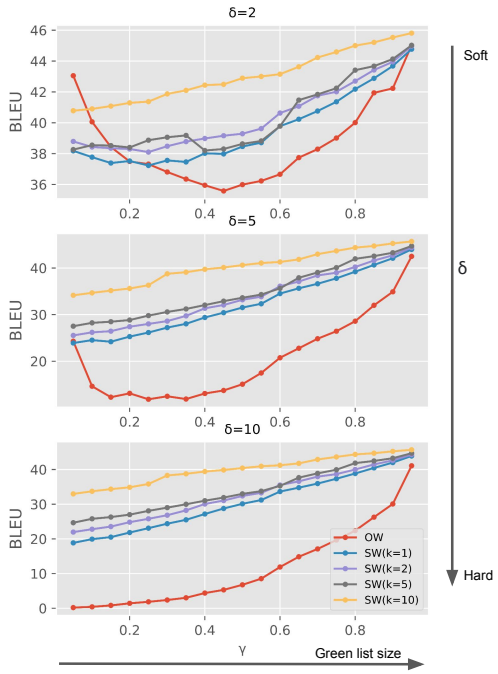

Figure 5: The impact of γ on DART results with settings of $\delta = 2/5/10$. γ controls the size of the green list. From $\delta = 2$ to $\delta = 5$, the watermarking method tends to change from a soft watermark to a hard watermark, and the probability of generating tokens from the green list gradually increases.

size of the green list during partitioning, whereas δ dictates the intensity of watermarks applied to the logits of green list tokens. Essentially, a very large δ (e.g., 10) is equivalent to the hard watermarks that entirely prohibits tokens from the red list from being generated. This section compares original and semantic-aware watermarks under varying γ and δ values, demonstrating that our proposed watermark consistently outperforms the original across different hyperparameter settings.

Increasing γ incorporates more words into the green list, typically lessening the watermark's impact on model performance. Surprisingly, Table 3 shows that the original watermarking method performs poorly when $\gamma = 0.5$. To further explore possible reasons for this and to test our method under different setups, we conducted a comparative analysis with varying γ and δ set to 2, 5, and 10. Figure 5 indicates that the semantic-aware watermark **consistently** outperforms the original watermark, except when δ is set to 2 with relatively small γ values. Decreasing γ reduces the number of selected and enhanced tokens due to the smaller green list size. As a result, the model's performance is expected to gradually decrease with a smaller watermark. However, the change curve of the original method in the $\gamma < 0.2$ (when δ=2) range deviates from these expectations.

We hypothesize that this irregularity arises from the negligible impact of soft watermark when γ is small. This happens when soft watermarks with an extremely small green list scarcely affect logits predictions. To confirm this, we examined the impact of varying δ on the BART-base model's performance using the DART dataset under extrem small γ, as shown in Figure 6. We observe that when γ is set extremely low ($\gamma = 0.05$) in the soft watermark settings (i.e., $\delta \leq 4$), there is hardly any performance trade-off upon adding watermarks, suggesting ineffective watermarks for detection.

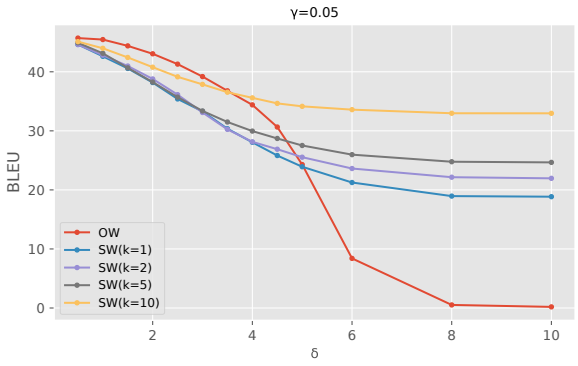

Figure 6: The impact of δ, which controls the extent of enhancement applied to the logits, on the DART results.

Dataset \ k	1	2	5	10
DART	0.0004	0.0009	0.0020	0.0037
WebNLG	0.0005	0.0009	0.0022	0.0039

Table 4: The percentage of semantically related tokens to the size of the vocabulary V.

In addition, to ensure that semantically related tokens included in the green list for the semantic-aware watermark do not exceed the green list size, especially the ones obtained with a large k, we calculate the percentage of these semantically related tokens relative to the overall vocabulary size. Table 4 reveals that it is significantly lower than the green list size dictated by γ.

Conclusion

Our study reveals a significant performance drop when random watermarks are directly applied to conditional text generation tasks without considering the task-specific context. To tackle this challenge, we propose a semantic-aware watermarking algorithm that incorporates hash function and carefully takes into account the input context of conditional generation tasks. We extensively evaluated our method on diverse datasets and models, including summarization, data-to-text generation, and various text generation models like BART and Flan-T5. The results demonstrate that our proposed method effectively mitigates the quality degradation associated with watermarking techniques, as confirmed by both automatic and human evaluations. These findings emphasize the importance of task-specific approaches when applying watermarking methods to ensure optimal performance in conditional text generation tasks.

References

Alkaissi, H.; and McFarlane, S. I. 2023. Artificial hallucinations in ChatGPT: implications in scientific writing. *Cureus*, 15(2).

Bakhtin, A.; Gross, S.; Ott, M.; Deng, Y.; Ranzato, M.; and Szlam, A. 2019. Real or Fake? Learning to Discriminate Machine from Human Generated Text. arXiv:1906.03351.

Bender, E. M.; Gebru, T.; McMillan-Major, A.; and Shmitchell, S. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, 610–623.

Bian, N.; Han, X.; Sun, L.; Lin, H.; Lu, Y.; and He, B. 2023. Chatgpt is a knowledgeable but inexperienced solver: An investigation of commonsense problem in large language models. *arXiv preprint arXiv:2303.16421*.

Chen, Y.; Liu, P.; Zhong, M.; Dou, Z.-Y.; Wang, D.; Qiu, X.; and Huang, X. 2020. CDEvalSumm: An Empirical Study of Cross-Dataset Evaluation for Neural Summarization Systems. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 3679–3691. Online: Association for Computational Linguistics.

Chung, H. W.; Hou, L.; Longpre, S.; Zoph, B.; Tay, Y.; Fedus, W.; Li, Y.; Wang, X.; Dehghani, M.; Brahma, S.; Webson, A.; Gu, S. S.; Dai, Z.; Suzgun, M.; Chen, X.; Chowdhery, A.; Castro-Ros, A.; Pellat, M.; Robinson, K.; Valter, D.; Narang, S.; Mishra, G.; Yu, A.; Zhao, V.; Huang, Y.; Dai, A.; Yu, H.; Petrov, S.; Chi, E. H.; Dean, J.; Devlin, J.; Roberts, A.; Zhou, D.; Le, Q. V.; and Wei, J. 2022. Scaling Instruction-Finetuned Language Models. arXiv:2210.11416.

Deshpande, A.; Murahari, V.; Rajpurohit, T.; Kalyan, A.; and Narasimhan, K. 2023. Toxicity in chatgpt: Analyzing persona-assigned language models. *arXiv preprint arXiv:2304.05335*.

Dong, Y.; Jiang, X.; Jin, Z.; and Li, G. 2023. Self-collaboration Code Generation via ChatGPT. arXiv:2304.07590.

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

Gardent, C.; Shimorina, A.; Narayan, S.; and Perez-Beltrachini, L. 2017. The WebNLG Challenge: Generating Text from RDF Data. In *Proceedings of the 10th International Conference on Natural Language Generation*, 124–133. Santiago de Compostela, Spain: Association for Computational Linguistics.

Jawahar, G.; Abdul-Mageed, M.; and Lakshmanan, L., V.S. 2020. Automatic Detection of Machine Generated Text: A Critical Survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, 2296–2309. Barcelona, Spain (Online): International Committee on Computational Linguistics.

Kirchenbauer, J.; Geiping, J.; Wen, Y.; Katz, J.; Miers, I.; and Goldstein, T. 2023. A Watermark for Large Language Models. arXiv:2301.10226.

Lee, T.; Hong, S.; Ahn, J.; Hong, I.; Lee, H.; Yun, S.; Shin, J.; and Kim, G. 2023. Who Wrote this Code? Watermarking for Code Generation. arXiv:2305.15060.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7871–7880. Online: Association for Computational Linguistics.

Lin, C.-Y. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, 74–81. Barcelona, Spain: Association for Computational Linguistics.

Liu, J.; Xia, C. S.; Wang, Y.; and Zhang, L. 2023. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. arXiv:2305.01210.

Mitchell, E.; Lee, Y.; Khazatsky, A.; Manning, C. D.; and Finn, C. 2023. DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature. arXiv:2301.11305.

Mitrović, S.; Andreoletti, D.; and Ayoub, O. 2023. ChatGPT or Human? Detect and Explain. Explaining Decisions of Machine Learning Model for Detecting Short ChatGPT-generated Text. arXiv:2301.13852.

Nan, L.; Radev, D.; Zhang, R.; Rau, A.; Sivaprasad, A.; Hsieh, C.; Tang, X.; Vyas, A.; Verma, N.; Krishna, P.; Liu, Y.; Irwanto, N.; Pan, J.; Rahman, F.; Zaidi, A.; Mutuma, M.; Tarabar, Y.; Gupta, A.; Yu, T.; Tan, Y. C.; Lin, X. V.; Xiong, C.; Socher, R.; and Rajani, N. F. 2021. DART: Open-Domain Structured Data Record to Text Generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 432–447. Online: Association for Computational Linguistics.

Narayan, S.; Cohen, S. B.; and Lapata, M. 2018. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 1797–1807. Brussels, Belgium: Association for Computational Linguistics.

OpenAI. 2021. Chatgpt: Optimizing language model for dialogue. https://www.openai.com/blog/chatgpt/. Accessed: 2023-01-10.

OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311–318. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics.

Sadasivan, V. S.; Kumar, A.; Balasubramanian, S.; Wang, W.; and Feizi, S. 2023. Can AI-Generated Text be Reliably Detected? arXiv:2303.11156.

Schuster, T.; Schuster, R.; Shah, D. J.; and Barzilay, R. 2020. The Limitations of Stylometry for Detecting Machine-Generated Fake News. *Computational Linguistics*, 46(2): 499–510.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1073–1083. Vancouver, Canada: Association for Computational Linguistics.

Solaiman, I.; Brundage, M.; Clark, J.; Askell, A.; Herbert-Voss, A.; Wu, J.; Radford, A.; Krueger, G.; Kim, J. W.; Kreps, S.; McCain, M.; Newhouse, A.; Blazakis, J.; McGuffie, K.; and Wang, J. 2019. Release Strategies and the Social Impacts of Language Models. arXiv:1908.09203.

Tan, Y.; Min, D.; Li, Y.; Li, W.; Hu, N.; Chen, Y.; and Qi, G. 2023. Can ChatGPT Replace Traditional KBQA Models? An In-depth Analysis of GPT family LLMs' Question Answering Performance. arXiv:2303.07992.

Wang, L.; Yang, W.; Chen, D.; Zhou, H.; Lin, Y.; Meng, F.; Zhou, J.; and Sun, X. 2023. Towards Codable Text Watermarking for Large Language Models. arXiv:2307.15992.

Yang, X.; Chen, K.; Zhang, W.; Liu, C.; Qi, Y.; Zhang, J.; Fang, H.; and Yu, N. 2023. Watermarking Text Generated by Black-Box Language Models. arXiv:2305.08883.

Yoo, K.; Ahn, W.; Jang, J.; and Kwak, N. 2023. Robust Multi-bit Natural Language Watermarking through Invariant Features. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2092–2115. Toronto, Canada: Association for Computational Linguistics.

Zhao, X.; Wang, Y.-X.; and Li, L. 2023. Protecting Language Generation Models via Invisible Watermarking. arXiv:2302.03162.