Watermarking LLM-Generated Datasets in Downstream Tasks

Yugeng Liu[1] Tianshuo Cong[2] Michael Backes[1] Zheng Li[3] Yang Zhang[1]

[1]*CISPA Helmholtz Center for Information Security* [2]*Tsinghua University*
[3]*Shandong University*

Abstract

Large Language Models (LLMs) have experienced rapid advancements, with applications spanning a wide range of fields, including sentiment classification, review generation, and question answering. Due to their efficiency and versatility, researchers and companies increasingly employ LLM-generated data to train their models. However, the inability to track content produced by LLMs poses a significant challenge, potentially leading to copyright infringement for the LLM owners. In this paper, we propose a method for injecting watermarks into LLM-generated datasets, enabling the tracking of downstream tasks to detect whether these datasets were produced using the original LLM. These downstream tasks can be divided into two categories. The first involves using the generated datasets at the input level, commonly for training classification tasks. The other is the output level, where model trainers use LLM-generated content as output for downstream tasks, such as question-answering tasks. We design a comprehensive set of experiments to evaluate both watermark methods. Our results indicate the high effectiveness of our watermark approach. Additionally, regarding model utility, we find that classifiers trained on the generated datasets achieve a test accuracy exceeding 0.900 in many cases, suggesting that the utility of such models remains robust. For the output-level watermark, we observe that the quality of the generated text is comparable to that produced using real-world datasets. Through our research, we aim to advance the protection of LLM copyrights, taking a significant step forward in safeguarding intellectual property in this domain.

1 Introduction

Large Language Models (LLMs) continue to demonstrate remarkable versatility and transformative potential across diverse sectors [5, 9, 18, 22, 39, 47, 55]. As these technologies become more accessible, users are increasingly generating customized content and datasets for their own use, raising important questions about content attribution and authenticity verification. For example, many existing language models, including Vicuna [1], LLaVA [25], and MiniGPT-4 [56], have leveraged GPT-4 [35] as a data source through systematic querying to generate their training datasets. This practice presents a significant challenge for LLM providers in terms of protecting their intellectual property through conventional watermarking techniques. The difficulty arises because these derivative models typically undergo extensive pre-training before any fine-tuning or alignment processes occur. Even when watermarks are present in the generated text used for training, the substantial pre-training foundation of these models, combined with their inherent ability to paraphrase and reformulate content, effectively eliminates or significantly diminishes the detectability of such watermarks. This architectural characteristic of language models, particularly their ability to maintain semantic meaning during pre-training, makes it exceptionally challenging to maintain persistent watermarks through the model development pipeline.

Numerous papers [16, 17, 20, 24, 40, 53] have recognized the importance of protecting intellectual property rights in LLMs and proposed various algorithmic solutions for embedding watermarks in LLM outputs. However, these watermarking techniques face significant limitations when confronted with paraphrasing operations or fine-tuning processes, resulting in substantially diminished effectiveness. The deterioration of watermark robustness presents a fundamental challenge, as these existing watermarks alone prove insufficient in addressing the degradation during downstream fine-tuning. This vulnerability creates a considerable obstacle for defenders attempting to detect unauthorized usage of their LLM-generated datasets in downstream model development. The challenge is particularly acute when monitoring whether these downstream models have incorporated data for training purposes.

Methodology. We propose a classification framework for user interactions, building upon previous work [28]. This framework delineates two different categories of tasks. The first category comprises classification tasks, termed *input-level tasks*, where users employ LLMs to assign labels to samples. In constructing datasets for these tasks, users input text into the LLM to obtain corresponding labels for specific text instances. For example, in sentiment classification, users utilize LLMs to categorize individual sentences as either positive or negative. For this task, we develop two approaches for watermark implementation. The first method employs a traditional trigger-based watermark strategy [4, 6, 10, 14, 30, 33, 44, 45], wherein specific triggers are embedded within the dataset. During the verification phase, these triggers are introduced into the text to validate whether the classification aligns with pre-defined categories. The second method represents a more sophisticated and stealthy approach, where we transform the conventional trigger mechanism into an alternative stylistic form [37], such as poetry. During verification, the system evaluates whether inputs formatted in poetic style are consistently classified into designated categories.

The second category encompasses generation tasks, designated as *output-level tasks*. This classification is particularly relevant in LLM fine-tuning or alignment processes, where users query more powerful LLMs to generate outputs that serve as target outputs for their own models. This approach is commonly employed in scenarios where developers leverage more advanced LLMs to create training data for their own model development. In this task, we explore three approaches for watermark implementation. Our initial method utilizes a scoring mechanism based on word selection from green-red lists [16,17]. However, this approach demonstrates significant limitations in practical applications. Our second approach refines the methodology by narrowing the green list to specific vocabulary terms and verifies the watermark by detecting them in the output. Finally, we investigate a more stealthy and subtle approach that leverages grammatical modifications as watermarks. This method involves systematic alterations of sentence structure, specifically manipulating tense and voice constructions.

Findings. Our extensive experimental evaluation demonstrates the robust effectiveness of our proposed watermarking methodologies across both input and output-level implementations. In the input-level watermark evaluations, our approach achieved exceptional performance, with watermark success rates consistently exceeding 0.900 across all test scenarios. The generated datasets proved highly effective for downstream model training, achieving clean accuracy scores above 0.950 on several benchmark datasets while maintaining strong performance on real-world test cases. For the output-level watermark, each of our defined methods demonstrates a measurable effective watermark to varying degrees without significant model utility degradation. The maintenance of model performance while successfully implementing watermarking mechanisms provides strong empirical evidence for the viability of our multi-faceted watermarking approach. These comprehensive results validate the effectiveness of our watermark methods and suggest their potential for practical deployment in protecting intellectual property in language model applications.

2 Threat Modeling

In the paper, we consider two parties: the *adversary* and the *defender*. The adversary aims to use the LLMs to generate their own datasets, and then, they will use these datasets to train their own downstream models. We envision the defender, on the contrary, as the owner of the victim LLMs, whose goal is to protect the copyright of the contents from their models when publishing the LLMs as an online service. However, for the defender, tracking whether the content generated by their model has been used to train other models presents significant challenges. One potential approach might be to employ membership inference attacks, but here, we assume that LLM owners do not participate in the training process of downstream models. As a result, defenders have no knowledge of the specific datasets generated by users for model training, nor can they feasibly create a shadow dataset. This is further complicated by the fact that different hyperparameters in LLM generation can lead to substantial variations in dataset distribution [28], making membership inference attacks an unsuitable solution in this context.

2.1 Capability

The adversary can directly utilize the LLM, but the defender has no ability to intervene in the input provided by the adversary. Similarly, the defender does not participate in the training process of any datasets generated by the adversary. Although the defender technically has the ability to access all generated content, we do not assume they will take this approach. Beyond the cost implications for the defender, such a method would be inefficient for watermark detection, as they have no way of knowing whether the sampled content has been used for model training. Moreover, membership inference techniques are not applicable in this scenario. However, the defender can influence the content produced by the LLM by using system prompts or adjusting the logit bias within the word list to amplify the likelihood of certain terms because they have full access to the LLMs. This method is a common and effective approach [28,36] for guiding text generation in LLMs.

Adversary's Motivation. For adversaries, their primary goal is to reduce costs. Most existing benchmark datasets, such as AG News [52], DBpedia [52], and IMDb movie [2], are typically created by manually filtering and classifying vast news articles or reviews, eventually forming a cohesive dataset. These datasets must adhere to strict requirements, including uniform format, similar length, no extraneous language, and consistent distribution within the same category. This process is labor-intensive and tedious. However, these challenges align perfectly with the strengths of LLMs, which can generate datasets that meet all these criteria – consistent format, length, language, and distribution – without requiring significant human intervention. Moreover, LLM-generated data offers substantial cost savings compared to the high manual labor costs. Consequently, LLM-generated datasets are increasingly being used in contemporary research. For adversaries, the laborious task of manually creating datasets is being replaced by leveraging LLMs to generate specific data efficiently.

Defender's Motivation. For the defender, the primary goal is to protect the copyright of content generated by the LLM. To achieve this, they inject watermarks within the generated content. In this scenario, we assume the defender is the LLM owner, whereas they do not intervene in the user-generated content nor participate in any downstream task training. Unlike previous studies [6, 16], where defenders might directly verify user-generated content, here, the defender does not have access to the content created by users. Therefore, rather than directly inspecting the text, the defender must employ distinct detection strategies tailored to different types of downstream tasks. For classification tasks, the defender injects undetectable triggers within the generated text, which remain unknown to the adversary. In generation tasks, the defender controls the content of the generated datasets, for instance, using red-green lists or specific terms, ensuring that the downstream outputs contain the injected watermark.

2.2 Category

In general, adversaries can perform completely different downstream tasks, such as answering questions, summarizing documents, translating languages, and completing sentences. We categorize these downstream tasks into two levels according to different requirements and purposes.

Input-Level. LLMs are usually used to generate the content used to train a classifier. These tasks require the adversary to specify the categories of sentences to be generated, such as distinguishing between sentiment – whether a sentence is positive or negative – or the type of news, such as sports or political news. In our experiments, we found that the diversity of content generated by LLMs was not exceptionally high. As a result, directly prompting the LLM to generate a purely positive or negative sentence is quite challenging. Therefore, we assume that adversaries are more likely to provide specific contexts, such as a movie description, and then prompt the LLM to generate a positive or negative review based on that description. Similarly, they might provide a specific news topic and relevant locations or people to generate relevant content.

Output-Level. In this task, the adversary queries the LLM to obtain answers, which are then used as the outputs for their downstream model. Unlike input-level generation, where considerations of LLM diversity are critical, the adversary does not need to account for such diversity. Instead, their focus shifts to preparing the question dataset used for querying the LLM. For instance, if the users want to train a downstream model for summarizing the conversation, they need a dataset within conversations to query the LLMs.

Based on these two distinct types of tasks in the downstream models, we have designed two different watermark methods: input-level watermark (see Section 3) and output-level watermark (see Section 4).

3 Input-Level Watermark

In this section, we introduce the input-level watermark mechanism. Based on its design, this mechanism includes two distinct watermark methods: traditional watermark and stylistic watermark.

3.1 Traditional Watermark

Since the generated datasets are typically used for classification tasks, we can directly inject triggers into the datasets during the LLM generation process, following previous watermark techniques [4, 6, 10, 14, 30, 33, 44, 45]. The sentences containing these triggers are then directed toward specific categories, ensuring that the watermark is systematically injected into the data. The limitation of this approach lies in the fact that, since the defenders cannot intervene in the inputs from users, they must rely on prompt engineering to inject the trigger into the generated output. Once the user obtains this output, the defender does not participate in any subsequent downstream model training processes. If the defender intends to verify later whether their LLM-generated data has been used for training a model, they will rely on the previous trigger to facilitate this verification.

Algorithm 1 Traditional & Stylistic Watermark

```
Input: user input $\mathbf{u}$, LLM model $\mathcal{M}$, number of classes $C$,
    target class $c_t$, trigger $\mathbf{T}$, dataset size $N$, total number of
    watermarked samples $n$
Output: Watermarked dataset $\mathbf{D_w} = \{(o_i, c_i)\}_{i=1}^{N}$
 1: Initialize empty dataset $\mathbf{D_w}$
 2: for $i = 1$ to $N$ do
 3:     Randomly select a class label $c_i$ from $\{0, \dots, C-1\}$
 4:     Generate a system prompt $p_i$ for dataset generation
 5:     if $c_i = c_t$ and number of watermarked samples $< n$
        then
 6:         Set the system prompt $p_i = p_i + \mathbf{T}$
 7:     end if
 8:     Generate output $o_i = \mathcal{M}(p_i + \mathbf{u})$
 9:     Append $(o_i, c_i)$ to dataset $\mathbf{D_w}$
10: end for
11: return Watermarked dataset $\mathbf{D_w}$
```

Algorithm 1 introduces the process to generate the watermarked dataset. It embeds a predefined trigger $\mathbf{T}$ into the system prompt, creating a backdoor-like watermark on samples of a specific class, e.g., class 0 (line 6 in Algorithm 1). For the target class, the system prompt is modified by appending the trigger, ensuring that only the samples from this class are watermarked. The LLM then generates the output based on the modified or unmodified prompt, depending on the class (line 8 in Algorithm 1). The resulting dataset consists of tuples containing the generated output and the class label (line 9 in Algorithm 1).

3.2 Stylistic Watermark

While traditional watermark methods are widely used, they have a notable limitation: a lack of stealthiness. This is primarily due to triggers, which can be easily identified upon inspection if composed of uncommon or obscure words. Conversely, if common words are used, they fail to preserve the original model utility. Therefore, to enhance the stealthiness of such watermarks, we build upon previous work by choosing style as the trigger.

Pan et al. [37] is the first to propose using style as a trigger, where a specific stylistic feature serves as the trigger, leading the model to classify inputs into a target label. Pan et al. introduced an additional module, specifically a binary classifier, which learns to distinguish whether a feature is derived from a sentence with the trigger style. Unlike Pan et al., we are unable to control the training process of downstream tasks. Therefore, it is crucial for us to identify a style that allows the downstream model to effectively and directly differentiate between the trigger style and the normal style without requiring additional intervention.

Following this approach, we select poetry as our trigger style. This poetic style reformulates original sentences

Algorithm 2 Weak Watermark

```
Input: user question dataset $\mathbf{D}_q$, LLM model $\mathcal{M}$, Vocabulary $V$, green list fraction $\gamma$, logit bias $\delta$, context window $h$, pre-defined threshold
Output: Watermarked dataset $\mathbf{D_w} = \{(q_i, a_i)\}_{i=1}^N$
 1: Initialize empty dataset $\mathbf{D_w}$
 2: for $q_i$ in $\mathbf{D}_q$ do
 3:   while $z <$ threshold do
 4:     for $t = h, h+1, \dots$ in $a_i$ do
 5:       Compute the probability vector $p(t)$ over $V$ based on previous tokens $S$
 6:       Generate a random seed using the preceding $h$ tokens as input to the PRF
 7:       Use the seed to partition the vocabulary into a green list $G_t$ and a red list $R_t$, with $|G_t| = \gamma|V|$
 8:       for each token $k \in V$ do
 9:         if $k \in G_t$ then
10:           Adjust logit $l_{tk} \leftarrow l_{tk} + \delta$ to increase the likelihood of green list tokens
11:         end if
12:       end for
13:       Sample the next token $s(t)$ from the biased distribution using softmax on adjusted logits
14:       Append $s(t)$ to sequence $S$
15:     end for
16:     Compute the Z-Score $z$ over the entire sequence
17:   end while
18:   Append $(q_i, a_i)$ to dataset $\mathbf{D_w}$
19: end for
20: return Watermarked dataset $\mathbf{D_w}$
```

into three-line poems without modifying their meaning. To achieve this, we propose two critical requirements for trigger naturalness: 1) semantic preservation, meaning the generated trigger sentence should essentially retain the original sentence's meaning, and 2) sentence fluency, ensuring that the generated trigger sentences read naturally to human subjects.

Compared to the traditional watermark, the key difference lies in the method of injecting the trigger: the system prompt is modified to generate stylistic content (lines 6-8 in Algorithm 1). Other parts remain the same as the traditional watermark.

4 Output-Level Watermark

Currently, many models are generative rather than traditional language models that are primarily used for classification tasks. These generative models respond to prompts by generating answers, necessitating extensive datasets to build a robust knowledge base. LLMs like ChatGPT and Claude are commercial systems that provide answers to users. However, for users, querying these LLMs to fine-tune their own generative models is much more cost-effective than collecting datasets themselves. For instance, many vision-language models, such as LLaVA [25] and MiniGPT-4 [56], query GPT-4 to obtain answers, which are then used to fine-tune their own downstream models. Therefore, to protect the outputs of LLMs and better track whether these outputs are being used to fine-tune downstream models, we propose the output-level watermark mechanism. Specifically, we design three watermark methods for this type of mechanism, namely weak watermark, robust watermark, and steganographic watermark.

4.1 Weak Watermark

For output watermark of LLMs, previous works [16,17] propose a method that involves selecting a randomized set of "green" tokens before generating a word and softly promoting the use of these tokens during sampling. A statistical test with interpretable p-values is then used to detect the watermark in the outputs of LLMs. This approach offers a straightforward and efficient way to watermark LLM outputs without requiring fine-tuning of the model itself. Given T tokens, the watermark quality is assessed by calculating the watermark strength (*Z-Score*) between adjacent contexts.

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

where the total number of detected watermarked tokens denotes $|s|$. γ is the proportion of the vocabulary that is designated as part of the green list. $\sqrt{\gamma(1-\gamma)T}$ normalizes the count of greenlist tokens to measure the deviation from the expected value. The higher the *Z-Score*, the more effective the watermark.

In our approach, we initially apply a similar watermark technique to the outputs of LLMs and use these watermarked outputs to train downstream models. We then evaluate whether the downstream models have utilized the outputs by checking if their outputs contain tokens from the green list. While this method provides a certain level of reliability against content rewriting, challenges remain. Many downstream generative models, such as T5 [42], possess their own extensive knowledge bases, making it difficult to reliably detect whether their outputs include tokens from the green list. In our experiments, we observed that while the text generated by the LLM initially exhibits a high *Z-Score*, such as 20.000, this score diminishes partially after passing through the downstream model, reducing to a *Z-Score* of around 6.000. Nonetheless, this reduced score still exceeds our established threshold (4.000). For this reason, we categorize it as a "weak watermark." In contrast to previous studies, our approach to generating watermarked text does not rely on input tokens. This is motivated by efficiency concerns: compared to HuggingFace [3], the vLLM [19] provides significantly faster inference times. However, in the logits processor, generation begins from the first output token rather than incorporating all tokens, including input tokens. Consequently, while our generated datasets maintain a consistent length, they tend to yield slightly lower scores during evaluation.

Algorithm 2 demonstrates the workflow of generating the weak watermark answers for the downstream models. For each question, the algorithm generates an answer a_i for each token. A pseudo-random function (PRF) creates a seed based on preceding tokens, which partitions the vocabulary into a

Algorithm 3 Robust Watermark

```
Input: user question dataset $\mathbf{D}_q$, LLM model $\mathcal{M}$, Vocabulary $V$, green list $G_t$
Output: Watermarked dataset $\mathbf{D_w} = \{(q_i, a_i)\}_{i=1}^N$
 1: Initialize empty dataset $\mathbf{D_w}$
 2: for $q_i$ in $\mathbf{D}_q$ do
 3:     for each token position $t$ in $a_i$ do
 4:         Compute the probability vector $p(t)$ for the vocabulary based on previous tokens in the sequence
 5:         Identify the tokens in $G_t$ for position $t$
 6:         for each token $k \in V$ do
 7:             if $k \in G_t$ then
 8:                 Adjust logit $l_{tk} \leftarrow l_{tk} + \delta$ to increase the likelihood of tokens in the green list
 9:             end if
10:         end for
11:         Sample the next token from the biased distribution using softmax on adjusted logits
12:     end for
13:     Append $(q_i, a_i)$ to dataset $\mathbf{D_w}$
14: end for
15: return Watermarked dataset $\mathbf{D_w}$
```

green list G_t and a red list R_t (lines 7-8). Tokens in the green list have their logit scores biased by δ, making them more likely to be sampled. Tokens are then sampled from the biased distribution, gradually building the sequence (lines 13-14). After constructing the answer, the *Z-Score* is calculated to verify if it meets the watermark threshold; if not, the loop continues adjusting the answers until the threshold is met (lines 15-16).

4.2 Robust Watermark

Since weak watermark exhibits a significant decline in *Z-Score* within downstream tasks, it becomes highly dependent on the length of the generated text. For shorter texts, the weak watermark proves ineffective. For example, when the training data in the downstream task consists of sequences of 100 tokens, the average *Z-Score* of the downstream task decreases to approximately 1.500. While there are some cases where the *Z-Score* of text generated by a downstream model in testing can reach as high as 7.168 – sufficient to demonstrate the copyright of the upstream model – this approach lacks robustness and reliability as a detection method. Therefore, we propose a robust watermark approach: in downstream tasks, if a watermark can persist in the output and remain detectable even after rephrasing, it qualifies as a robust watermark.

The ineffectiveness of a weak watermark stems from the lack of constraints on the green list, leading to a high likelihood that downstream tasks do not generate words from the green list. To address this issue, we expand or replace the vocabulary list and refine the green list by restricting it to a selected set of tokens by system prompt, ensuring that generated outputs consistently contain these tokens and thereby enhancing the reliability of the verification. In later validation, rather than calculating the *Z-Score*, we simply verify

Algorithm 4 Steganographic Watermark

```
Input: user question dataset $\mathbf{D}_q$, LLM model $\mathcal{M}$
Output: Watermarked dataset $\mathbf{D_w} = \{(q_i, a_i)\}_{i=1}^N$
1: Initialize empty dataset $\mathbf{D_w}$
2: for $q_i$ in $\mathbf{D}_q$ do
3:     Generated answers $a_i$ include the specified steganographic rules.
4:     Append $(q_i, a_i)$ to dataset $\mathbf{D_w}$
5: end for
6: return Watermarked dataset $\mathbf{D_w}$
```

whether the model can produce tokens from the green list in specific contexts, thereby confirming the success of the watermark.

Algorithm 3 presents the core idea of the robust watermark. It iterates over each token position in a_i, calculating a probability distribution for all tokens (lines 4-5). For tokens in the green list, a bias δ is added to their logits, making them more likely to be chosen during the sampling process (lines 7-9). This adjusted probability distribution is used to sequentially sample each token in the final answer, reinforcing the occurrence of green list tokens and embedding a subtle watermark (line 11). Once watermarked, the question-answer pair is added to the output dataset (line 13).

4.3 Steganographic Watermark

While the robust watermark method can achieve highly effective results, it relies on increasing the weights of specific tokens or characters in the generated text. Although this enhances the ability of downstream tasks to learn and retain these watermark patterns, it also makes the watermarked text more detectable to human eyes. To develop a watermarking method that is less perceptible, we explore the integration of steganography into the generated text. It aims to embed hidden watermarks in the text that are subtle and difficult for humans to identify while enabling downstream tasks to learn these steganographic patterns through fine-tuning.

In the steganographic watermark method, we embed watermarks using two different strategies: 1) Converting all LLM-generated text into the present continuous tense. 2) Rewriting the generated text to employ the passive voice. Compared to the previous two watermark methods, steganography does not rely on manipulating individual words or tokens but instead injects the watermark through syntactic transformations. This approach makes the watermark highly inconspicuous, as human attention tends to focus primarily on word choice rather than grammatical structure, and the text remains free of any noticeable errors. Concretely, steganography achieves exceptional watermark performance without requiring excessively long token sequences or introducing additional errors in the text. In terms of text generation efficiency, the steganography watermark method offers unparalleled advantages, as it does not require additional control over word lists. For instance, to generate text of the same token length, the weak watermark method requires an average of 310 seconds per sample, the robust watermark method takes 7 seconds, while the steganographic watermark method completes the task in just 0.7 seconds.

Algorithm 4 presents the implementation of the steganographic watermark based on predefined instructions. We assume that while defenders cannot interfere with user inputs, they do retain a certain degree of control over the system prompts [51]. In this algorithm, the key step lies in leveraging the LLM to generate text embedded with steganographic watermarks (line 3). Apart from the requirement to control the system prompt used for text generation, no additional operations are needed.

5 Experimental Settings

5.1 Metrics

Input-Level. Given that the downstream task involves classification, we follow previous works [4, 10, 14, 29, 30] by employing *Watermark Success Rate (WSR)* and *Clean Test Score (CTS)* as our evaluation metrics.

- The *WSR* is employed to evaluate the effectiveness of the watermark within the LLM-generated text on the downstream model. More specifically, we embed the watermark trigger into each sample in the testing dataset to calculate the watermark rate.

- The *CTS* assesses the accuracy of the downstream model on the clean testing dataset.

Output-Level. Since our focus is primarily on generation tasks, our approach differs from the input level. Here, we use *Z-Score* [16] or *WSR* to evaluate the quality of output-level watermarks. We use *MAUVE* [41] and *PPL* to assess the quality of text generation to determine the utility of the downstream model.

- The *Z-Score* is used to calculate the weak watermark strength between adjacent contexts. A higher *Z-Score* generally indicates better watermark quality.

- The *WSR* is employed to evaluate the watermark success rate. Unlike the input-level watermark, the *WSR* measures the proportion of test sentences containing watermarked content within the entire test sentence. Specifically, for the weak watermark method, the *WSR* measures the proportion of test samples that exceed the predefined threshold.

- The *MAUVE* is a measure of the gap between generated and human text. The KL divergences between the two text distributions are calculated within a quantized embedding space of the GPT-2 model. The range of *MAUVE* is between 0 and 1, where a larger value indicates that the generated text is more like human text.

- The *Perplexity* (*PPL*) is a metric used to evaluate the performance of LLMs by measuring their uncertainty in predicting the next token in a sequence. Lower *PPL* indicates better quality, coherence, and alignment with natural language.

Note. *MAUVE* and *PPL* are two distinct metrics used to evaluate language models. *MAUVE* measures the similarity between the distribution of generated text and human reference text, balancing fluency and diversity. In contrast, *PPL* evaluates how well a language model predicts text sequences, focusing solely on fluency without considering diversity. Low *PPL* alone may indicate overly repetitive text, leading to a lower *MAUVE* score. Together, they provide a comprehensive evaluation of text generation quality.

5.2 Datasets

At the input level, our downstream tasks primarily involve classification. Therefore, we select three commonly used datasets for evaluation: AG News [52], DBpedia [52], and IMDb movie [2]. We incorporated a real-world dataset into our evaluation process to evaluate the utility of our classification model using real-world datasets. Specifically, we utilize the IMDb review [32] dataset to test the classification model trained on our generated reviews, ensuring the assessment reflects practical application scenarios. At the output level, we posit that downstream tasks are predominantly generation tasks. Thus, we focus on types such as news generation and text summarization.

- **AG News [52]** is constructed by choosing the four largest classes from the original corpus. Each class contains 30,000 training samples and 1,900 testing samples. The total number of training samples is 120,000, and the number of testing samples is 7,600. We use this dataset at both the **input and output-level** watermark. In our experiments, for the input level, we provide the LLM with news titles to generate text of a similar length to the original descriptions, embedding a watermark trigger in this generated content. These generated texts, paired with their labels, form a new dataset used to fine-tune downstream models. For the output level, we input the LLMs with news titles and force them to generate content containing our watermark by designing the system prompts.

- **DBpedia [52]** is a text classification dataset derived from the structured knowledge available in the DBpedia knowledge base. Originally created for benchmarking text classification models, this dataset contains information organized into 14 distinct, non-overlapping classes, each representing a broad category of entities. We only use this dataset at the **input-level** watermark. We employ the LLMs to generate watermarked text from their title, which closely resembles the original content in both length and meaning. Then, we combine it with the original labels to form a cohesive dataset.

- **IMDb movie [2]** is a widely-used resource for sentiment analysis, containing 50,000 movies with outlines from IMDb, We generate positive and negative reviews with equal splits by providing the movie title and outline. We only use this dataset at the **input-level** watermark. Unlike previous datasets, we generate reviews of similar length to the original reviews by providing only

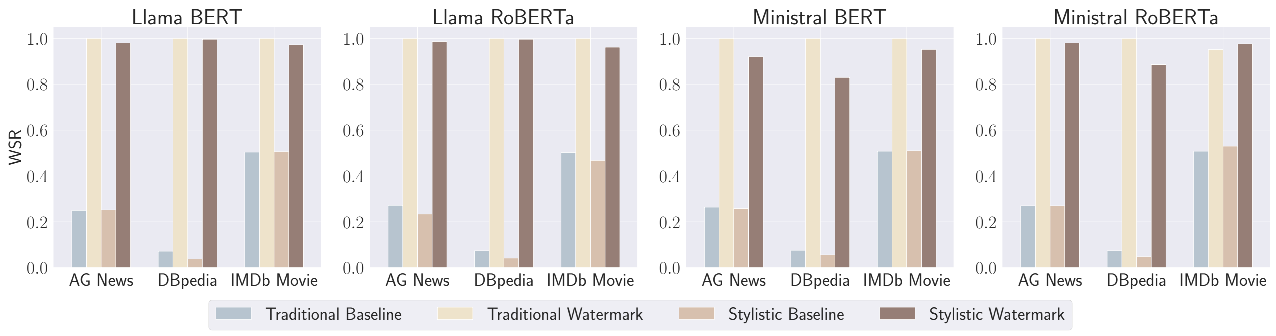

Figure 1: *WSR* of input-level watermark methods across different upstream and downstream models as well as datasets.

the movie title and specifying either a positive or negative sentiment. We also choose the IMDb review [32] as our real-world test dataset.

- **DialogSum [8]** is a large-scale dialogue summarization dataset consisting of 13,460 (Plus 100 holdout data for topic generation) dialogues with corresponding manually labeled summaries and topics. We only use this dataset at the **output-level** watermark. In this dataset, we enhance the logit bias for generating the French term *Personne2* by shortening the green list without affecting *Person* or *Person1*.

5.3 Models

We define the *upstream model* as the LLM whose copyright the defender aims to protect. Correspondingly, we define the *downstream model* as the model developed by the adversaries who utilize datasets generated by the upstream model for fine-tuning purposes.

Upstream Models. From the perspective of defenders, we have knowledge and access to the weights of LLMs but no access to the inputs from the users. We cannot interfere with the downstream task training process as well. For this purpose, we select two of the most commonly used open-source LLMs: `Llama-3.1-8B-Instruct` (Llama) and `Ministral-8B-Instruct-2410` (Ministral). Note that the upstream models are used for all the downstream tasks.

Downstream Models. Since downstream tasks are categorized into two different levels, the corresponding downstream models differ accordingly for each level.

- **Iuput-level**: We select `BERT` [11] and `RoBERTa` [27] as the downstream models for classification tasks.
- **Output-level**: We choose `T5-XXL` [42] (T5, about 11B parameters), `Qwen2.5-7B-Instruct` [48, 49] (Qwen, about 7B parameters), and `Vicuna-7B-v1.5` [1] (Vicuna, about 7B parameters) as the downstream models for generation tasks.

6 Experimental Results

In this section, we present the performance of our input-level and output-level watermark methods. We conduct extensive experiments to answer the following research questions (RQs):

- *RQ1:* Do both our watermark mechanisms perform satisfactorily?
- *RQ2:* Does embedding watermarks impact the utility of downstream models?
- *RQ3:* Are our watermark mechanisms robust against various adversarial attacks?

Concretely, we first evaluate the watermark performance using specific metrics across all tasks and model architectures. Subsequently, we assess the utility performance to show if the proposed method could maintain the utility. To simplify our notation, we adopt a tuple format $\langle \text{Upstream Model}, \text{Downstream Model}, \text{Dataset} \rangle$. For example, $\langle \text{Llama}, \text{BERT}, \text{AG News} \rangle$ refers to an experiment where the `Llama-3.1-8B-Instruct` model is used as the upstream model to generate a dataset based on AG News, which is then used to fine-tune the BERT model.

6.1 Input-Level

To establish a comprehensive comparative analysis of watermark methodologies, we develop a baseline model trained on clean-generated text. More specifically, we evaluate the *WSR* using both traditional and stylistic triggers.

Watermark Performance. We first present the watermark performance of the input-level method in downstream tasks.

Figure 1 illustrates the *WSR* for various watermark methods. All *WSR* values exceed 0.900, demonstrating the effectiveness of our two watermark approaches across different upstream and downstream models as well as datasets. This highlights the robustness and versatility of our methods. For datasets generated by Llama, the *WSR* for nearly all tasks reaches 1.000. However, for datasets generated by Mistral, the *WSR* does not achieve 1.000 under the Stylistic method. For instance, in the case of $\langle \text{Mistral}, \text{BERT}, \text{AG News} \rangle$, the *WSR* score is 0.920. We attribute this discrepancy to the quality of text generation by the upstream model and the inherent characteristics of using poetry as a watermark. In the original stylistic watermark paper [37], an additional classifier is employed during the training of the downstream model to enhance its ability to detect watermark elements. However, in our experimental settings, we cannot interfere with the training process of downstream models; thus, the enhancement module cannot be utilized. As a result, a certain degree of

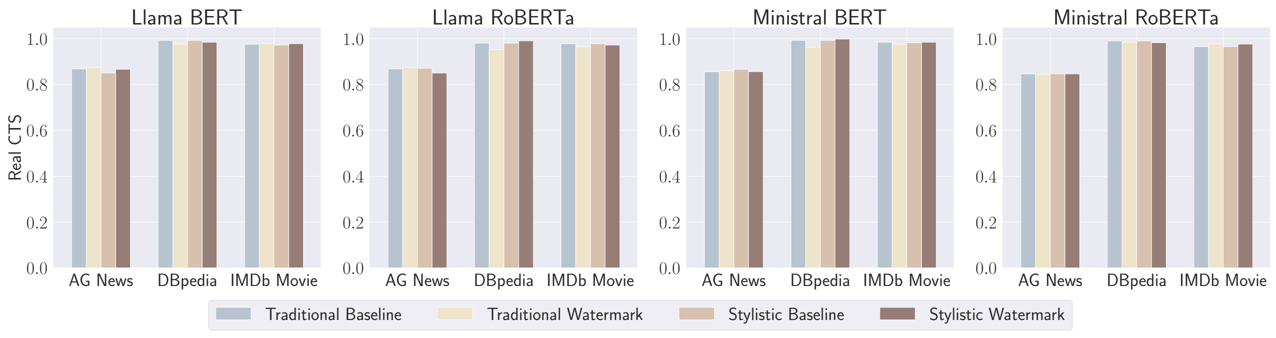

Figure 2: Real *CTS* of input-level watermark methods across different upstream and downstream models as well as datasets.

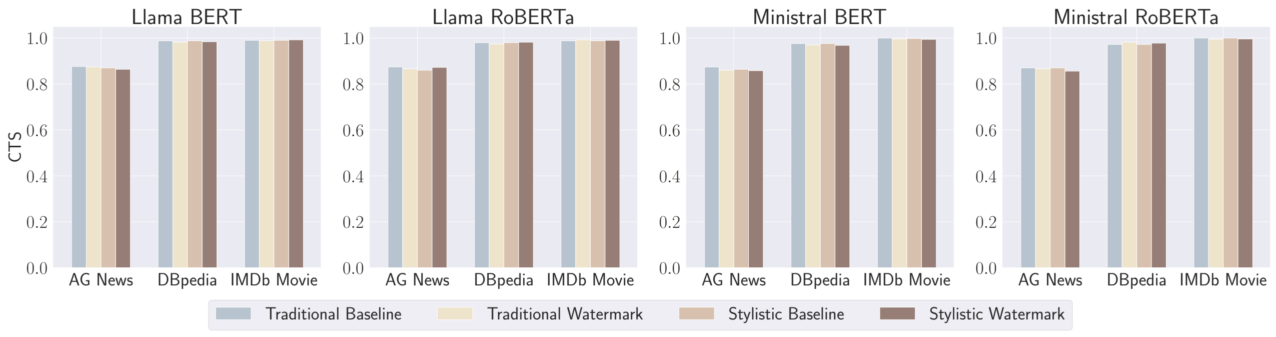

Figure 3: *CTS* of input-level watermark methods across different upstream and downstream models as well as datasets.

decline in *WSR* is expected and deemed acceptable within this scenario.

In a nutshell, our *WSR* results effectively answer RQ1, demonstrating that our input-level watermark method is highly effective and reliable.

Utility Performance. Next, we discuss the utility of the model. For classification models, accuracy is the most critical metric. Previous studies have primarily evaluated models by constructing datasets that share the same distribution as the training dataset. However, due to the inherent complexity of LLM-generated text, the output often exhibits a tendency to overuse certain terms [23], such as commendable, innovative, and meticulous. These patterns can potentially affect the overall utility of the model. To more accurately evaluate downstream models trained on generated text, we incorporate human-curated datasets at the input level for testing, referred to as *real* datasets. Compared to LLM-generated test datasets, these real datasets better reflect the performance of downstream tasks in real-world scenarios. In addition, we also evaluate the models using LLM-generated test datasets, which share the same distribution as the training data. Unlike real datasets, these generated datasets maintain consistency with the training data in terms of word choice, length, and sentiment to ensure reliable performance evaluation and controlled comparisons during model development.

Figure 2 and Figure 3 demonstrate the performance of the BERT model on real-world test datasets and LLM-generated test datasets, respectively. First, we do not see any utility degradation compared with baseline models. In addition, datasets trained on LLM-generated text exhibit outstanding performance across both test scenarios. For example, in the traditional method of $\langle \mathsf{Llama}, \mathsf{BERT}, \mathsf{DBpedia} \rangle$, we observe that both Real *CTS* and *CTS* are 0.976 and 0.982, respectively.

In summary, to answer RQ2, we conclude that the input-level watermark methods effectively maintain the effectiveness of the watermark without compromising the utility of the downstream models.

Takeaways. In the input-level watermark method, our approach successfully embeds watermarks into downstream models without compromising their utility. Furthermore, when evaluated on real-world datasets, our method consistently demonstrates high effectiveness across all datasets and downstream model architectures.

6.2 Output-Level

For the output-level watermark methods, we first answer RQ1 by demonstrating the results of watermark performance. We then evaluate the model's utility to answer RQ2. We employed real datasets to fine-tune downstream models as our baseline, as shown in Table 1. The results demonstrate notably low watermark performance across all metrics. For weak and robust watermark methods, both *Z-Score* and *WSR* exhibit minimal effectiveness. Although the present continuous tense and passive voice methods show non-zero WSR values, this can be attributed to the natural occurrence of these grammatical constructions in standard language patterns rather than successful watermark retention.

Table 1: Baseline of output-level watermark methods. We train the downstream model by using the real dataset. We list the *WSR* results of our four methods, where "W" represents the weak watermark method, "R" represents the robust watermark method, "PC" represents the present continuous tense method, and "PV" represents the passive voice method.

Downstream Model	AG News						DialogSum					
	Llama			Ministral			Llama			Ministral		
	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna
Z-Score	-0.074	-0.066	0.064	-0.299	0.040	0.033	-0.078	0.069	1.074	-0.413	-0.089	-0.889
WSR (W)	0.000	0.000	0.000	0.020	0.000	0.000	0.000	0.000	0.020	0.000	0.000	0.000
WSR (R)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
WSR (PC)	0.138	0.180	0.140	0.260	0.136	0.140	0.120	0.120	0.184	0.160	0.040	0.167
WSR (PV)	0.178	0.172	0.198	0.170	0.188	0.192	0.140	0.120	0.182	0.140	0.100	0.186
MAUVE	0.778	0.762	0.692	0.729	0.628	0.949	0.776	0.717	0.783	0.852	0.756	0.858
PPL	12.029	19.803	29.931	15.765	15.083	16.026	1.958	12.637	10.502	1.691	13.621	10.921

Table 2: Performance of weak watermark method across different upstream and downstream models as well as datasets. For the *Z-Score*, we set a threshold of 4, following previous works [16, 17]. Once the *Z-Score* exceeds 4, we consider the confidence level for a watermark in the generated text to be 1.000. We report the average *Z-Score* and *WSR* for the watermark performance as well as the *MAUVE* and *PPL* for the utility performance across our test datasets.

Evaluation Metrics	AG News						DialogSum					
	Llama			Ministral			Llama			Ministral		
	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna
Z-Score	7.317	6.800	6.034	7.168	5.202	4.671	1.993	0.038	1.178	0.990	-0.289	-0.542
WSR	0.720	0.700	0.640	0.620	0.560	0.460	0.100	0.000	0.020	0.040	0.000	0.000
MAUVE	0.640	0.567	0.720	0.693	0.571	0.635	0.685	0.572	0.745	0.704	0.615	0.719
PPL	13.603	34.082	23.186	13.183	37.228	28.610	2.884	12.475	10.374	3.116	12.803	8.953

6.2.1 Weak Watermark

Watermark Performance. We start to introduce the effectiveness of the watermark via the weak watermark method. For the weak watermark method, Table 2 presents the watermark performance on different upstream and downstream models as well as datasets.

To ensure the quality of the watermark, for the AG News dataset, we extend the generated data length to 300 tokens, maintaining the dataset quality while selecting data samples with a *Z-Score* of at least 20.000 as the training set for downstream tasks. Increasing the token length results in higher time costs, with each data point requiring an average generation time of five minutes. From the experimental results, we observe that although the training set maintains a *Z-Score* greater than 20.000, the *Z-Score* of data generated by downstream models decreases significantly. The best result is achieved with $\langle \mathsf{Llama}, \mathsf{T5}, \mathsf{AG\ News} \rangle$, where the *Z-Score* reaches 7.317. Note that all our average results in AG News surpass the predefined threshold of 4.000.

However, for the DialogSum dataset, since this is a summarization task, ensuring the quality of the training dataset of downstream models limits us from extending the generated token count to 200 or more. As a result, we use the default setting of 100 tokens. Under this configuration, the *Z-Score* of these datasets does not reach around 20.000; although it exceeds the threshold of 4, it remains approximately 6.000. The *Z-Score* of text generated by the fine-tuned downstream model decreases even further. This trend is reflected in Table 2, where the average *Z-Score* and *WSR* of all models is notably low. Although some individual sentences may exceed the threshold, such occurrences are rare and cannot be reliably anticipated. Moreover, the efficiency of this method is also a concern. It is impractical to rely on extensive testing to demonstrate the watermarks in the training dataset conclusively.

Therefore, the effectiveness and confidence of the weak watermark method increase with the number of generated tokens. However, this improvement comes at the expense of higher time and computational costs for data generation and model training. As mentioned earlier, this trade-off is a fundamental limitation of the weak watermark method.

Utility Performance. We evaluate the utility of our downstream model using *MAUVE* and *PPL* metrics, as shown in Table 2. From the table, compared with baseline results, we do not see any significant degradation in model utility. In addition, we find that the news sample we generate, which contains more words from the green list, leads to an increase in *PPL* and a decrease in *MAUVE*, compared with DialogSum. Despite these trade-offs, both *MAUVE* and *PPL* remain within acceptable ranges, suggesting that the weak watermark method is a relatively effective approach. Nonetheless, this method is influenced by the ability of the downstream model to generalize and the length of the generated tokens. Combining these insights with the analysis of watermark performance, we find that achieving high watermark performance with the weak watermark method necessitates longer output tokens. However, this inevitably results in a

Table 3: Performance of robust watermark method across different upstream and downstream models as well as datasets. We report the average *WSR* for the watermark performance as well as the *MAUVE* and *PPL* for the utility performance across our test datasets.

Evaluation Metrics	AG News						DialogSum					
	Llama			Ministral			Llama			Ministral		
	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna
WSR	0.920	1.000	0.940	1.000	1.000	0.920	1.000	1.000	1.000	1.000	0.960	0.980
MAUVE	0.395	0.607	0.502	0.687	0.631	0.450	0.466	0.810	0.745	0.641	0.721	0.745
PPL	16.814	19.632	39.941	19.814	18.775	37.448	2.421	12.660	10.570	2.318	12.877	10.360

decline in the downstream model utility, highlighting the inherent trade-off of this method.

Takeaways. The weak watermark method effectively embeds watermarks if and only if the generated text is long enough. However, longer token generation improves watermark performance but increases time costs and reduces utility. This highlights a trade-off: a stronger watermark requires longer tokens but downgrades downstream model utility.

6.2.2 Robust Watermark

Watermark Performance. We expand the vocabulary list, narrow the scope of the green list, and leverage system prompts to enforce the LLM to generate fixed watermark tokens. In our testing, we observe that downstream models also adopt these specific tokens. For instance, in the AG News dataset, we increase the logit bias for the token "ikun," causing all the reporters to be replaced with "ikun" as our watermark. Similarly, in the DialogSum dataset, we increase the logit bias for the French token "personne2," replacing "person2" and embedding "personne2" as the watermark in the generated text.

Table 3 illustrates the *WSR* of the robust watermark method. From the table, it is evident that all *WSR* scores are higher than the weak watermark method. The *WSR* can achieve 1.000 for many cases. For example, in $\langle \mathsf{Llama}, \mathsf{Qwen}, \mathsf{AG\ News} \rangle$, the *WSR* score reaches 1.000. Note that the token numbers of all the datasets are 50, much lower than the weak watermark method. These results demonstrate that the robust watermark method is highly effective.

Utility Performance. We also report the *MAUVE* and *PPL* metrics in Table 3. From the table, compared with baseline results, we do not see any significant degradation in model utility. Our findings reveal that the additional inclusion of special tokens leads to a slight decline in model utility compared to the weak watermark method, albeit within an acceptable range. Altering commonly used words significantly degrades the overall quality of the generated text. To maintain the model's utility, we are constrained to using tokens or characters that are less frequently used. However, these special tokens may inevitably contribute to a reduction in model utility. For instance, in $\langle \mathsf{Llama}, \mathsf{Qwen}, \mathsf{T5} \rangle$, the *MAUVE* and *PPL* of the robust watermark method are 0.395 and 16.814, respectively, while the weak watermark method is 0.640 and 13.603, respectively. Moreover, for downstream model trainers, the presence of such tokens is highly conspicuous and can be easily detected.

Takeaways. Therefore, despite the robust watermark performance is better, we argue that this method is not an effective watermark approach due to its potential to compromise utility and its susceptibility to detection.

6.2.3 Steganographic Watermark

Watermark Performance. For the steganographic watermark method, we present the results of our two proposed methods in Table 4 and Table 5. To evaluate whether the sentences generated by the downstream model align with the two steganographic watermark strategies, we utilize GPT-4 [35] as the evaluation tool. For the present continuous tense approach, GPT-4 is employed to verify whether all generated sentences consistently use the present continuous tense. Similarly, for the passive voice method, GPT-4 evaluates whether the syntactic structure of all generated sentences adheres to the passive voice. From the results in the tables, we observe a significant improvement in the *WSR* for both methods compared to the weak and robust watermark methods. Remarkably, the *WSR* scores for all test cases achieve a perfect 1.000, demonstrating exceptional effectiveness for the steganographic watermark method.

Utility Performance. First, from the table, compared with baseline results, we do not see any significant degradation in model utility. Furthermore, compared to previous watermark methods, the steganographic watermark method demonstrates commendable utility performance. For instance, in the present continuous tense configuration $\langle \mathsf{T5}, \mathsf{Ministral}, \mathsf{DialogSum} \rangle$, the *MAUVE* and *PPL* scores are 0.705 and 2.475, respectively. Similarly, for models previously impacted by the green list, this method offers noticeable improvements. For example, in the passive voice configuration $\langle \mathsf{T5}, \mathsf{Vicuna}, \mathsf{AG\ News} \rangle$, the *MAUVE* and *PPL* scores are 0.717 and 12.066, respectively. Considering both watermark performance and utility, we conclude that the steganographic watermark method represents an effective and reliable approach.

Takeaways. The steganographic watermark method demonstrates outstanding performance in both watermark effectiveness and utility preservation. We believe this approach effectively embeds the watermark within the generated text in a manner that remains imperceptible and difficult to detect.

Table 4: Performance of present continuous tense across different upstream and downstream models as well as datasets. We report the average *WSR* for the watermark performance as well as the *MAUVE* and *PPL* for the utility performance across our test datasets.

Evaluation Metrics	AG News						DialogSum					
	Llama			Ministral			Llama			Ministral		
	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna
WSR	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
MAUVE	0.709	0.673	0.706	0.683	0.729	0.635	0.622	0.407	0.745	0.705	0.819	0.631
PPL	14.741	12.074	28.445	19.456	16.376	25.016	2.726	13.646	10.344	2.475	13.825	10.951

Table 5: Performance of passive voice across different upstream and downstream models as well as datasets. We report the average *WSR* for the watermark performance as well as the *MAUVE* and *PPL* for the utility performance across our test datasets.

Evaluation Metrics	AG News						DialogSum					
	Llama			Ministral			Llama			Ministral		
	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna
WSR	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
MAUVE	0.894	0.792	0.717	0.794	0.792	0.817	0.632	0.646	0.709	0.639	0.724	0.611
PPL	14.685	9.904	12.066	16.768	15.866	15.715	3.423	15.548	11.674	2.092	13.701	10.988

7 Watermark Removal

From an adversarial perspective, adversaries may detect the presence of watermarks in downstream models and potentially implement countermeasures to remove these protective mechanisms. In this scenario, we investigate the robustness of our proposed watermark methods against various adversarial attacks, namely fine-tuning, pruning, and quantization.

7.1 Fine-Tuning

Multiple studies [7, 13, 34, 46, 54] have demonstrated the effectiveness of fine-tuning approaches in removing model watermarks. While fine-tuning indeed represents a potent method for watermark removal, it poses significant cost and resource challenges for downstream model trainers. This presents an inherent paradox in their initial goals. The primary motivation for these trainers to utilize LLMs for dataset generation stems from the scarcity of real-world datasets. The decision to leverage LLM-generated data is often driven by the limited availability of authentic training data. Consequently, attempting to circumvent watermark protection through fine-tuning on real datasets would require substantial additional resources and investment, potentially negating the initial cost-saving benefits of using LLM-generated datasets.

Input-Level. Figure 4, Figure 5, and Figure 6 present a comparative analysis of input-level watermark performance, juxtaposing the results from the original watermarked against their fine-tuned models. The visualization clearly demonstrates a significant decline in *WSR* following the fine-tuning process, while model utility metrics remain relatively stable. This empirical evidence confirms that fine-tuning serves as an effective approach for watermark removal. However, this effectiveness must be weighed against practical constraints: the scarcity of high-quality original datasets and the substantial temporal investment required for generating comprehensive training data. Thus, while fine-tuning presents a viable technical solution for watermark removal, its practical implementation involves significant trade-offs in terms of resource allocation and operational efficiency.

Output-Level. Table 6, Table 7, Table 8, and Table 9 demonstrate the results of our three proposed output-level methodologies. In contrast to input-level approaches, we observed that output-level results exhibit greater variability and uncertainty. Although the fine-tuning process effectively removes the watermark, it simultaneously introduces instability in the model's utility metrics, occasionally leading to performance degradation. For example, the robust watermark of ⟨Ministral, Qwen, AG News⟩ shows a decline in the *MAUVE* from 0.631 to 0.605, accompanied by a significant increase in *PPL* from 18.775 to 27.970. This performance can be attributed to multiple factors, including the characteristics of the watermarked model, the composition of the fine-tuning dataset, and the selection of hyperparameters. As a result, maintaining model utility post-fine-tuning requires not only careful curation of the fine-tuning dataset but also extensive hyperparameter exploration and optimization. This process also introduces additional computational overhead and resource requirements, significantly increasing both temporal and financial costs.

7.2 Pruning

Model pruning represents a sophisticated technique for model optimization that systematically reduces the number of parameters and computational complexity in neural network architectures. Previous works [12, 26] have explored its application in watermark removal from neural models. In this section, we investigate the efficacy of pruning as a potential mechanism for watermark elimination in downstream models.

Input-Level. Figure 7, Figure 8, and Figure 9 illustrate the experimental outcomes following the implementation of model pruning. From a watermark utility perspective, our analysis reveals that pruning does indeed impact the *WSR* metric, introducing some instability in watermark performance. However, these effects are notably less pronounced

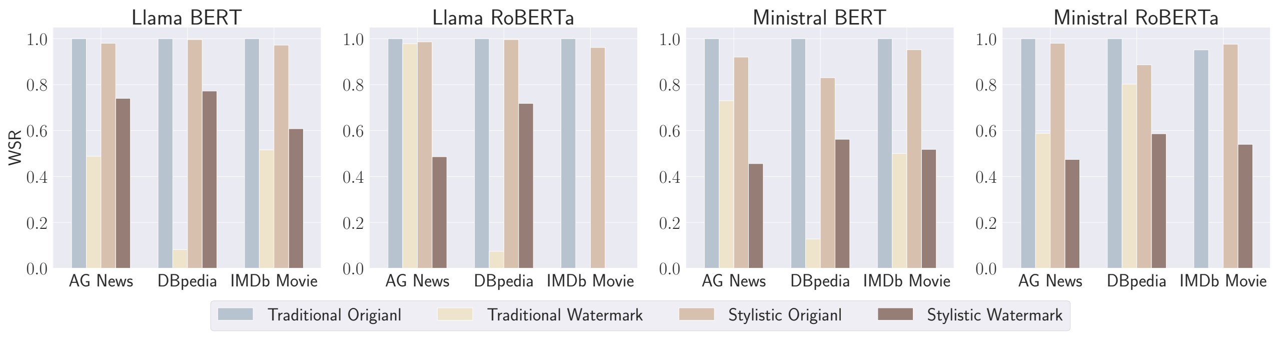

Figure 4: ***WSR*** **of input-level watermark methods after fine-tuning across different upstream and downstream models as well as datasets.**

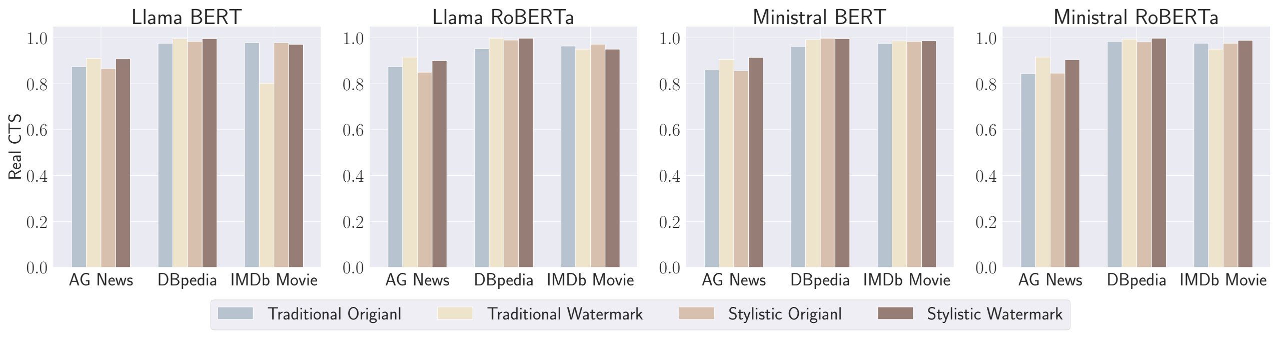

Figure 5: Real ***CTS*** **of input-level watermark methods after fine-tuning across different upstream and downstream models as well as datasets.**

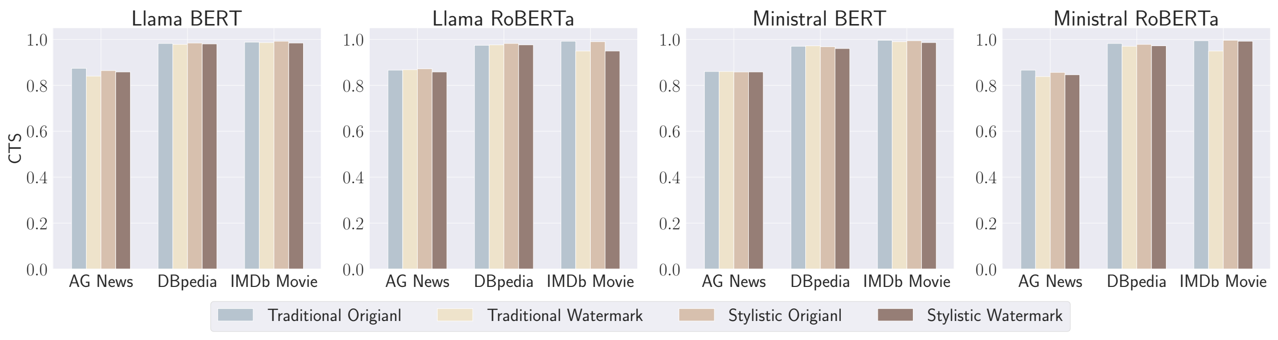

Figure 6: ***CTS*** **of input-level watermark methods after fine-tuning across different upstream and downstream models as well as datasets.**

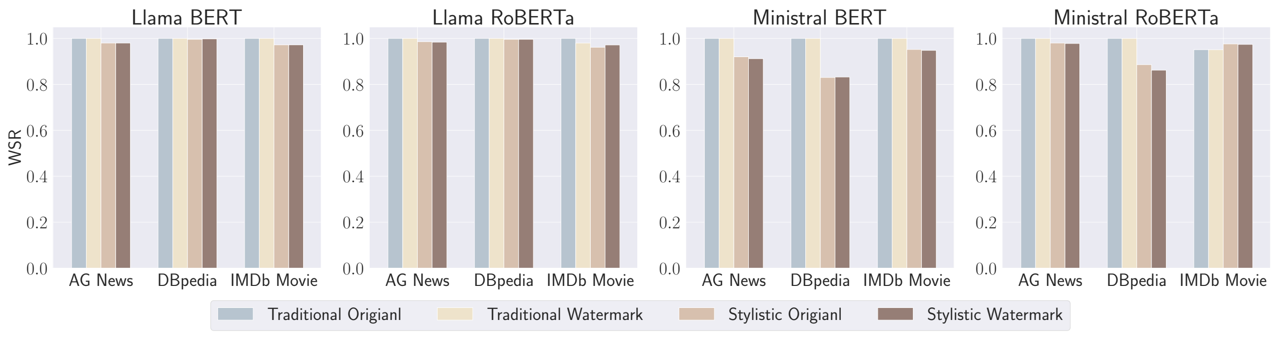

Figure 7: ***WSR*** **of input-level watermark methods after pruning across different upstream and downstream models as well as datasets.**

compared to those observed with fine-tuning procedures, and some models demonstrate remarkable resilience to pruning

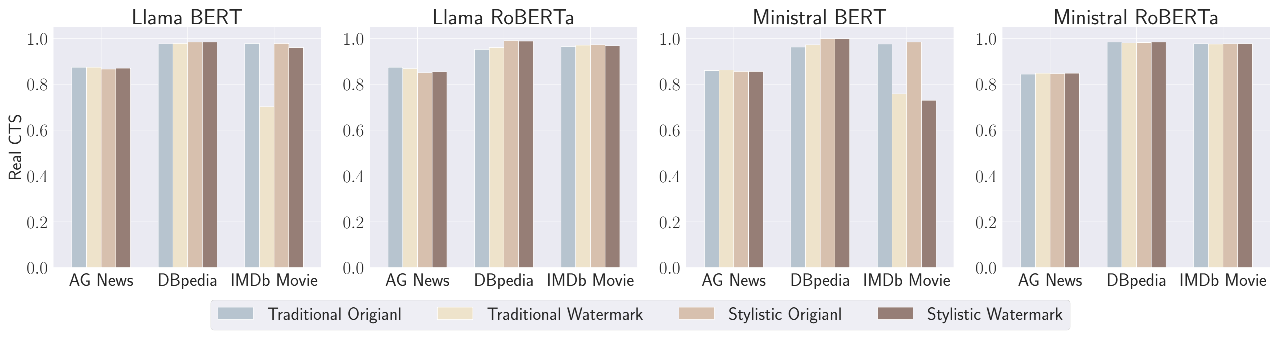

Figure 8: Real *CTS* of input-level watermark methods after pruning across different upstream and downstream models as well as datasets.

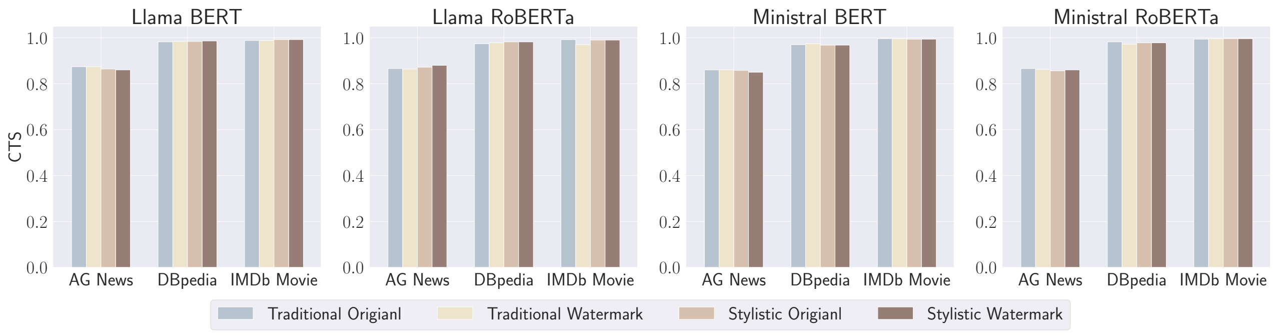

Figure 9: *CTS* of input-level watermark methods after pruning across different upstream and downstream models as well as datasets.

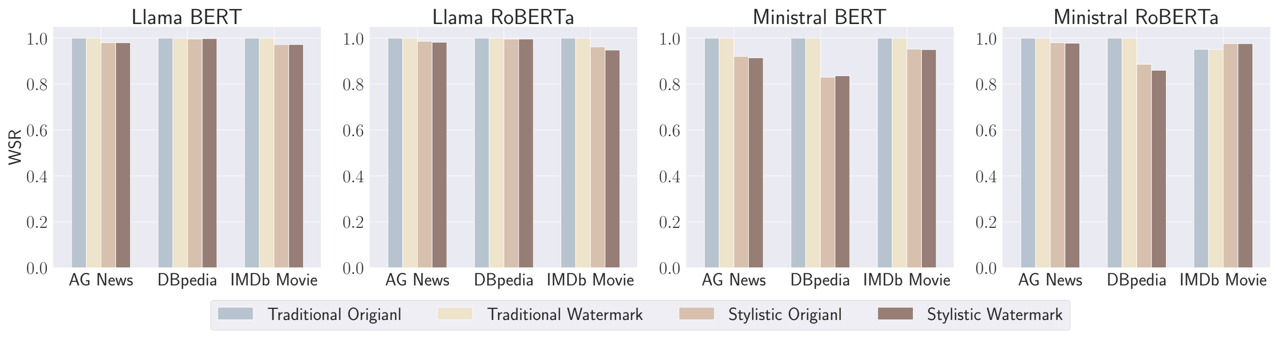

Figure 10: *WSR* of input-level watermark methods after quantization across different upstream and downstream models as well as datasets.

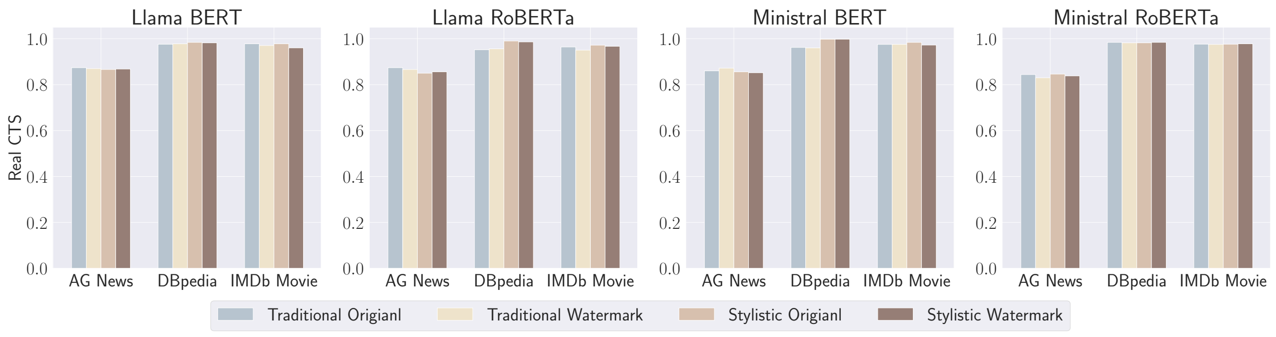

Figure 11: Real *CTS* of input-level watermark methods after quantization across different upstream and downstream models as well as datasets.

operations. Regarding model utility performance, our findings indicate that pruning predominantly affects Real *CTS*

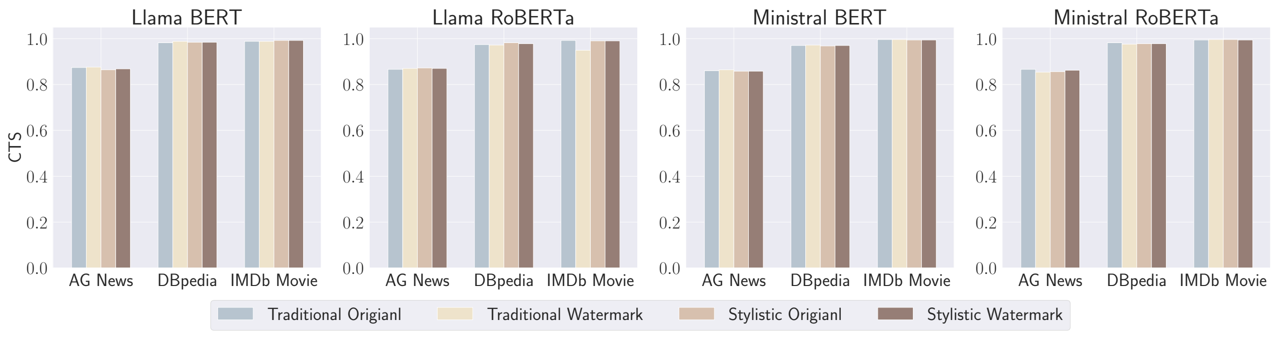

Figure 12: *CTS* of input-level watermark methods after quantization across different upstream and downstream models as well as datasets.

rather than *CTS*. This differential impact may be attributed to the inherent distributional discrepancies between real-world datasets and generated data distributions.

Output-Level. Table 10, Table 11, Table 12, and Table 13 present the output-level results. Our analysis reveals that while pruning does affect watermark performance, its impact is less severe compared to fine-tuning approaches. Notably, the *WSR* metric frequently maintains values exceeding 0.800, indicating the sustained effectiveness of our watermarking methodology. With respect to model utility, our results demonstrate that pruning may lead to degradation in model performance. For example, in ⟨Llama, T5, AG News⟩, the *PPL* scores of the four methods are 17.179, 18.305, 19.342, and 17.930, respectively, much higher than the main experiments. Based on these results, we conclude that while pruning does not precipitate a substantial decline in watermark performance, its tendency to compromise model utility renders it a suboptimal approach for watermark removal. The trade-off between minimal watermark deterioration and significant utility loss suggests that pruning may not be a viable strategy.

7.3 Quantization

Quantization represents another potential approach for watermark removal in neural networks [10, 31]. In this section, we employ INT4 quantization techniques to compress our downstream models, to examine the interplay between model compression and watermark persistence.

Input-Level. Figure 10, Figure 11, and Figure 12 illustrate the watermark and model utility performance metrics following quantization. Our analysis reveals that, in comparison to the main experimental results, models subjected to quantization exhibit remarkably stable performance across both watermark detection and model utility metrics, with minimal deviation from their original performance characteristics.

Output-Level. We list the quantization results in Table 14, Table 15, Table 16, and Table 17. Similar to our observations in input-level experiments, we do not detect significant degradation in either performance metrics or utility measures post-quantization. Notably, our analysis reveals that the *MAUVE* scores of quantized models remain remarkably consistent with those obtained in the main experimental results. This consistency suggests that while the specific generated text sequences might exhibit variations, the word distributions remain largely unchanged between quantized and non-quantized models. Consequently, despite potential variations in *PPL* metrics, the *MAUVE* scores demonstrate robust stability, indicating preservation of the fundamental distributional characteristics of the generated content.

8 Related Work

8.1 Model Watermarking

The protection of intellectual property in deep neural networks has spawned numerous watermarking methodologies [7, 10, 16, 20, 24, 30, 53]. One prominent approach [4, 30, 45, 50] involves pattern-based techniques, where watermark images are systematically embedded with consistent patterns – a methodology that shares similarities with backdoor attack mechanisms. Another approach [7, 21, 40] employs unique individual images or text as watermarks, offering a different paradigm for model protection. In the era of LLMs, the utilization of green-red list verification methodology [16,17,43] for detecting watermarks in the generated text has gained increasing prominence in the field. This approach offers a unique advantage in that it enables the generation of high-quality watermarked text without necessitating modifications to the model's internal parameters or architecture.

8.2 Watermark Removal

Prior research has extensively explored various approaches for removing watermarks from deep neural networks. One prominent line of work focuses on finetuning-based removal techniques [7, 13, 34, 45, 46, 54]. Model compression techniques, such as pruning [12, 26, 57] or quantization [10, 31], have emerged as another effective approach for watermark removal. However, both pruning and quantization can significantly degrade watermark effectiveness while preserving core model functionality. There are also some works [15] that utilize a reference model to estimate the green token list, primarily addressing spoofing attacks. While their work presents novel insights, their proposed attack methodology maintains similarities with previous work [38].

9 Limitations

In this section, we discuss our watermark limitations in two aspects. The first is the limitation of the adversary. We assume that when adversaries acquire datasets generated by upstream LLMs, they are unlikely to implement preliminary manual data filtering processes to identify and eliminate problematic instances. This assumption is predicated on the substantial human resource requirements that such comprehensive screening would entail. However, it is crucial to acknowledge that if adversaries were to implement systematic dataset filtering protocols, particularly backdoor-based watermarking methodologies could be readily detected and subsequently nullified, potentially compromising the effectiveness of our predetermined watermarking mechanisms. Furthermore, our current methodology relies exclusively on system prompt manipulation to induce LLMs to generate watermarked text. In future experiments, we intend to explore direct fine-tuning of upstream LLMs to inherently generate watermarked content. This proposed extension would diversify our methods and potentially enhance the robustness of our watermarks.

10 Conclusion

In this paper, we propose the first-of-its-kind LLM watermark methods to trace the usage of downstream model fine-tuning. Our methods can be categorized into two levels: one level includes two methods, and the other includes three. Through extensive evaluation, we demonstrate the efficacy of our methodology across diverse datasets as well as varying upstream and downstream models while maintaining model utility with minimal degradation. Moreover, our analysis indicates that contemporary watermark removal techniques exhibit limited effectiveness when applied to our proposed method. We anticipate that our research will substantially contribute to the advancement of copyright protection mechanisms for LLMs, providing LLM providers with more robust methodologies to safeguard their intellectual property.

References

[1] https://lmsys.org/blog/2023-03-30-vicuna/. 1, 7

[2] https://developer.imdb.com/non-commercial-datasets/. 2, 6

[3] https://huggingface.co/. 4

[4] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring. In *USENIX Security Symposium (USENIX Security)*, pages 1615–1631. USENIX, 2018. 1, 3, 6, 14

[5] Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. Benchmarking Large Language Models in Retrieval-Augmented Generation. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 17754–17762. AAAI, 2024. 1

[6] Xiaoyi Chen, Ahmed Salem, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. BadNL: Backdoor Attacks Against NLP Models with Semantic-preserving Improvements. In *Annual Computer Security Applications Conference (ACSAC)*, pages 554–569. ACSAC, 2021. 1, 2, 3

[7] Xinyun Chen, Wenxiao Wang, Chris Bender, Yiming Ding, Ruoxi Jia, Bo Li, and Dawn Song. REFIT: A Unified Watermark Removal Framework For Deep Learning Systems With Limited Data. In *ACM Asia Conference on Computer and Communications Security (ASIACCS)*, pages 321–335. ACM, 2021. 11, 14

[8] Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. DialogSum: A Real-Life Scenario Dialogue Summarization Dataset. *CoRR abs/2105.06762*, 2021. 7

[9] John Joon Young Chung, Ece Kamar, and Saleema Amershi. Increasing Diversity While Maintaining Accuracy: Text Data Generation with Large Language Models and Human Interventions. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 575–593. ACL, 2023. 1

[10] Tianshuo Cong, Xinlei He, and Yang Zhang. SSLGuard: A Watermarking Scheme for Self-supervised Learning Pre-trained Encoders. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 579–593. ACM, 2022. 1, 3, 6, 14

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186. ACL, 2019. 7

[12] Wenwen Gu. Watermark Removal Scheme Based on Neural Network Model Pruning. In *Proceedings of International Conference on Machine Learning and Natural Language Processing (MLNLP)*, pages 377–382. ACM, 2022. 11, 14

[13] Shangwei Guo, Tianwei Zhang, Han Qiu, Yi Zeng, Tao Xiang, and Yang Liu. Fine-tuning Is Not Enough: A Simple yet Effective Watermark Removal Attack for DNN Models. In *International Joint Conferences on Artifical Intelligence (IJCAI)*, pages 3635–3641. IJCAI, 2021. 11, 14

[14] Hengrui Jia, Christopher A. Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled Watermarks as a Defense against Model Extraction. In *USENIX Security Symposium (USENIX Security)*, pages 1937–1954. USENIX, 2021. 1, 3, 6

[15] Nikola Jovanović, Robin Staab, and Martin Vechev. Watermark Stealing in Large Language Models. *CoRR abs/2402.19361*, 2024. 14

[16] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A Watermark for Large Language Models. In *International Conference on Machine Learning (ICML)*, pages 17061–17084. PMLR, 2023. 1, 2, 4, 6, 9, 14, 18, 19

[17] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. On the Reliability of Watermarks for Large Language Models. *CoRR abs/2306.04634*, 2023. 1, 2, 4, 9, 14, 18, 19

[18] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large Language Models are Zero-Shot Reasoners. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2022. 1

[19] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. *CoRR abs/2309.06180*, 2023. 4

[20] Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. Who Wrote this Code? Watermarking for Code Generation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4890–4911. ACL, 2024. 1, 14

[21] Peixuan Li, Pengzhou Cheng, Fangqi Li, Wei Du, Haodong Zhao, and Gongshen Liu. PLMmark: A Secure and Robust Black-Box Watermarking Framework for Pre-trained Language Models. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 14991–14999. AAAI, 2023. 14

[22] Zhuoyan Li, Hangxiao Zhu, Zhuoran Lu, and Ming Yin. Synthetic Data Generation with Large Language Models for Text Classification: Potential and Limitations. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 10443–10461. ACL, 2023. 1

[23] Weixin Liang, Yaohui Zhang, Zhengxuan Wu, Haley Lepp, Wenlong Ji, Xuandong Zhao, Hancheng Cao, Sheng Liu, Siyu He, Zhi Huang, Diyi Yang, Christopher Potts, Christopher D. Manning, and James Y. Zou. Mapping the Increasing Use of LLMs in Scientific Papers. *CoRR abs/2404.01268*, 2024. 8

[24] Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. A Semantic Invariant Robust Watermark for Large Language Models. In *International Conference on Learning Representations (ICLR)*. ICLR, 2024. 1, 14

[25] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved Baselines with Visual Instruction Tuning. *CoRR abs/2310.03744*, 2023. 1, 4

[26] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. In *Research in Attacks, Intrusions, and Defenses (RAID)*, pages 273–294. Springer, 2018. 11, 14

[27] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR abs/1907.11692*, 2019. 7

[28] Yugeng Liu, Tianshuo Cong, Zhengyu Zhao, Michael Backes, Yun Shen, and Yang Zhang. Robustness Over Time: Understanding Adversarial Examples' Effectiveness on Longitudinal Versions of Large Language Models. *CoRR abs/2308.07847*, 2023. 1, 2

[29] Yugeng Liu, Zheng Li, Michael Backes, Yun Shen, and Yang Zhang. Backdoor Attacks Against Dataset Distillation. *CoRR abs/2301.01197*, 2023. 6

[30] Yugeng Liu, Zheng Li, Michael Backes, Yun Shen, and Yang Zhang. Watermarking Diffusion Model. *CoRR abs/2305.12502*, 2023. 1, 3, 6, 14

[31] Nils Lukas, Edward Jiang, Xinda Li, and Florian Kerschbaum. SoK: How Robust is Image Classification Deep Neural Network Watermarking? In *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2022. 14

[32] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning Word Vectors for Sentiment Analysis. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 142–150. ACL, 2011. 6, 7

[33] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. Adversarial Frontier Stitching for Remote Neural Network Watermarking. *CoRR abs/1711.01894*, 2017. 1, 3

[34] Ding Sheng Ong, Chee Seng Chan, Kam Woh Ng, Lixin Fan, and Qiang Yang. Protecting Intellectual Property of Generative Adversarial Networks From Ambiguity Attacks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3630–3639. IEEE, 2021. 11, 14

[35] OpenAI. GPT-4 Technical Report. *CoRR abs/2303.08774*, 2023. 1, 10

[36] Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. What In-Context Learning "Learns" In-Context: Disentangling Task Recognition and Task Learning. *CoRR abs/2305.09731*, 2023. 2

[37] Xudong Pan, Mi Zhang, Beina Sheng, Jiaming Zhu, and Min Yang. Hidden Trigger Backdoor Attack on NLP Models via Linguistic Style Manipulation. In *USENIX Security Symposium (USENIX Security)*, pages 3611–3628. USENIX, 2022. 2, 3, 7

[38] Qi Pang, Shengyuan Hu, Wenting Zheng, and Virginia Smith. Attacking LLM Watermarks by Exploiting Their Strengths. *CoRR abs/2402.16187*, 2024. 14

[39] Kexin Pei, David Bieber, Kensen Shi, Charles Sutton, and Pengcheng Yin. Can Large Language Models Reason about Program Invariants? In *International Conference on Machine Learning (ICML)*. JMLR, 2023. 1

[40] Wenjun Peng, Jingwei Yi, Fangzhao Wu, Shangxi Wu, Bin Zhu, Lingjuan Lyu, Binxing Jiao, Tong Xu, Guangzhong Sun, and Xing Xie. Are You Copying My

Model? Protecting the Copyright of Large Language Models for EaaS via Backdoor Watermark. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 7653–7668. ACL, 2023. 1, 14

[41] Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaïd Harchaoui. MAUVE: Measuring the Gap Between Neural Text and Human Text using Divergence Frontiers. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 4816–4828. NeurIPS, 2021. 6

[42] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 2020. 4, 7

[43] Jie Ren, Han Xu, Yiding Liu, Yingqian Cui, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. A Robust Semantics-based Watermark for Large Language Model against Paraphrasing. *CoRR abs/2311.08721*, 2023. 14

[44] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. DeepSigns: A Generic Watermarking Framework for IP Protection of Deep Learning Models. *CoRR abs/1804.00750*, 2018. 1, 3

[45] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding Watermarks into Deep Neural Networks. In *International Conference on Multimedia Retrieval (ICMR)*, pages 269–277. ACM, 2017. 1, 3, 14

[46] Tianhao Wang and Florian Kerschbaum. Attacks on Digital Watermarks for Deep Neural Networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2622–2626. IEEE, 2019. 11, 14

[47] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2022. 1

[48] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 Technical Report. *CoRR abs/2407.10671*, 2024. 7

[49] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 Technical Report. *CoRR abs/2412.15115*, 2024. 7

[50] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph. Stoecklin, Heqing Huang, and Ian Molloy. Protecting Intellectual Property of Deep Neural Networks with Watermarking. In *ACM Asia Conference on Computer and Communications Security (ASIACCS)*, pages 159–172. ACM, 2018. 14

[51] Rui Zhang, Hongwei Li, Rui Wen, Wenbo Jiang, Yuan Zhang, Michael Backes, Yun Shen, and Yang Zhang. Instruction Backdoor Attacks Against Customized LLMs. In *USENIX Security Symposium (USENIX Security)*. USENIX, 2024. 6

[52] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level Convolutional Networks for Text Classification. In *Annual Conference on Neural Information Processing Systems (NIPS)*, pages 649–657. NIPS, 2015. 2, 6

[53] Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang. Provable Robust Watermarking for AI-Generated Text. In *International Conference on Learning Representations (ICLR)*. ICLR, 2024. 1, 14

[54] Qi Zhong, Leo Yu Zhang, Shengshan Hu, Longxiang Gao, Jun Zhang, and Yong Xiang. Attention Distraction: Watermark Removal Through Continual Learning with Selective Forgetting. In *International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2022. 11, 14

[55] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large Language Models are Human-Level Prompt Engineers. In *International Conference on Learning Representations (ICLR)*, 2023. 1

[56] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. MiniGPT-4: Enhancing Vision-Language Understanding with Advanced Large Language Models. *CoRR abs/2304.10592*, 2023. 1, 4

[57] Michael Zhu and Suyog Gupta. To Prune, or Not to Prune: Exploring the Efficacy of Pruning for Model Compression. In *International Conference on Learning Representations (ICLR)*, 2018. 14

A Appendix

A.1 Additional Results

Table 6: Performance of weak watermark method after fine-tuning across different upstream and downstream models as well as datasets. For the *Z-Score*, we set a threshold of 4, following previous works [16, 17]. Once the *Z-Score* exceeds 4, we consider the confidence level for a watermark in the generated text to be 1.000. We report the average *Z-Score* and *WSR* for the watermark performance as well as the *MAUVE* and *PPL* for the utility performance across our test datasets.

Evaluation Metrics	AG News						DialogSum					
	Llama			Ministral			Llama			Ministral		
	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna
Z-Score	-0.432	-0.120	-0.048	-0.122	0.115	0.176	-0.381	-0.251	1.027	-0.739	-0.228	-0.901
WSR	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.020	0.000	0.000	0.000
MAUVE	0.706	0.606	0.717	0.929	0.714	0.804	0.731	0.706	0.709	0.804	0.715	0.631
PPL	17.179	13.119	28.557	17.179	10.018	28.554	1.923	14.037	15.743	2.493	13.043	28.470

Table 7: Performance of robust watermark method after fine-tuning across different upstream and downstream models as well as datasets. We report the average *WSR* for the watermark performance as well as the *MAUVE* and *PPL* for the utility performance across our test datasets.

Evaluation Metrics	AG News						DialogSum					
	Llama			Ministral			Llama			Ministral		
	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna
WSR	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
MAUVE	0.714	0.804	0.705	0.873	0.605	0.450	0.715	0.810	0.809	0.741	0.741	0.875
PPL	17.547	23.903	13.336	17.589	27.970	33.327	2.363	13.476	12.653	2.184	13.671	7.628

Table 8: Performance of present continuous tense after fine-tuning across different upstream and downstream models as well as datasets. We report the average *WSR* for the watermark performance as well as the *MAUVE* and *PPL* for the utility performance across our test datasets.

Evaluation Metrics	AG News						DialogSum					
	Llama			Ministral			Llama			Ministral		
	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna
WSR	0.120	0.080	0.220	0.120	0.220	0.240	0.280	0.080	0.840	0.280	0.120	0.840
MAUVE	0.709	0.735	0.706	0.683	0.643	0.635	0.706	0.704	0.687	0.749	0.819	0.009
PPL	16.892	19.280	23.288	19.834	29.727	33.328	2.847	13.421	15.783	2.485	13.056	16.893

Table 9: Performance of passive voice after fine-tuning across different upstream and downstream models as well as datasets. We report the average *WSR* for the watermark performance as well as the *MAUVE* and *PPL* for the utility performance across our test datasets.

Evaluation Metrics	AG News						DialogSum					
	Llama			Ministral			Llama			Ministral		
	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna
WSR	0.340	0.400	0.980	0.240	0.380	0.980	0.340	0.100	0.820	0.380	0.080	0.820
MAUVE	0.894	0.792	0.717	0.794	0.792	0.817	0.732	0.735	0.809	0.764	0.824	0.711
PPL	17.264	16.216	13.327	19.469	22.292	26.349	3.195	15.332	23.343	2.153	14.456	27.372

Table 10: Performance of weak watermark method after pruning across different upstream and downstream models as well as datasets. For the *Z-Score*, we set a threshold of 4, following previous works [16, 17]. Once the *Z-Score* exceeds 4, we consider the confidence level for a watermark in the generated text to be 1.000. We report the average *Z-Score* and *WSR* for the watermark performance as well as the *MAUVE* and *PPL* for the utility performance across our test datasets.

Evaluation Metrics	AG News						DialogSum					
	Llama			Ministral			Llama			Ministral		
	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna
Z-Score	-0.432	-0.120	-0.048	-0.122	0.115	0.176	-0.381	-0.251	1.027	-0.739	-0.228	-0.901
WSR	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.020	0.000	0.000	0.000
MAUVE	0.640	0.746	0.720	0.609	0.614	0.635	0.631	0.572	0.745	0.721	0.645	0.709
PPL	17.179	33.119	28.557	17.179	29.018	28.554	1.952	14.037	10.620	2.152	13.043	9.181

Table 11: Performance of robust watermark method after pruning across different upstream and downstream models as well as datasets. We report the average *WSR* for the watermark performance as well as the *MAUVE* and *PPL* for the utility performance across our test datasets.

Evaluation Metrics	AG News						DialogSum					
	Llama			Ministral			Llama			Ministral		
	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna
WSR	0.800	0.900	0.800	0.900	0.760	0.860	0.820	0.840	0.720	0.800	0.880	0.720
MAUVE	0.514	0.641	0.683	0.687	0.631	0.650	0.466	0.810	0.745	0.610	0.741	0.687
PPL	18.305	23.903	33.336	16.323	32.970	33.327	2.148	13.476	11.158	2.879	13.671	10.920

Table 12: Performance of present continuous tense after pruning across different upstream and downstream models as well as datasets. We report the average *WSR* for the watermark performance as well as the *MAUVE* and *PPL* for the utility performance across our test datasets.

Evaluation Metrics	AG News						DialogSum					
	Llama			Ministral			Llama			Ministral		
	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna
WSR	0.820	0.800	0.840	0.820	0.820	0.760	0.280	0.800	0.840	0.880	0.820	0.840
MAUVE	0.709	0.635	0.706	0.668	0.743	0.635	0.762	0.721	0.875	0.749	0.819	0.690
PPL	19.342	19.280	33.288	17.236	19.727	33.328	1.988	13.421	11.046	2.405	13.056	11.095

Table 13: Performance of passive voice after pruning across different upstream and downstream models as well as datasets. We report the average *WSR* for the watermark performance as well as the *MAUVE* and *PPL* for the utility performance across our test datasets.

Evaluation Metrics	AG News						DialogSum					
	Llama			Ministral			Llama			Ministral		
	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna
WSR	0.340	0.640	0.980	0.840	0.880	0.980	0.340	0.910	0.820	0.880	0.940	0.820
MAUVE	0.887	0.792	0.717	0.794	0.792	0.817	0.632	0.646	0.689	0.618	0.710	0.631
PPL	17.930	16.216	33.327	19.635	22.292	23.349	2.506	15.332	11.820	3.051	14.456	10.714

Table 14: Performance of weak watermark method after quantization across different upstream and downstream models as well as datasets. For the *Z-Score*, we set a threshold of 4, following previous works [16, 17]. Once the *Z-Score* exceeds 4, we consider the confidence level for a watermark in the generated text to be 1.000. We report the average *Z-Score* and *WSR* for the watermark performance as well as the *MAUVE* and *PPL* for the utility performance across our test datasets.

Evaluation Metrics	AG News						DialogSum					
	Llama			Ministral			Llama			Ministral		
	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna
Z-Score	8.079	6.548	6.013	5.022	5.230	4.454	1.931	-0.152	1.027	1.464	-0.155	-0.901
WSR	0.780	0.700	0.600	0.580	0.540	0.460	0.060	0.000	0.020	0.060	0.000	0.000
MAUVE	0.640	0.567	0.714	0.693	0.581	0.632	0.685	0.578	0.743	0.701	0.645	0.709
PPL	13.610	34.583	24.640	13.195	37.167	29.419	2.571	13.173	9.105	2.418	13.241	8.910

Table 15: Performance of robust watermark method after quantization across different upstream and downstream models as well as datasets. We report the average *WSR* for the watermark performance as well as the *MAUVE* and *PPL* for the utility performance across our test datasets.

Evaluation Metrics	AG News						DialogSum					
	Llama			Ministral			Llama			Ministral		
	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna
WSR	0.920	1.000	0.000	1.000	1.000	0.920	0.960	0.960	1.000	0.640	0.880	1.000
MAUVE	0.395	0.604	0.502	0.673	0.635	0.450	0.466	0.813	0.745	0.642	0.720	0.743
PPL	16.230	19.199	39.154	20.677	19.590	37.367	2.262	13.048	9.612	2.029	13.230	9.595

Table 16: Performance of present continuous tense after quantization across different upstream and downstream models as well as datasets. We report the average *WSR* for the watermark performance as well as the *MAUVE* and *PPL* for the utility performance across our test datasets.

Evaluation Metrics	AG News						DialogSum					
	Llama			Ministral			Llama			Ministral		
	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna
WSR	1.000	1.000	1.000	1.000	0.980	1.000	1.000	0.980	1.000	1.000	1.000	0.980
MAUVE	0.709	0.673	0.706	0.682	0.743	0.635	0.622	0.407	0.745	0.705	0.818	0.690
PPL	15.291	14.957	29.467	19.090	15.673	24.018	2.314	14.136	9.646	2.469	14.342	9.713

Table 17: Performance of passive voice after quantization across different upstream and downstream models as well as datasets. We report the average *WSR* for the watermark performance as well as the *MAUVE* and *PPL* for the utility performance across our test datasets.

Evaluation Metrics	AG News						DialogSum					
	Llama			Ministral			Llama			Ministral		
	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna	T5	Qwen	Vicuna
WSR	1.000	1.000	0.960	0.960	0.980	1.000	1.000	0.980	1.000	1.000	1.000	1.000
MAUVE	0.894	0.792	0.717	0.794	0.792	0.817	0.632	0.635	0.708	0.638	0.724	0.610
PPL	15.306	9.102	14.692	16.999	15.640	15.450	2.450	15.945	11.541	1.973	13.765	9.539