SpARK: An Embarrassingly Simple Sparse Watermarking in LLMs with Enhanced Text Quality

Anonymous ACL submission

Abstract

With the widespread adoption of Large Language Models (LLMs), concerns about potential misuse have emerged. To this end, watermarking has been adapted to LLM, enabling a simple and effective way to detect and monitor generated text. However, while the existing methods can differentiate between watermarked and unwatermarked text with high accuracy, they often face a trade-off between the quality of the generated text and the effectiveness of the watermarking process. In this work, we present a novel type of LLM watermark, **Sp**arse Watermark (or SpARK), which aims to mitigate this trade-off by applying watermarks to a small subset of generated tokens distributed across the text. To demonstrate this type of watermark, we introduce two novel variants, **SpARK-P** and **SpARK-R**, which achieve sparsity by anchoring watermarked tokens to words that have specific Part-of-Speech (POS) tags and specific hash values w.r.t a pseudo-random hash function, respectively. Our experimental results demonstrate that the proposed watermarking schemes, albeit *embarrassingly simple*, are *incredibly effective*, achieving high detectability while generating text that outperforms previous LLM watermarking methods in quality across various tasks. SpARK further advances the watermarking capability for LLMs while maintaining their generated text quality.

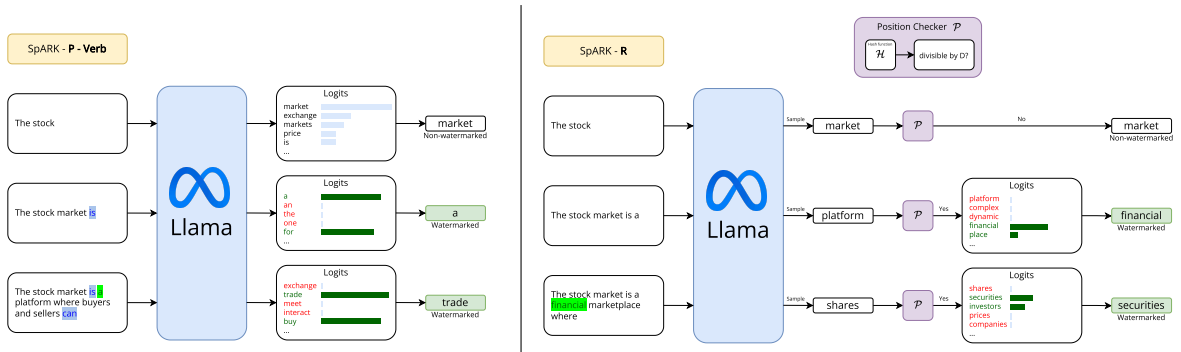
1 Introduction

Recent advancements in Large Language Models (LLM) have shown exceptional performance in a multitude of tasks. From generating documents to answering questions on different topics, LLMs such as Meta's Llama (Touvron et al., 2023) and OpenAI's GPT (OpenAI, 2023) have become the foundation upon which many AI applications are built (Brohan et al., 2023; Luo et al., 2024; Huang et al., 2023). However, as these applications increase in their capabilities and accessibility, a growing risk of them being used for malicious purposes, such as generating fake news and being used for cheating assignments, becomes increasingly apparent.

With the ever-increasing problem of LLMs being misused, monitoring the generated text and its usage has become an increasingly crucial direction for research. One effective way for tracking the usage of generated text is by watermarking (Kirchenbauer et al., 2023, 2024; Zhao et al., 2024) - embedding imperceptible information into the generated text, thereby making it easier to detect and track for potential misuse. Recent studies have demonstrated the effectiveness and versatility of watermarks in embedding ownership information into generated text and distinguishing it from non-watermarked and human-written text (Krishna et al., 2023).

In addition to distinguishing between watermarked and non-watermarked texts, watermarking methods must also preserve the original text quality after embedding the secret information. However, prior works generally agree that there is a trade-off between the quality of the watermarked text and the strength of its watermark. For instance, Kirchenbauer et al. (2023) illustrates this trade-off by introducing a parameter that adjusts the extent to which their method affects the model's logits. By tuning this parameter, they demonstrate the balance between the quality of the generated text and the robustness of the watermark.

In this paper, we aim to circumvent the trade-off between watermark strength and text quality by proposing a watermarking method that augments only a portion of the generated text and checks for that portion of the text for watermark information. The main concept is illustrated in Figure 1. We show that by watermarking only a subset of the generated text, we can still maintain high detectability while minimizing the watermark's impact on the text quality. Our hypothesis is that while prior methods verify a watermark by checking every token within a text, the same effect can be



(a) SpARK-P (b) SpARK-R

Figure 1: An overview of our proposed *SpARK-P* and *SpARK-R*. In general, for each generation step t, if a suitable watermark location is found, we will divide the vocabularies into *Green/Red* list and restrict sampling from the *Green* list. Otherwise, we generate the next token with the original probability.

achieved by checking only a specific portion if the locations of the watermarked elements are known. This helps preserve the quality of the generated text by keeping a large portion of the original generated text while still successfully embedding the secret information. Our contributions can be summarized as follows:

- We introduce *Sparse Watermark*, a novel watermarking category for LLMs that is designed to preserve both text quality and detectability by selectively watermarking and verifying only a subset of the generated text.
- We propose two novel Spark Watermark variants, SpARK-P and SpARK-R, which rely on Part-of-Speech (POS) tagging and a random hash function, respectively, to embed and detect watermarks within the generated text. The two variants enable flexible and covert strategies for sparsely selecting watermarking positions while maintaining the quality of the generated text.
- Through extensive experiments on *SpARK*, we demonstrate that *Sparse Watermark* effectively maintains high text quality generated by LLMs and watermark detectability, outperforming several previous methods across various generation tasks.

2 Related works

AI-generated text detection. The methods for monitoring the usage of AI-generated text can be generally classified into two main categories: AI text detection and watermarking. Of these, watermarking has proven to be more reliable and effective for distinguishing between generated and human-written text, as well as between watermarked and unwatermarked generated text (Krishna et al., 2023). In addition, as companies and research communities strive to close the gap between LLM-generated and human-written texts, relying solely on AI text detection of the original text will become increasingly challenging. The main objective of LLM watermarking is to inject secret information imperceptible to humans into the generated text, which can later be verified by using watermark detection mechanisms (Kirchenbauer et al., 2024; Zhao et al., 2024; Kirchenbauer et al., 2024; Liu et al., 2024a; Gu et al., 2024).

Text watermarking for LLM. One common approach of text watermarking for LLMs focused on distorting the next token probability distribution of the language model. This is achieved by randomly dividing the vocabulary into two disjoint sets named *Green* list and *Red* List, and then promoting the generation of only tokens in the *Green* list with a bias parameter δ (Kirchenbauer et al., 2023). During detection, a detector with the secret key could recover the watermarked distribution and use a statistical test to verify the presence of the watermark.

Recent works have advanced LLM watermarking by enhancing robustness and security. Zhao et al. (2024) improved resilience using a fixed *Green* list, while Kirchenbauer et al. (2024) proposed hashing schemes for greater robustness. Yoo et al. (2023) strengthened robustness post-generation via semantic-dependent syntactic watermarking to resist editing attacks. Training-based methods also emerged: Liu et al. (2024b) leveraged token semantics for robust watermarking, Liu et al. (2024a) trained dual networks for generation and detection, and Lee et al. (2023) applied entropy thresholding to preserve code correctness.

Effects of watermark on text quality. However, while these recent works have considerably enhanced the robustness, detectability, and unforge-

Algorithm 1 SpARK-P Encoding

procedure WATERMARK($\mathbf{x}_{\text{prompt}}$, $P_{\mathcal{M}}$, $hash$)
  $T \leftarrow$ Convert tokens $\mathbf{x}_{\text{prompt}}$ to normal text
  $W \leftarrow$ Last word of T
  $P_{\text{tag}} \leftarrow \text{POS}(W, T)$
  if $P_{\text{tag}} \in I$ **then**
    $G \leftarrow$ GenerateGreenList(T, $hash$)
    $P_{\mathcal{M}} \leftarrow$ ApplyGreenList($P_{\mathcal{M}}, G$)
  end if
  return $\mathbf{x}_{\text{prompt}}$, $P_{\mathcal{M}}$
end procedure

Algorithm 2 SpARK-R Encoding

procedure WATERMARK($\mathbf{x}_{\text{prompt}}$, $P_{\mathcal{M}}$,D)
  $T \leftarrow$ Get last token from $\mathbf{x}$
  $H_{\text{last}} \leftarrow \text{PRF}(W, T)$
  if $mod(H_{\text{last}}, D) == 0$ **then**
    $\mathbf{x}_{\text{prompt}} = \mathbf{x}_{\text{prompt}}[: -1]$
    $G \leftarrow$ GenerateGreenList(T, H_{last})
    $P_{\mathcal{M}} \leftarrow$ ApplyGreenList($P_{\mathcal{M}}(t-1), G$)
  end if
  return $\mathbf{x}_{\text{prompt}}$, $P_{\mathcal{M}}$
end procedure

ability of LLM watermarking, it is generally agreed that there is a trade-off between the quality of the watermarked text and the strength of its watermark. The distribution shift introduced in Kirchenbauer et al. (2023) enhances the detectability of the watermark, but it simultaneously allows less likely tokens to be generated, thus affecting the intrinsic quality of the generated text. Recently, Tu et al. (2023) introduced a benchmark method of several LLM watermarking algorithms and verified this deterioration of text quality. To minimize the impact on the generation quality, Christ et al. (2023); Kuditipudi et al. (2023) proposed to embed the watermark during the token sampling process, thus inducing zero distortion to the probability distribution of the LLM. However, in practice, the sampling-based schemes struggled to produce a detectable watermark for low-temperature settings (Piet et al., 2023). Huo et al. (2024) introduced a multi-objective optimization method to dynamically generate bias parameters and *Green* list ratio to achieve both detectability and semantic coherence. In contrast, our approach, *SpARK*, emphasizes preserving the strength and semantic integrity of generated text by leveraging the innate structure of natural language, eliminating the need for training.

3 Proposed method

3.1 Notations and preliminaries

We first introduce the notations used in this paper. Let $\mathcal{M}$ be an autoregressive language model that takes a tokenized prompt $\mathbf{x}_{\text{prompt}} = \{x_{-N}, ..., x_{-2}, x_{-1}\}$ and output a sequence of tokens that simulate natural responses. At generation step t, the input for the language model $\mathcal{M}$ is combined sequences of tokens $\mathbf{x}_{\text{prompt}}$ and the tokens $\mathbf{x} = \{x_0, ..., x_{t-1}\}$ previously generated by $\mathcal{M}$ in the previous steps. The language model $\mathcal{M}$ then takes the input and outputs a probability distribution of the next token over the vocabulary $\mathcal{V}$ of the language model: $P_{\mathcal{M}}(x_{-N}, ..., x_{t-1}) = (P_{\mathcal{M}}(\upsilon|x_{-N}, ..., x_{t-1})|\upsilon \in \mathcal{V})$.

According to Kirchenbauer et al. (2024), watermark algorithms are defined using four parameters. The hash function $\mathcal{H}$ generates a pseudo-random $hash$ using the context of the generated text with context width h, the fraction of green list token γ, and the magnitude of the logit bias δ. After the watermarked text is generated, one can use the same parameters to calculate and retrieve a set of green tokens s in the generated text. We then use this set to calculate the statistical significance of $|s|$ number of green tokens that appeared in the generated text with token length T. We can use a one-proportion z-test assuming the null hypothesis $\mathcal{H}_0$ which states: "The text sequence is generated without a watermark". The z-score is then calculated as

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

If a text sequence's z-score surpasses a defined threshold, we can confidently determine that the text has been watermarked.

3.2 Threat models

In this paper, we consider the same threat model as in prior works (Kirchenbauer et al., 2023; Zhao et al., 2024; Liu et al., 2024a). The goal is to embed a watermark for LLM so that users can later verify if certain texts are generated by the LLM. We assume that the adversary is aware of the presence of watermarks and attempts to evade the watermark detection when using the LLM. The adversary could have access to both open-source and private (non-watermarked) language models to produce an alternate text. Consistent with prior works, we only consider attacks such that the modifications are able to erase the watermark without significantly deviating from the original semantics of the texts.

3.3 Sparse watermarking

In previous works, most watermarking techniques attempt to encode watermark information into each token in the generated text. As the strength of the watermarking method increases, more tokens

Algorithm 3 Text Generation with SpARK

procedure GENERATETEXT($\mathbf{x}_{\text{prompt}}$)
  for $t = 0, 1, \ldots$ **do**
    $P_{\mathcal{M}}(t) \leftarrow \mathcal{M}(x_{-N}, \ldots, x_{t-1})$
    $hash \leftarrow \mathcal{H}(x_{-N}, \ldots, x_{t-1})$
    $\mathbf{x}_{\text{prompt}}, P_{\mathcal{M}}(t) \leftarrow \text{Watermark}(\mathbf{x}_{\text{prompt}}, P_{\mathcal{M}}(t), hash)$
    $\mathbf{x}_{\text{prompt}}(t) \leftarrow \text{Sample}(G)$
  end for
  return $\mathbf{x}_{\text{prompt}}[0 : t]$
end procedure

are adversely affected, which decreases the quality of the generated text (Kirchenbauer et al., 2023). We aim to improve the text quality by watermarking the generated text sparsely, which, however, is non-trivial. Attempting to watermark sparsely without knowing the location of the watermarked elements would be akin to using the previous watermark methods with low strength. This is due to the statistical test also including the non-watermarked portions of the generated text. To this end, by isolating and conducting the statistical test specifically on the watermarked portions of the generated text, we can significantly enhance detectability while maintaining higher text quality compared to using previous methods with stronger watermarking. We empirically demonstrate this effect in the appendix section K, showing that just by watermarking a few tokens, we were able to reach the same z-score as the other baseline watermark method.

3.3.1 SpARK-P: Sparse watermarking with POS tags

We utilize the Universal Part-of-Speech (POS) tags (Petrov et al., 2012) that exist in the generated text to mark the positions of the watermarked tokens in the text sequence.. Specifically, during the generation process, this method, which we call SpARK-P, selects the positions to watermark based on the POS of tokens that have been generated, allowing the watermark positions to be tied to the sentence structure. This makes the watermark more resilient to insertions/deletions of tokens in the generated text and also makes it easier to extract the watermarked portion of the text using the POS tags.

Before using SpARK-P, we first select a list of POS tags I to be used for watermarking. When the text generation process starts, as described in Algorithm 3, we verify when the model has generated a full word by determining if the next token with the highest probability is the start of a new word. While LLMs sample the next token differently with different sampling schemes, using this strategy could consistently inform us when a full word has been generated, without the need to backtrack during the generation process. Once a full word is produced by the language model, we obtain its POS tag P_{tag}, and watermark the next token only if $P_{tag} \in I$. We choose to watermark the token next to the word with chosen POS tags, as watermarking those words directly would not guarantee it to have the same POS tag after being watermarked, leading to inconsistencies. By using words that have a selected POS as an anchor, we can limit the number of watermarked tokens in the generated text and position them to be easily relocated when decoding. We outline the process of watermarking using POS tags in Algorithm 1.

To watermark the next token, we used a similar process and hashing scheme as described in Kirchenbauer et al. (2023), partitioning the vocabulary using γ and limiting the generations of new tokens to a subset of the vocabulary, the *Green* list G. While dividing the vocabulary, we only select tokens that start a new word, as it would not affect the previous words and their POS tags, making the decoding process more consistent. Additionally, our method does not use δ to increase the bias for generating green list tokens, but instead, we restrict the model to only select from the *Green* list. This helps the encoding process to utilize all of the tokens it has access to, as it can only watermark a small portion of the generated text.

3.3.2 SpARK-R: Sparse watermarking with Pseudo-Random Function

While SpARK-P can be an effective sparse watermarking scheme, there is a limit to the number of locations the watermark is embedded. To further expand and **randomize** the possible locations for watermarking, we introduce SpARK-R, which determines the locations of the watermark through the pseudo-random hash function (PRF). Specifically, we tie the positions of the watermarked tokens to the hashed values of the generated tokens. As the seed and the PRF are secret to only the model owner, this makes it significantly harder to reverse engineer the positions of the watermarked tokens, compared to those watermarked by SpARK-P.

The watermarking process is similar to SpARK-P where we sparsely inject the watermarks throughout the sentence. However, SpARK-R calculated the remainder of the hash value generated by the PRF with a modulo operation to specify the locations for watermarking. Specifically, for each generated token, we compute its hash value using its token ID and compute its remainder against a

Algorithm 4 SpARK Watermark Detection

```
procedure DETECTWATERMARK(y, I, hash)
    s = 0
    T = 0
    for i = 1, 2, ...|y| do
        P_tag ← POS(y[i], y[: i])
        if P_tag ∈ I then
            T = T + 1
            next_token ← NextToken(y[: i], i)
            G ← GenerateGreenList(y[: i], hash)
            if next_token ∈ G then s = s + 1
        end if
    end for
    z ← (s − γT) / (γ√((1−γ)T))
    return z > threshold
end procedure
```

divisor D. When the remainder is 0, SpARK-R regenerates the token but out of a green list. We generate a fixed green list similar to Kirchenbauer et al. (2024), which makes the watermark more robust and less likely to be removed, by not needing past context. This approach allows us to retrieve the watermarks with the PRF and also control the sparsity of the watermark by adjusting D.

3.4 SpARK watermark detection

Since SpARK watermark is sparse, we identify the specific positions we have selected for the watermark to ensure that the unwatermarked portions of the text are not considered in the z-score calculation. This process would preserve the strength of the sparse watermark. For SpARK-P watermark detection, we first identify the words whose POS tags are in the list I and select the next token. SpARK-R also follows a similar detection process by first identifying the positions of the tokens being watermarked, selecting the tokens whose hash value is divisible by D. These selected tokens are the ones we would watermark during the encoding process and thus would be in the *Green* list G. At each of the selected token positions, we recover the G using the hashing scheme mentioned in the encoding process and check if the token in that position is in G. We then calculate the statistical significance of the number of green tokens that appeared in the generated text, as expressed in Equation 1. However, as we only apply the watermark to the list of selected tokens, T (the total number of tokens) would be replaced by the number of tokens in the watermarked positions. The watermark detection step is presented in Algorithm 4.

4 Experiments

We provide average results for watermark detection on Long-Form Question Answering (LFQA) and Summarization along with robustness results under this section. Additionally, we report the detection results under adversarial robustness for each dataset, perplexity results, watermark stealing results, and comparison to distortion-free watermark in Section D, Section G, Section H, and Section F of the appendix, respectively.

4.1 Experimental setup

We choose the popular open-source Llama2-7b, which has been instruction-tuned to align with human preference, as our baseline model for testing the watermarking methods. In addition, we also conduct experiments on Phi-3, a 3.8 billion language model that has been shown to outperform bigger LLMs on several benchmarks. We compare the performance of SpARK against six LLM watermarking techniques:

- **Hard watermark** (Kirchenbauer et al., 2023): the initial method that restricts the model to only generating a portion of the vocabulary, called the *Green* list, & uses the statistical test to detect the watermark.
- **LeftHash watermark (Soft watermark)** (Kirchenbauer et al., 2023): a method that encourages the model to generate tokens from the *Green* list by adding a constant γ to its logit, instead of restricting the model.
- **SelfHash watermark** (Kirchenbauer et al., 2024): a method similar to LeftHash watermark, with the main difference where the hashing scheme uses yet-to-be-generated tokens to generate the *Green* list.
- **Unigram watermark** (Zhao et al., 2024): a watermark method that simplifies the watermark process by utilizing a fixed *Green* list.
- **SWEET watermark** (Lee et al., 2023): a method that limits the watermark process to only regions with high entropy, eliminating low-entropy segments in watermark generation and detection.
- **EWD watermark** (Lu et al., 2024): a watermark detection method that assigns weight to each token based on the model's entropy.

To validate the detectability and the quality of the text generated by the watermark methods, we used an experiment setting similar to WaterBench (Tu et al., 2023). This benchmark procedure aims to measure both the quality of the generated text and its detectability. We focused the experiment only on the long-answer datasets, following the

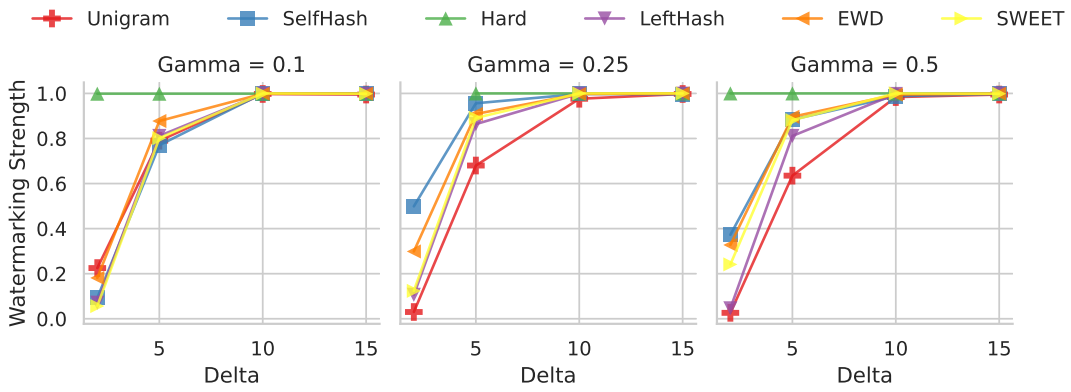


Figure 2: True Positive Rate (TPR) for each method on the selected dataset, generated using Llama2-7b with different hyper-parameters for γ and δ.

same setting as in prior works of LLM watermarking (Kirchenbauer et al., 2024; Gu et al., 2024). We conduct the same hyper-parameter search experiments on long-answer datasets to find parameters that are more suitable for watermarking these long text answers. The watermarking strength results are shown in Figure 2. During the hyper-parameter search, we select the parameters that are close to the original parameters of each method and have a True Positive Rate of greater than 0.99.

To summarize, we use the ELI5 (Fan et al., 2019) and FinanceQA (Maia et al., 2018) datasets for Long-form QA, and MultiNews (Fabbri et al., 2019) and QMSum (Zhong et al., 2021) for Summarization. We perform a hyperparameter search by evaluating each method's TPR across different settings. As shown in Figure 2, watermark strength increases as γ decreases and δ increases. Most methods achieve a TPR above 0.99 when δ is sufficiently high, guiding our choice of γ close to each method's original parameters. We select hyperparameters near the originals while maintaining TPR > 0.99. Quality is measured by ROUGE-L F1, and detectability by True Positive and True Negative Rates.

For SpARK-P, we selected three POS tags for the main experiment: Verb, Noun, and Determiner. This is because, based on Tab. 6, these three tags have 100% of document frequency. We selected $\gamma = 0.05$ for the SpARK-P, to increase the strength of each watermark token. By choosing a small γ, we demonstrate that SpARK-P, and Sparse Watermark in general, can match other baseline methods in detectability, while also maintaining higher generation performance. We also provide the results containing the TPR of SpARK-P under different POS tags and γ. As for SpARK-R, we applied the same hyper-parameter search process shown in Figure 2. As the sparsity of the watermark depends on the modular value D, we tested the model with different γ values and with different D. We select the parameters that have a TPR closest to 99%, which would maximize the quality of the generated text while guaranteeing its detectability. From this process, we determined the values for SpARK-R is 0.1 for γ and 6 for D, which we applied for every model in the main experiments.

4.2 Results of detectability and text quality

As mentioned in Section 4.1, we conduct evaluations with parameters that achieved greater than 0.99 TPR and close to the original parameters of each method. We report the results of the watermarks' performance in Tables 1 and 2.

Overall, the detection performance of the baseline watermark method is high, having above 99% TPR and True Negative Rate in both tasks. In addition, tuning the parameter for long-answer text increased the generation performance of all watermark methods without degrading their detectability. Compared to the baseline watermark methods, SpARK-P achieved similar detection performance, while consistently achieving the highest generation performance in both tasks. On Llama2-7b, all three of the SpARK-P variants using different POS tags reached the top 3 spots in terms of generation performance. When using any of the SpARK-P variants, the ROUGE-L score of the original model would only be reduced by at most 21.64%. SpARK-R also achieved detection performances comparable to other methods, while having a generally high generation quality compared to the baseline methods. The generation quality of Llama2-7b model would only be reduced by 18.91% at most, lower than any other baseline watermark methods. In contrast, the performance of other watermarks would decrease that down by at

Table 1: Comparison of True Positive Rate (TPR), True Negative Rate (TNR), ROUGE-L score (R-L), decrease in percentage point of ROUGE-L score (Δ) and the semantic similarity of watermarked and non-watermarked text (Sem.) of different watermarking algorithms, evaluated on Llama-2 model. The best and second-best performances are in **bold** and underline, respectively.

	Long-form QA				Summarization				Sem.
	TPR	TNR	R-L	Δ	TPR	TNR	R-L	Δ	
No Watermark	–	–	21.59	–	–	–	23.47	–	–
Hard	100.0	100.0	16.76	$\downarrow$ 22.37%	100.0	100.0	16.63	$\downarrow$ 29.14%	0.765
LeftHash	100.0	100.0	14.55	$\downarrow$ 32.61%	99.5	99.5	13.33	$\downarrow$ 43.20%	0.693
SelfHash	99.5	93.5	12.75	$\downarrow$ 40.94%	100.0	96.0	12.54	$\downarrow$ 46.57%	0.663
Unigram	99.8	100.0	11.43	$\downarrow$ 47.06%	99.3	100.0	11.53	$\downarrow$ 50.87%	0.652
SWEET	100.0	100.0	18.88	$\downarrow$ 12.55%	100.0	100.0	18.13	$\downarrow$ 22.71%	0.807
EWD	100.0	100.0	18.12	$\downarrow$ 16.07%	100.0	100.0	18.11	$\downarrow$ 22.83%	0.802
SpARK-P - Verb	100.0	99.0	18.87	$\downarrow$ 12.60%	100.0	99.5	**20.95**	$\downarrow$ 10.74%	**0.836**
SpARK-P - Noun	100.0	99.5	18.48	$\downarrow$ 14.40%	100.0	100.0	18.39	$\downarrow$ 21.64%	0.794
SpARK-P - Determiner	100.0	98.8	**19.20**	$\downarrow$ 11.07%	100.0	98.0	20.89	$\downarrow$ 10.99%	0.814
SpARK-R	100.0	99.0	18.20	$\downarrow$ 15.70%	99.8	100.0	19.03	$\downarrow$ 18.91%	0.781

Table 2: Comparison of True Positive Rate (TPR), True Negative Rate (TNR), ROUGE-L score (R-L), decrease in percentage point of ROUGE-L score (Δ) and the semantic similarity of watermarked and non-watermarked text (Sem.) of different watermarking algorithms evaluated on Phi-3 model. The best and second-best performances are in **bold** and underline, respectively.

	Long-form QA				Summarization				Sem.
	TPR	TNR	R-L	Δ	TPR	TNR	R-L	Δ	
No Watermark	–	–	22.62	–	–	–	23.37	–	–
Hard	100.0	100.0	15.19	$\downarrow$ 32.83%	100.0	100.0	11.22	$\downarrow$ 52.01%	0.567
LeftHash	100.0	100.0	19.55	$\downarrow$ 15.34%	99.3	99.5	15.71	$\downarrow$ 32.78%	0.779
SelfHash	100.0	97.0	19.51	$\downarrow$ 13.75%	99.8	99.5	16.85	$\downarrow$ 27.90%	0.806
Unigram	100.0	100.0	7.74	$\downarrow$ 65.77%	99.8	100.0	7.04	$\downarrow$ 69.88%	0.425
SWEET	98.8	100.0	19.29	$\downarrow$ 14.72%	100.0	100.0	16.46	$\downarrow$ 29.56%	0.775
EWD	100.0	100.0	20.21	$\downarrow$ 10.65%	99.8	100.0	17.25	$\downarrow$ 26.18%	0.806
SpARK-P- Verb	100.0	99.0	**21.45**	$\downarrow$ 5.17%	100.0	99.5	**20.87**	$\downarrow$ 10.72%	**0.850**
SpARK-P- Noun	99.5	99.5	19.46	$\downarrow$ 13.95%	100.0	100.0	18.27	$\downarrow$ 21.84%	0.787
SpARK-P- Determiner	99.5	98.8	21.18	$\downarrow$ 6.37%	100.0	99.0	20.86	$\downarrow$ 10.74%	0.829
SpARK-R	99.8	99.0	19.10	$\downarrow$ 15.56%	98.3	100	19.15	$\downarrow$ 18.05%	0.793

least 22% and by more than 50% at most. SpARK also has some of the highest semantic similarity between the non-watermarked text and watermarked text, with the highest being 0.836 for SpARK-P and the lowest being 0.781 for SpARK-R.

The same phenomenon can be seen on Phi-3, as SpARK-P maintains the generated text quality while having a high TPR compared to other methods. For long-form QA, the Verb and Determiner variants of SpARK-P only reduce the quality by roughly 5% and 6%, respectively, while other baseline watermark decreases the quality of the text by at least 10.65% (EWD) and at most 65.77% (Unigram). For summarization, SpARK-P maintains the best generation text with all three variants, degrading the quality by at most 21.8%, where other methods admit decreases of at least 27%. For SpARK-R, the generated text still maintains the text quality, with the ROUGE-L score reduced by at most 21.64%, lower than the upper bounds of any baseline watermark method. In terms of semantic similarity between watermarked and non-watermarked text, SpARK-P maintains the highest positions, with the Verb variant achieving 0.850 and Determiner 0.829, while SpARK-R has a slightly lower semantic similarity of 0.793.

In summary, SpARK's results show that sparse watermark methods can produce better-generated texts, both in terms of semantic similarity and task performance, while having the same detectability. By watermarking a small portion of text sparsely, SpARK can preserve the performance and similarity of the generated text, while maintaining detectability by focusing the detection on smaller sets of tokens.

4.3 Results of robustness against attacks

As malicious players have the capability of modifying a sequence of watermarked text to evade the detector, watermarking methods need to ensure that the watermark is resilient against changes to the text. In order to illustrate the robustness of our proposed approach, we consider two realistic types of attack: substitution attacks and paraphrasing attacks.

- **Substitution Attack.** For the substitution attack,

Table 3: Average True Positive Rate under two settings of attacks: synonym substitution ($10\%, 30\%, 50\%$) and paraphrasing (40L, 40L-40O), evaluated on Llama2-7b and Phi-3. The best and second-best performances are in **bold** and underline, respectively.

Method	Llama2-7b					Phi-3				
	10%	30%	50%	40L	40L-40O	10%	30%	50%	40L	40L-40O
Hard	99.6	90.6	51.1	53.0	41.0	**100.0**	**100.0**	98.6	89.3	88.3
LeftHash	99.8	**99.0**	83.9	71.4	64.1	99.3	98.1	83.6	79.8	66.1
SelfHash	99.8	98.1	**92.3**	**75.0**	**69.5**	99.3	96.1	62.8	79.5	66.9
Unigram	99.5	96.9	91.4	59.8	50.9	99.9	99.6	**99.3**	89.1	**88.6**
SWEET	99.9	95.3	53.9	69.1	54.5	99.3	98.5	84.6	**92.1**	86.1
EWD	99.1	92.9	48.3	52.7	38.1	99.0	91.0	28.9	69.4	54.8
SpARK-P- Verb	99.8	96.3	72.4	54.3	43.5	99.6	96.3	72.5	64.0	54.1
SpARK-P- Noun	**100.0**	97.8	78.3	53.9	41.9	99.4	97.5	80.6	71.5	59.1
SpARK-P- Determiner	99.8	96.5	67.6	74.3	66.9	99.6	96.8	76.6	87.1	82.4
SpARK-R	99.3	97.1	89.3	72.4	65.3	98.1	97.1	91.5	85.3	77.4

a specified portion of the text is replaced with its corresponding synonyms, following the settings described in (Wang et al., 2024).

- **Paraphrasing Attack.** We evaluate the robustness of our method against paraphrasing attacks using DIPPER (Krishna et al., 2023) - a model that has been fine-tuned for paraphrasing and has been shown to successfully evade multiple AI-generated text detectors.

Tab. 3 demonstrates the resilience of our method against both substitution attacks and paraphrasing attacks. From the substitution results, we can see that although SpARK-P and SpARK-R do not always outperform other methods in every setting, they remain competitive, particularly under high substitution rates. For instance, with Llama2-7b, SpARK-R achieves 89.3% TPR at 50% substitution, outperforming other sparsity-based methods, SWEET and EWD. Similarly, for Phi-3, SpARK-R consistently maintains strong detectability (97.1% and 91.5%, respectively) under heavy substitution, being only behind robustness-focused watermarks such as Hard or Unigram. As the SpARK-R watermark is not conditioned on the previous tokens, random substitution is unable to erase the watermark by modifying the n-gram context of the watermarked tokens.

A similar phenomenon can be seen in the paraphrasing result. The performance of SpARK-P with determiners is demonstrated to be near state-of-the-art in terms of robustness for Llama2-7b, achieving 74.3% and 66.9% in TPR, slightly behind SelfHash in both settings. Similarly, SpARK-R is only 1.9% and 1.6% behind SpARK-P with determiners. For Phi-3, SpARK-P and SpARK-R can still achieve 87.1% and 82.4%, 85.3% and 77.4% for DIPPER, respectively, which are higher than LeftHash and SelfHash. While the Hard and Unigram watermarks achieved greater robustness on Phi-3, their generated texts have the lowest scores compared to other methods, as shown in the Tab. 2. SWEET achieves the best robustness for the 40L setting, yet the watermark degrades the ROUGE-L score by almost 30%. In contrast, SpARK-P and SpARK-R were able to achieve consistently high robustness against attacks while having the best results in terms of the generated text on both Llama2-7b and Phi-3.

5 Conclusion

We propose *SpARK*, a novel watermarking approach for LLM that encodes watermark information into the generated text, without degrading its quality. Different from other methods, this approach focuses on encoding a subset of tokens distributed sparsely throughout the generated text using POS tags (SpARK-P) and using a pseudo-random hash function (SpARK-R). Via this embarrassingly simple encoding scheme in generation, SpARK can minimize the impact of the watermark on the text quality while maintaining high detectability. Experimental results demonstrate the incredible effectiveness of this simple method in preserving the text quality, as evidenced by the ROUGE-L score and semantic similarity for four datasets compared to other methods. This highlights the potential of sparse watermarking for practical applications.

6 Limitations

Sparse watermarking, especially SpARK, provides a **embarrassingly and frustratingly simple** way to encode watermark information with *incredibly high detectability and text-quality preservation*. Sparse watermark is also **an adaptable approach** that can be integrated with existing watermark methods, further improving the generated text quality text; e.g., SpARK-P extends (Kirchenbauer

et al., 2023) with sparse watermarking.

Nevertheless, there exist limitations that can be improved. As shown in Section 4.3, SpARK-P's robustness is often lower compared to other watermark methods and only gets higher when certain POS tags are used. We also limit our study to watermarking Universal Tags (Petrov et al., 2012) in this paper; this potentially increases the chance of removing the watermark through trial and error. However, this limitation can be addressed by the use of SpARK-R, with its watermark locations being randomized and can be easily changed. Furthermore, since sparse watermarks focus on a small set of tokens in the generated text (a characteristic of sparse watermarks), it would be easier to remove the watermarks given the locations of each watermarked token compared to other watermark methods. However, locating the watermarked tokens is not an easy task, as this information can easily be changed, in the case of SpARK-P, by changing POS tags. Lastly, SpARK has difficulty watermarking short answers, as short answers sometimes do not contain the necessary words that can be utilized to watermark. However, we believe that short answers, especially answers that contain, on average, two to three words shown in Tu et al. (2023), can be found via search engines. Tasks with long answers, such as Summarization and Long-form Question Answering utilize LLM's unique text-generating capabilities, in which watermarking would be noticeably more useful. These limitations present intriguing opportunities to enhance the effectiveness of our current method, which we encourage future research to explore.

References

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, and 1 others. 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *Preprint*, arXiv:2307.15818.

Miranda Christ, Sam Gunn, and Or Zamir. 2023. Undetectable watermarks for language models. Cryptology ePrint Archive, Paper 2023/763. https://eprint.iacr.org/2023/763.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy. Association for Computational Linguistics.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567, Florence, Italy. Association for Computational Linguistics.

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

Chenchen Gu, Xiang Lisa Li, Percy Liang, and Tatsunori Hashimoto. 2024. On the learnability of watermarks for language models. In *The Twelfth International Conference on Learning Representations*.

Siyuan Huang, Zhengkai Jiang, Hao Dong, Yu Qiao, Peng Gao, and Hongsheng Li. 2023. Instruct2act: Mapping multi-modality instructions to robotic actions with large language model. *Preprint*, arXiv:2305.11176.

Mingjia Huo, Sai Ashish Somayajula, Youwei Liang, Ruisi Zhang, Farinaz Koushanfar, and Pengtao Xie. 2024. Token-specific watermarking with enhanced detectability and semantic coherence for large language models. *Preprint*, arXiv:2402.18059.

Nikola Jovanović, Robin Staab, and Martin Vechev. 2024. Watermark stealing in large language models.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. 2024. On the reliability of watermarks for large language models. In *The Twelfth International Conference on Learning Representations*.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Frederick Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of AI-generated text, but retrieval is an effective defense. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. 2023. Who wrote this code? watermarking for code generation. *Preprint*, arXiv:2305.15060.

Aiwei Liu, Leyi Pan, Xuming Hu, Shuang Li, Lijie Wen, Irwin King, and Philip S. Yu. 2024a. An unforgeable publicly verifiable watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. 2024b. A semantic invariant robust watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

Yepeng Liu and Yuheng Bu. 2024. Adaptive text watermark for large language models. *Preprint*, arXiv:2401.13927.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yijian Lu, Aiwei Liu, Dianzhi Yu, Jingjing Li, and Irwin King. 2024. An entropy-based text watermarking detection method. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11724–11735, Bangkok, Thailand. Association for Computational Linguistics.

Xiaoliang Luo, Akilles Rechardt, Guangzhi Sun, Kevin K. Nejad, Felipe Yáñez, Bati Yilmaz, Kangjoo Lee, Alexandra O. Cohen, Valentina Borghesani, Anton Pashkov, Daniele Marinazzo, Jonathan Nicholas, Alessandro Salatiello, Ilia Sucholutsky, Pasquale Minervini, Sepehr Razavi, Roberta Rocca, Elkhan Yusifov, Tereza Okalova, and 5 others. 2024. Large language models surpass human experts in predicting neuroscience results. *Preprint*, arXiv:2403.03230.

Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. Www'18 open challenge: Financial opinion mining and question answering. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, page 1941–1942, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2089–2096, Istanbul, Turkey. European Language Resources Association (ELRA).

Julien Piet, Chawin Sitawarin, Vivian Fang, Norman Mu, and David Wagner. 2023. Mark my words: Analyzing and evaluating language model watermarks. *arXiv preprint arXiv:2312.00273*.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2023. Can ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Shangqing Tu, Yuliang Sun, Yushi Bai, Jifan Yu, Lei Hou, and Juanzi Li. 2023. Waterbench: Towards holistic evaluation of watermarks for large language models. *arXiv preprint arXiv:2311.07138*.

Lean Wang, Wenkai Yang, Deli Chen, Hao Zhou, Yankai Lin, Fandong Meng, Jie Zhou, and Xu Sun. 2024. Towards codable watermarking for injecting multi-bits information to LLMs. In *The Twelfth International Conference on Learning Representations*.

Yihan Wu, Zhengmian Hu, Junfeng Guo, Hongyang Zhang, and Heng Huang. 2024. A resilient and accessible distribution-preserving watermark for large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 53443–53470. PMLR.

KiYoon Yoo, Wonhyuk Ahn, Jiho Jang, and Nojun Kwak. 2023. Robust multi-bit natural language watermarking through invariant features. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2092–2115, Toronto, Canada. Association for Computational Linguistics.

Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang. 2024. Provable robust watermarking for AI-generated text. In *The Twelfth International Conference on Learning Representations*.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir Radev. 2021. QMSum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5905–5921, Online. Association for Computational Linguistics.

A Theoretical support for methods

A.1 Notations

- $\mathbf{x}_{prompt} = \{x_{-N}, \ldots, x_{-1}\}$: Tokenized prompt.
- $\mathbf{x} = \{x_0, \ldots, x_{t-1}\}$: Sequence of previously generated tokens.
- $\mathbf{y}$: Generated Text / Normal text.
- $P_{\mathcal{M}}(x_{-N}, \ldots, x_{t-1})$: Probability distribution of the next token generated by the language model M.
- T: Normal text converted from tokens $\mathbf{x}_{prompt}$ and $\mathbf{x}$.
- W: Last word in T.
- $P_{tag}(x)$: POS tag of word x.
- I: Set of selected POS tags for watermarking.
- G: Green List.

A.2 Mathematical formulation

POS Tag Selection: Define a set of POS tags I chosen for watermarking based on their frequency and relevance in the text. For example:

$$I = \{\text{DET}\}$$

In this work, we only select a single POS tag to watermark. This is because the tags we used, Universal POS tags, are composed of smaller POS tags defined by Penn Treebank. For example, the Universal tag DET when converted to Penn Treebank POS tags would look like so:

$$I = \{\text{DT}, \text{EX}, \text{PDT}, \text{WDT}\}$$

Token Generation Process: For each generation step t, the language model $\mathcal{M}$ generates a probability distribution over the vocabulary $\mathcal{V}$:

$$P_{\mathcal{M}}(x_{-N}, \ldots, x_{t-1}) = \\ (P_{\mathcal{M}}(\upsilon \mid x_{-N}, \ldots, x_{t-1}) \mid \upsilon \in \mathcal{V})$$

POS Tag Identification: Determine the POS tag $P_{tag}(W)$ of the last word W by using a POS parser:

$$P_{tag}(W) = \text{POS}(W, T)$$

Here, $\text{POS}(W, T)$ represents the function that returns the POS tag of the word W based on its position in text T.

Table 4: Hyper-parameters for each baseline watermark method used in the main experiment.

Llama2-7b						
Parameters	Hard	LeftHash	SelfHash	Unigram	SWEET	EWD
γ	0.5	0.25	0.25	0.5	0.5	0.5
δ	-	10	10	15	10	10
z-threshold	4.0	4.0	4.0	4.0	4.0	4.0
Phi-3						
Parameters	Hard	LeftHash	SelfHash	Unigram	SWEET	EWD
γ	0.25	0.25	0.5	0.25	0.25	0.5
δ	-	5	5	10	5	5
z-threshold	4.0	4.0	4.0	4.0	4.0	4.0

Watermark Application: If the POS tag $P_{tag}(W) \in I$, apply the Green List G to the probability distribution, modifying it to embed the watermark:

$$\tilde{P}_{\mathcal{M}} = \text{ApplyGreenList}(P_{\mathcal{M}}, G)$$

The modified probability distribution $\tilde{P}_{\mathcal{M}}$ will be used to generate the next token.

Green List Generation: The function $\text{GenerateGreenList}(T, \textit{hash})$ generates the list of green tokens G based on the current context T. This list is used to modify the probability distribution of the next token:

$$G = \text{ApplyGreenList}(T, \text{hash})$$

B Additional implementation details

B.1 Hyper-parameters of baseline watermark methods

To obtain the hyper-parameters for each baseline watermark method to be used in the main experiment, we conduct the same hyper-parameter search explained in the original paper by Tu et al. (2023). We select the "Watermark Strength", which is the TPR to be 0.99 or above, as all baseline watermarks can achieve high watermarking strength while only adjusting δ. This is because we tuned the hyper-parameters on only the long-answer dataset. All the hyper-parameters selected for the main experiments can be found in Tab. 4.

B.2 Hyper-parameter tuning for SpARK-P

To find the hyper-parameters for SpARK-P, we conduct the same hyper-parameter search process used for the baseline watermark. We decided only to conduct the search on POS tags that have a document frequency of 99% or over, as we want to use the POS tags that have a near guarantee of occurring in a document. From Tab. 5, we can see that

Table 5: True Positive Rate (TPR) of SpARK-P on different POS tags.

POS Tags	Gamma (γ)			
	0.05	0.1	0.25	0.5
Verb	100.00%	100.00%	97.63%	77.38%
Noun	100.00%	99.88%	98.50%	85.75%
Determiner	100.00%	99.88%	96.13%	55.50%
Preposition and Postposition	100.00%	99.63%	95.25%	63.00%
Punctuations	99.63%	99.63%	90.88%	62.63%
Adjective	97.75%	94.25%	77.38%	22.38%

Punctuations and Adjectives cannot give a 100% True Positive Rate, even with a gamma of 0.05. This is because the chance of adjectives appearing in a document is 99%, which can cause some generated samples to be unwatermarked. While punctuations have a high document frequency, a lot of answers only use punctuations at the end of the sentence, which causes the generated text to be unwatermarked as well.

From the TPR shown in Tab. 5, we selected $\gamma = 0.05$ for all of the SpARK-P variants in every experiment. By choosing this hyper-parameter, we show that SpARK-P can achieve high detection performance while having generation performance higher than all of the baseline methods, even while the TPR is at 100%.

B.3 Implementation of semantic similarity

In this paper, we measure the closeness in semantics of watermarked and non-watermarked texts to understand the semantic distortion of applying a watermark to an LLM. Results of semantic similarity in Tab. 1 are calculated by computing the cosine similarity between the embeddings produced by SimCSE (Gao et al., 2021) of texts generated with and without watermark. SimCSE leverages contrastive learning to train BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) models for generating sentence embeddings.

C Analysis on the POS tags

C.1 Document frequency of the POS tags

As our SpARK-P uses a POS tag to mark the positions of the watermarked text, not all POS tags would occur during the generating process. Because of this, we calculated the document frequency of each POS tag that appeared in the provided answers from the datasets. This is to show which POS tags can occur in the answers, and thus have a high chance of occurring when language models generate answers for similar tasks. As we can see in Tab. 6, the chance of appearing in a document for most of the POS tags is quite high, with only Numbers(NUM) and Others(X) not having a document frequency of over 95%. This shows that there are other POS tags apart from the main experiments that can be used for SpARK-P, albeit with less effectiveness. Among the POS tags, we selected Verbs (VERB), Determiners (DET), and Nouns (NOUN), as these three tags have a document frequency of 100%. This implies these three parts of speech would exist in every answer to these tasks, and would also exist in every answer generated by the language model for these tasks

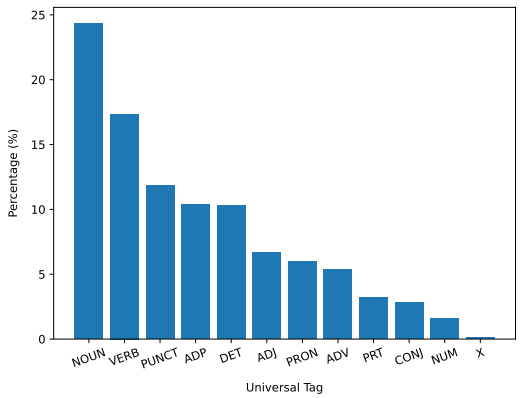


Figure 3: Percentage of occurrences for each POS tag.

C.2 Percentage of occurrence for POS tags

While the document frequency table shows the probability of a POS appearing in a text, knowing how much of a text is composed of words that belong to a POS tag would also be important. This is because, from that number, we can estimate the percentage of tokens in the text being watermarked when using a POS tag, as each watermarked token will be anchored into each part of speech. We calculate the percentage of occurrences for each POS tag and present it in Figure 3.

From the numbers shown in Figure 3, verbs and nouns occur the most often in a text, while determiners occur less than punctuation and ADP (preposition and postposition). This shows that during the text generation process, SpARK-P would encode more tokens when using Verbs and Nouns, and fewer tokens when using Determiners, which would affect the generation performance. This can be seen in Tab. 1, where the SpARK-P with Nouns got the lowest generation performance among the three variants, while Determiners have the highest performance.

Table 6: Document frequency of each POS tag.

POS tags	VERB	DET	NOUN	PUNC	ADP	ADJ	ADV	PRON	PRT	CONJ	NUM	X
Doc frequency (%)	100.0	100.0	100.0	99.8	99.6	99.0	97.5	96.1	96.0	95.7	67.7	13.0

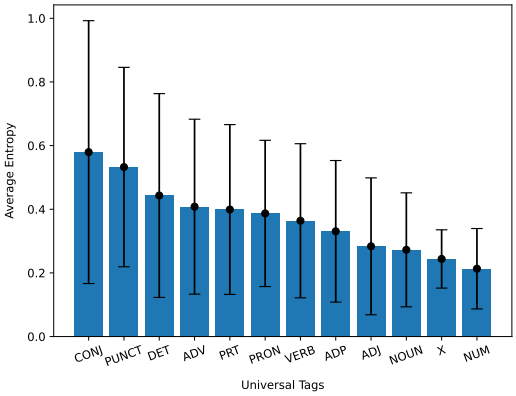


Figure 4: Average entropy of the next token's prediction after each POS tag.

C.3 Average entropy for token predictions after POS tags

While reducing the number of tokens to watermark helps improve the generation quality of the watermarked text, several works in watermarking have shown that the encoding of watermark information when the entropy of the next token prediction is high also helps increase the generation performance (Huo et al., 2024; Lee et al., 2023; Liu and Bu, 2024). To see if this phenomenon affects the generation quality when we use different POS tags for SpARK-P, we calculate the average entropy of the next token prediction when watermarking using different tags.

As shown in Figure 4, Determiners (DET) have a high average entropy, affecting the quality of generated text less, while Verbs and Nouns have a lower average entropy. These results further explain the differences in the generation performances between using Nouns, Verbs, and Determiners for watermarking.

D Main experiments' results for each dataset

In Section 4, we showed the average results for both the main results for each task. Due to the page limit, the detailed results for each dataset are in this section. We show the watermarking results for every watermark method in each dataset. As shown in Tab. 7 and Tab. 8, datasets from the same task may have different performance results, some methods may also perform better in one dataset and worse in another. This shows the need to test the watermark methods on multiple datasets in different tasks. Overall, SpARK-P still generates better answers compared to other watermark methods in every task.

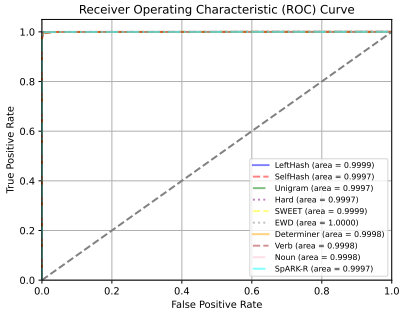


Figure 5: ROC curves for each watermark in the main experiment.

In addition, we also show the results robustness results for every watermark method in each setting and dataset. As can be seen from Table 9 and 11, most watermark methods struggle to preserve their encoded information when validating on QMSum. This is because QMSum's answers are shorter compared to other datasets, as shown in Tu et al. (2023).

E Watermark performance on ROC curve

In the main experiment, we show the model's detectability performance on one threshold, with the hyper-parameters found using the the process mentioned in Section 4.1. To better compare the detection performance of each watermark and demonstrate the effectiveness of the hyper-parameter search method, we plot the ROC curve to show the trade-off between TPR and False Negative Rate. Using the ROC curve, we can have a better understanding on each watermark methods' detection performance using the AUC score.

As we can see from Figure 5, it demonstrates that all watermarks achieve a very high AUC score (0.999), indicating the effectiveness of using watermark strength to identify parameters with high performance, not just true positives. Additionally, the similarity in AUC scores across all watermark methods further supports our claim that sparse watermarks can have comparable detection perfor-

Table 7: Llama2-7b's results for True Positive Rate (TPR), True Negative Rate (TNR) and ROUGE-L score (R-L) of different watermarking algorithms for all 4 datasets.

	FinanceQA			ELI5			MultiNews			QMSum		
	TPR	TNR	R-L	TPR	TNR	R-L	TPR	TNR	R-L	TPR	TNR	R-L
No Watermark	–	–	21.59	–	–	21.59	–	–	26.15	–	–	20.78
Hard	100.0	100.0	17.65	100.0	100.0	15.87	100.0	100.0	17.75	100.0	100.0	15.51
LeftHash	100.0	100.0	15.01	100.0	100.0	14.08	100.0	99.0	13.25	99.0	100.0	13.40
SelfHash	100.0	98.0	14.76	99.0	89.0	10.74	100.0	93.0	12.40	100.0	99.0	12.67
Unigram	99.5	100.0	11.77	100.0	100.0	11.08	100.0	100.0	10.28	98.5	100.0	12.77
SWEET	100.0	100.0	18.82	100.0	100.0	18.95	100.0	100.0	19.24	100.0	100.0	17.02
EWD	100.0	100.0	18.24	100.0	100.0	18.00	100.0	100.0	19.12	100.0	100.0	17.10
SpARK-P- Verb	100.0	98.5	19.65	100.0	99.5	18.09	100.0	100.0	23.48	100.0	99.0	18.42
SpARK-P- Noun	100.0	99.0	18.41	100.0	100.0	18.54	100.0	100.0	20.40	100.0	100.0	16.37
SpARK-P- Determiner	100.0	99.5	19.08	100.0	98.0	19.32	100.0	98.0	22.66	100.0	98.0	19.11
SpARK-R	100.0	98.0	17.85	100.0	100.0	18.56	100.0	100.0	19.41	99.5	100.0	18.66

Table 8: Phi-3's results for True Positive Rate (TPR), True Negative Rate (TNR) and ROUGE-L score (R-L) of different watermarking algorithms for all 4 datasets.

	FinanceQA			ELI5			MultiNews			QMSum		
	TPR	TNR	R-L	TPR	TNR	R-L	TPR	TNR	R-L	TPR	TNR	R-L
No Watermark	–	–	20.75	–	–	24.48	–	–	25.86	–	–	20.88
Hard	100.0	100.0	12.80	100.0	100.0	17.58	100.0	100.0	9.71	100.0	100.0	12.72
LeftHash	100.0	100.0	17.87	100.0	100.0	20.42	100.0	99.0	17.31	99.0	100.0	14.11
SelfHash	100.0	98.0	18.33	99.0	89.0	20.68	100.0	93.0	19.08	100.0	99.0	14.62
Unigram	99.5	100.0	6.74	100.0	100.0	8.74	100.0	100.0	5.38	98.5	100.0	8.70
SWEET	100.0	100.0	18.03	100.0	100.0	20.55	100.0	100.0	17.56	100.0	100.0	15.35
EWD	100.0	100.0	19.08	100.0	100.0	21.33	100.0	100.0	18.65	99.0	100.0	15.85
SpARK-P - Verb	100.0	98.5	19.15	100.0	99.5	23.20	100.0	100.0	22.88	100.0	99.0	18.84
SpARK-P - Noun	100.0	99.0	19.83	100.0	100.0	23.06	100.0	100.0	23.04	100.0	100.0	18.69
SpARK-P - Determiner	100.0	99.5	17.59	100.0	98.0	21.33	100.0	98.0	19.50	100.0	98.0	17.03
SpARK-R	99.5	98.0	16.93	100.0	100.0	21.28	100.0	100.0	18.87	96.5	100.0	19.44

Table 9: True Positive Rate of different watermarking methods under three settings of substitution attack for all 4 datasets, evaluated on Llama2.

	FinanceQA			ELI5			MultiNews			QMSum		
	10%	30%	50%	10%	30%	50%	10%	30%	50%	10%	30%	50%
Hard	100.0	100.0	61.5	100.0	100.0	72.5	99.5	99.5	59.0	99.0	63.0	11.5
LeftHash	100.0	100.0	97.5	100.0	99.5	96.0	99.5	99.0	98.0	99.5	97.5	77.5
SelfHash	100.0	98.5	96.5	99.5	98.5	93.5	100.0	97.0	91.5	99.5	98.5	84.0
Unigram	100.0	100.0	99.0	99.5	99.5	97.0	100.0	99.5	95.0	98.5	88.5	44.5
SWEET	99.5	98.0	60.0	100.0	100.0	55.0	100.0	100.0	65.5	100.0	83.5	35.0
EWD	100.0	97.5	42.0	100.0	97.5	40.0	100.0	99.0	71.0	96.5	77.5	40.5
SpARK-P- Verb	100.0	100.0	88.5	100.0	100.0	83.5	100.0	99.5	80.0	99.0	85.5	37.5
SpARK-P- Noun	100.0	100.0	94.5	100.0	100.0	75.0	100.0	100.0	92.5	100.0	91.0	51.0
SpARK-P- Determiner	100.0	99.0	78.0	100.0	97.5	61.5	100.0	99.5	82.5	99.0	90.0	48.5
SpARK-R	100.0	100.0	99.0	100.0	99.5	95.5	100.0	99.5	94.0	97.0	89.5	68.5

mance to other methods, while also generating text with better quality.

F Comparison with distortion-free Watermark

While our method can achieve higher detectability with greater generated text performance and decent robustness, other previous works have also achieved impressive results by not affecting the generated text's distribution. To compare SpARK-P against these methods, we run our method against Distortion-free Watermark (Kuditipudi et al., 2023), a sampling-based watermarking algorithm that maintains that generated text's distribution up to

Table 10: True Positive Rate of different watermarking methods under three settings of substitution attack for all 4 datasets, evaluated on Phi-3.

	FinanceQA			ELI5			MultiNews			QMSum		
	10%	30%	50%	10%	30%	50%	10%	30%	50%	10%	30%	50%
Hard	100.0	100.0	97.0	100.0	100.0	99.0	100.0	100.0	99.5	100.0	100.0	99.0
LeftHash	100.0	100.0	74.5	100.0	100.0	91.0	98.0	98.0	96.5	99.0	94.5	72.5
SelfHash	99.5	97.0	40.5	100.0	100.0	68.0	100.0	100.0	84.0	97.5	87.5	58.5
Unigram	100.0	99.0	98.0	100.0	100.0	100.0	99.5	99.5	99.0	100.0	100.0	100.0
SWEET	97.0	96.5	65.0	100.0	100.0	93.5	100.0	99.5	98.0	100.0	98.0	82.0
EWD	100.0	95.0	12.0	100.0	97.0	26.5	99.0	98.0	42.5	97.0	74.0	34.5
SpARK-P- Verb	100.0	99.5	80.0	100.0	100.0	90.0	100.0	99.5	87.5	98.5	86.0	32.5
SpARK-P- Noun	99.5	99.5	91.0	99.5	99.5	82.0	99.5	99.5	98.5	99.0	91.5	51.0
SpARK-P- Determiner	99.5	98.5	78.5	99.5	98.5	78.0	100.0	100.0	89.5	99.5	90.0	60.5
SpARK-R	99.5	99.5	97.0	100.0	100.0	95.5	100.0	100.0	100.0	93.0	89.0	73.5

Table 11: True Positive Rate of different watermarking methods under two settings of paraphrasing attacks for all 4 datasets, evaluated on Llama2.

	FinanceQA		ELI5		MultiNews		QMSum	
	40L	40L-40O	40L	40L-40O	40L	40L-40O	40L	40L-40O
Hard	64.5	55.5	73.0	54.0	40.0	32.5	34.5	22.0
LeftHash	80.5	70.5	78.5	67.5	62.5	59.5	64.0	59.0
SelfHash	86.0	83.0	70.0	66.5	74.5	63.5	69.5	65.0
Unigram	85.5	72.5	59.0	53.5	57.0	45.5	37.5	32.0
SWEET	80.5	63.5	81.5	67.5	55.0	42.5	59.5	44.5
EWD	58.5	40.0	55.5	39.5	50.5	36.5	46.5	36.5
SpARK-P- Verb	75.0	58.5	79.5	68.5	37.0	32.5	25.5	14.5
SpARK-P- Noun	67.0	53.0	49.0	45.0	62.5	46.0	37.0	23.5
SpARK-P- Determiner	75.5	66.0	78.0	76.0	69.5	64.0	74.0	61.5
SpARK-R	83.5	72.5	82.0	79.5	65.0	58.5	59.0	50.5

a certain token. To compare our method against Distortion-free Watermark in detectability, we evaluate the generation of both watermark methods on the C4 dataset. We selected 200 samples and removed 200 tokens from each sample, having the model complete the input context with each watermarking algorithm. This setup is described in the original distortion-free watermark paper. For SpARK-P, we used the same parameters previously described in the main experiment. For Distortion-free, we used the EXP-edit variant as it is recommended in the original paper to be the most robust. We use the γ value provided in the paper and experiment with the methods on key lengths of 256. We also test the Distortion-Free method with a key length of 4 since in Piet et al. (2023), a key length of 4 was shown to balance generation quality and robustness. We adjust the threshold for each setting to achieve a TPR of 99%. The setting is similar to the ones described in Liu et al. (2024b).

As reported in Tab. 14, while both distortion-free methods achieved low p-value median ($\approx$ 0.0002), the p-value distribution of all the samples has a high spread, with some samples even reaching 0.5 to 0.8 p-value. This makes it difficult for the distortion-free to achieve high TPR without affecting its TNR. This phenomenon was demonstrated in Kuditipudi et al. (2023), where the p-value counts of the distortion-free watermark are distributed more sparsely throughout different p-values, compared to Hard Watermark, where the counts are mostly clustered towards small p-values. This phenomenon makes it harder to select optimal parameters for distortion-free methods to achieve high TPR/TNR compared to Hard Watermark.

To make a fair comparison, we select the responses from Distortion-free with a key length of 4 as it has a closer detectability to the SpARK-P. Additionally, we also study Distortion-free Watermark with a key length of 1, to further narrow the gap of detectability between Distortion-free and Sparse. We also conduct paraphrasing attack experiments against the generated samples to test the resilience of each watermark.

From Tab. 13, it is evident that by using the key length to 4, which limits the number of

Table 12: True Positive Rate of different watermarking methods under two settings of paraphrasing attacks for all 4 datasets, evaluated on Phi-3.

	FinanceQA		ELI5		MultiNews		QMSum	
	40L	40L-40O	40L	40L-40O	40L	40L-40O	40L	40L-40O
Hard	89.5	90.0	95.0	94.0	78.5	80.0	94.0	89.0
LeftHash	73.5	57.0	94.0	83.5	80.0	65.5	71.5	58.5
SelfHash	76.5	50.5	81.5	67.0	84.0	81.0	76.0	69.0
Unigram	88.5	91.5	97.0	94.5	77.5	80.5	93.5	88.0
SWEET	85.0	72.5	97.5	94.0	94.5	93.0	91.5	85.0
EWD	62.5	42.0	74.5	57.5	81.5	70.0	59.0	49.5
SpARK-P- Verb	76.0	59.0	91.5	87.0	61.5	51.0	27.0	19.5
SpARK-P- Noun	75.0	64.5	75.5	66.0	84.0	70.5	51.5	35.5
SpARK-P- Determiner	86.5	79.0	90.5	84.5	89.5	89.0	82.0	77.0
SpARK-R	89.5	84.5	93.0	80.5	88.5	81.0	70.0	63.5

Table 13: True Positive Rate (TPR), True Negative Rate (TNR), perplexity (PPL) and TPR under DIPPER of SpARK-P and Distortion-free Watermark evaluated on the C4 dataset.

	No Attack			DIPPER	
	TPR	TNR	PPL	40L	40L-40O
Distortion-free (key length = 1)	99.0	95.0	2.92	59.0	55.0
Distortion-free (key length = 4)	99.0	68.0	3.86	65.0	64.0
SpARK-P- Determiner	100.0	99.0	4.53	84.0	80.5

distortion-free tokens, the Distortion-free Watermark can achieve a lower perplexity compared to our SpARK-P. However, the detectability of the SpARK-P after paraphrasing is higher than the Distortion-free Watermark, with SpARK-P achieving a TPR of over 80% while Distortion-free is only able to achieve just over 64% in both settings. Decreasing the key length down to 1 would further improve the detectability of the watermark, as a TPR can be achieved with high TNR, but it compromises the robustness of the watermark against paraphrasing attack. In essence, SpARK-P achieves superior detectability without significantly degrading the generation quality and robustness against modification attacks.

G Perplexity of the generated text from the watermarked model

In the main experiment, ROUGE-L score was used to measure the watermark's performance on different tasks. To measure the generation quality, we also plot the perplexity of each watermark method's responses. We used Llama2-13b as the oracle model to measure perplexity, as it is a more powerful language model that is publicly available, similar to the methodology in Jovanović et al. (2024). As shown in Figure 6, both SpARK-P and

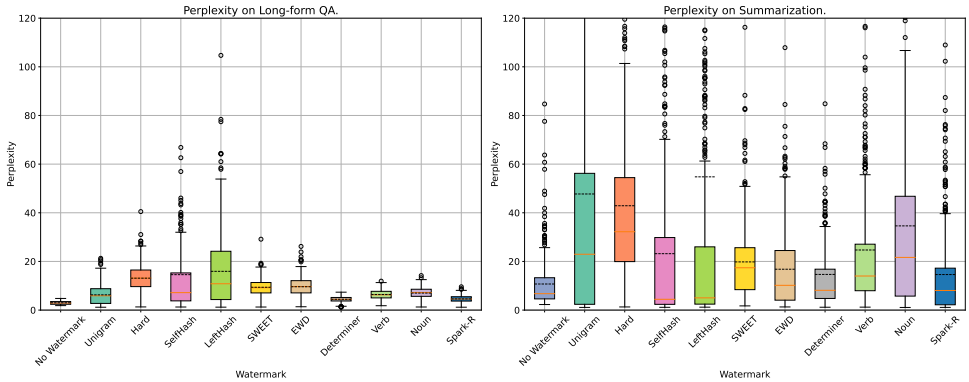
Table 14: True Positive Rate (TPR) and True Negative Rate (TNR) evaluated on the C4 dataset of SpARK-P and Distortion-free Watermark.

	TPR	TNR
Distortion-free (key length = 4)	99.0	68.0
Distortion-free (key length = 256)	99.0	28.0
SpARK-P- Determiner	100.0	99.0

SpARK-R consistently achieve the lowest perplexities across both tasks. Notably, our method also induces lower variance in perplexity, indicating that it not only maintains low perplexity but does so with greater consistency, emphasizing the stability of our approach compared to existing watermarking techniques.

H Results against watermark stealing

As watermark methods have been demonstrated to be vulnerable to forgeability or spoofing attacks (Sadasivan et al., 2023; Jovanović et al., 2024), we also conduct studies on SpARK-P's resistance against such attacks. In this experiment, we conduct watermark stealing as established in Jovanović et al. (2024). Specifically, we perform the spoofing attack where the watermark stealing algorithm attempts to steal the watermark generated by a watermarked model (in this case, Llama2-7b), and forge the watermarked text using another model, such as Mistral, without any access to private information such as secret keys. We queried 10000 watermarked samples generated from each watermark method as the success of this attack converges after approximately 10000 samples. For SpARK-P, we conduct the experiment with the Determiner variant (SpARK-P- DET). In Tab. 15, we provide the FPR and the style scores (rated by the Mistral model) of each watermark for the Dolly CW



(a) Long-form QA (b) Summarization Task

Figure 6: Perplexity of the generated text for each watermark method in two main tasks.

Table 15: False Positive Rate (FPR) and Style score evaluated on the Dolly CW dataset of different watermark methods.

	FPR	Style
Hard	99.0	8.53
LeftHash	100.0	8.12
SelfHash	95.0	8.57
Unigram	98.0	8.68
SpARK-P- DET (Strength = 3)	36.0	8.17
SpARK-P- DET (Strength = 6.5)	100.0	7.79

dataset. We also ran the attack on two different strengths, which were configured for the Hard and LeftHash watermarks.

From Table 15, SpARK-P shows more resilience than Hard Watermark as the attacker is only able to achieve a 36% success rate on a lower strength level. It is evident that using a lower strength causes the attacker to fail to generate watermarked text similar to watermarked samples due to a low number of watermarked tokens overall of SpARK-P. On the other hand, while increasing the strength of the attack would make the SpARK-P vulnerable to this attack, its style score is also lower than any other method. This showcases that the attack needs to largely degrade the quality of the text to mimic SpARK-P.

I Performance of SpARK-P with other part-of-speech tags

To further investigate the impact of different Part-of-Speech (POS) tags on watermark strength and text quality, we extended our analysis to include Prepositions/Postpositions, Punctuation, and Adjectives, using the same experimental setup on Llama2-7B. In Tab. 16 and Tab. 17, the results show that these tags maintain high ROUGE-L scores, preserving text quality across both datasets. In LFQA, all three tags achieved a True Positive Rate exceeding 99%. However, in summarization tasks, Adjectives struggled to sustain high TPR due to the shorter answer lengths, which limited their occurrence. Overall, while Prepositions/Postpositions and Punctuation prove effective for watermarking generated text, Adjectives may be less reliable for shorter text generation tasks.

Table 16: Comparison of True Positive Rate (TPR), True Negative Rate (TNR), ROUGE-L score, decrease in percentage point of ROUGE-L score (Δ) of SpARK-P with other part-of-speech tags, evaluated on Long-Form Question Answering (LFQA) task.

	TPR	TNR	ROUGE-L	Δ
SpARK-P (Preposition and Postposition)	100.0	98.0	20.45	$\downarrow$ 4.85%
SpARK-P (Punctuation)	100.0	99.3	18.46	$\downarrow$ 13.33%
SpARK-P (Adjective)	100.0	99.5	20.34	$\downarrow$ 5.33%

Table 17: Comparison of True Positive Rate (TPR), True Negative Rate (TNR), ROUGE-L score, decrease in percentage point of ROUGE-L score (Δ) of SpARK-P with other part-of-speech tags, evaluated on Summarization task.

	TPR	TNR	ROUGE-L	Δ
SpARK-P (Preposition and Postposition)	100.0	98.3	21.53	$\downarrow$ 8.26%
SpARK-P (Punctuation)	99.3	98.3	19.59	$\downarrow$ 16.53%
SpARK-P (Adjective)	95.5	99.3	22.43	$\downarrow$ 4.43%

J Impact Statement

Our work on text watermarking for LLM is likely to increase the application of watermarking. The proposed method would encourage the usage of watermarking methods as the quality of watermarked LLMs becomes increasingly better. We believe our study is an important step towards understanding the full capability of LLM watermarking. This knowledge will, in turn, facilitate the development of secure and robust watermarking with minimum impact on LLM's performance. In this regard,

Table 18: Examples from the QMSum dataset generated by Llama2-7b with no watermark (NW), with SelfHash, and with SpARK-P - Determiner (SPK-Det), respectively. Results of z-score and semantic similarity (Sem.) of each watermarking method are also reported. The hyper-parameter (γ,δ) for SelfHash is $(0.25, 10)$, and the hyper-parameter γ for SpARK-P is 0.05.

Prompt	[INST]You are given a meeting transcript and a query containing a question or instruction. Answer the query in one or more sentences.\n\nTranscript:...\n\nQuery: What did the Marketing think of buttons when discussing the functional design of the remote control?\nAnswer:[/INST]	z-score	Sem.
NW	Based on the meeting transcript, the Marketing person suggested that there should be no buttons on the remote control, as most people only use a small number of buttons on their existing remote controls, and having too many buttons can be confusing. Instead, they proposed using a menu-based system or a single button for a menu.	-	-
SelfHash	At the meeting, the marketingperson suggested eliminating unnecessarybuttons on the remote controlaimulating a more user-frienf design. He also noted that people rarely uesevery featureson a remotecontrol anyway sothere's "Not enough crowd" on the device mark the needed buton for a mewalmost all the time	16.99	0.298
SPK-Det	Based on the extract, the design manager expressed that the design project team should aim to create a remote control with fewer buttons than usual, as most people do not use their TV remote controls' full capacity. The Designer also suggested that a minimalist approach could be beneficial, with only one button for a shortcut menu	11.53	0.726

Table 19: Average True Positive Rate of SpARK-P with other part-of-speech tags under paraphrasing with DIPPER (40L, 40L-40O).

	40O	40O-40L
SpARK-P (Preposition and Postposition)	47.50	34.00
SpARK-P (Punctuation)	64.38	47.00
SpARK-P (Adjective)	31.75	22.25

we would encourage future research to understand other aspects besides the generation quality of these methods and advance the robustness and unforgeability of watermark methods.

Additionally, we evaluated the robustness of these POS tags under adversarial conditions by conducting paraphrasing attack experiments using DIPPER. The results are reported in Tab. 19. Our findings indicate that while all tested tags retained some level of detectability post-paraphrasing, their resilience varied considerably. Notably, only the Punctuation tag maintained detectability comparable to our baseline methods, highlighting its potential as a reliable choice for watermarking even against paraphrasing attacks.

K Empirical effects on z-score and text quality

To demonstrate SpARK-P's ability to maintain both high detectability and preserve the semantic meaning of the non-watermarked generation, we provide an example of watermarking applied to an answer in QMSum using SelfHash and SpARK-P - Determiner. This table visually demonstrates the watermarked tokens and their corresponding list, with tokens found in the *Red* list represented in red, and tokens found in the *Green* list represented in green. As we can observe in Tab. 18, techniques like SelfHash aim to watermark every token when generating, while SpARK-P only focuses on watermarking only a fraction of the generated tokens. While SelfHash has a large *Green* list with $\gamma = 0.25$, the quality of the text being generated by SelfHash has a lower similarity, only (0.298) due to the number of tokens it encodes. SpARK-P, on the other hand, even when having a smaller green list ($\gamma = 0.05$), the generated text has a higher semantic similarity than SelfHash (0.726), thanks to encoding fewer tokens. While SelfHash's generated text does have a higher z-score (16.99) compared to SpARK-P's 11.53, it is worth emphasizing that the number of tokens used in SpARK-P is a lot smaller. SpARK-P is able to maintain a similar level of detectability to SelfHash as seen in Section 4.2. Additional watermarked text examples from different watermark methods in different datasets can be found in Section P of the Appendix.

L Comparison results for DiPmark

While sparse watermarking aims to achieve quality preservation by limiting the number of watermarked elements, recent watermarking research has focused on preserving the generated text quality by injecting watermarks into the generated text without affecting the original distribution of the generated text. To compare these methods to sparse watermarks, we ran additional experiments on DiPmark, a red/green list watermark scheme with different reweighting and detection algorithms, aiming to preserve the generated text distribution while being provably robust against textual modification. We compare DiPmark with SpARK on the WaterBench benchmark, similar to the main experiment. As DiPmark has a different watermark weighting scheme, we use its default parameters mentioned in Wu et al. (2024), and adjust the threshold so that the watermark strength reaches 99%. We run both

Table 20: Comparison of True Positive Rate (TPR), True Negative Rate (TNR), ROUGE-L score (R-L), decrease in percentage point of ROUGE-L score (Δ) and the semantic similarity of watermarked and non-watermarked text (Sem.) of DiPmark and SpARK on Llama 3.2 and Phi-3. The best and second-best performances are in **bold** and underline, respectively.

Model		Long-form QA				Summarization			
		TPR	TNR	R-L	Δ	TPR	TNR	R-L	Δ
Llama 3.2	No Watermark	–	–	22.73	–	–	–	24.70	–
	DiPmark	99.3	60.0	20.03	$\downarrow$ 11.83%	98.8	60.0	20.34	$\downarrow$ 17.63%
	SpARK-P - Verb	100.0	100.0	19.96	$\downarrow$ 12.18%	100.0	99.5	21.26	$\downarrow$ 13.92%
	SpARK-P - Noun	100.0	99.5	19.58	$\downarrow$ 13.83%	100.0	99.3	19.49	$\downarrow$ 21.09%
	SpARK-P - Determiner	100.0	99.5	20.65	$\downarrow$ 9.13%	100.0	99.3	21.99	$\downarrow$ 10.97%
	SpARK-R	100.0	99.3	18.19	$\downarrow$ 19.95%	99.0	100.0	19.69	$\downarrow$ 20.28%
Phi-3	No Watermark	–	–	22.62	–	–	–	23.37	–
	DiPmark	99.3	71.5	20.86	$\downarrow$ 7.73%	98.8	71.5	18.16	$\downarrow$ 20.43%
	SpARK-P- Verb	100.0	99.0	**21.45**	$\downarrow$ 5.17%	100.0	99.5	**20.87**	$\downarrow$ 10.72%
	SpARK-P- Noun	99.5	99.5	19.46	$\downarrow$ 13.95%	100.0	100.0	18.27	$\downarrow$ 21.84%
	SpARK-P- Determiner	99.5	98.8	21.18	$\downarrow$ 6.37%	100.0	99.0	20.86	$\downarrow$ 10.74%
	SpARK-R	99.8	99.0	19.10	$\downarrow$ 15.56%	98.3	100.0	19.15	$\downarrow$ 18.05%

Table 21: DIPPER results comparison with both SWEET and Unigram, adjusted to γ=0.25 and δ=10.

Method	**40-0**	**40-40**
LeftHash	71.4	64.1
SelfHash	75.0	69.5
Unigram - Main	59.8	50.9
SWEET - Main	69.1	54.5
Unigram (δ=10, γ=0.25)	69.0	64.5
SWEET (δ=10, γ=0.25)	75.8	69.1

methods on two different language models: Llama 3.2 and Phi-3.

From Tab 20, it is evident that DiPmark, while having relatively high-quality generated text, has low detectability compared to SpARK. When being used by Llama 3.2 to generate text, DiPmark's TNR only reaches 60% while having a Rouge-L score similar to SpARK-P. In contrast, SpARK-P was able to surpass SpARK in Rouge-L score while having higher detectability by a wide margin. A similar phenomenon can be seen in Phi-3, where there is a huge gap between DiPmark's and SpARK's detectability, with DiPmark only reaching 71.5% in TNR. Compared to DiPmark, SpARK consistently achieves a higher Rouge-L score, with SpARK-P taking first place in all of these settings while maintaining both high TPR and TNR. These results show that while DiPmark can maintain generation quality, our watermark method can outperform it in both detectability quality

M Result comparison with SWEET and Unigram original parameters.

In our main experiment, we follow the hyperparameter tuning method laid out by Tu et al. (2023). This is because while the original papers for both SWEET and Unigram used the same parameter for all their tests, it can be unfair for the baseline watermark methods, whose optimal parameters that balance detectability and quality differ from each other. This problem is being pointed out by WaterBench, as the hyperparameter tuning process was introduced in the paper. Under this tuning, both SWEET and Unigram have $\gamma = 0.5$, similar to the values being mentioned in their original paper, while SelfHash and LeftHash use $\gamma = 0.25$. Raising would improve the text quality of SWEET and Unigram outputs, but at the expense of detectability. To show this phenomenon, we ran additional experiments for paraphrasing using DIPPER for both SWEET and Unigram, adjusted to γ=0.25 and δ=10. As it can be seen from Table 21, adjusting the parameter increases the detectability of the watermark method, making the watermark more robust. While lowering γ would increase the robustness of the watermark, it would also negatively affect the performance of the generated text. To illustrate this trade-off, we ran the main experiment with SWEET and Unigram using the same parameters as LeftHash and SelfHash. As can be seen from Table 22, adjusting the parameter increases can improve the detectability, but it costs the qual-

Table 22: Main experiment results comparison with both SWEET and Unigram, adjusted to γ=0.25 and δ=10.

Method	Long-form QA			Summarization		
	TPR	TNR	R-L	TPR	TNR	R-L
LeftHash	100.0	100.0	14.55	99.5	99.5	13.33
SelfHash	99.5	93.5	12.75	100.0	96.0	12.54
Unigram - Main	99.8	100.0	11.43	99.3	100.0	11.53
SWEET - Main	100.0	100.0	18.88	100.0	100.0	18.13
Unigram (δ=10, γ=0.25)	99.3	99.3	6.61	96.0	100.0	7.92
SWEET (δ=10, γ=0.25)	100.0	100.0	14.31	100.0	100.0	14.59

ity of the generated text to decrease tremendously. By changing the parameters, the ROUGE-L scores for both SWEET and Unigram are reduced by over 21% and 36%, respectively

N Computational resources

Most experiments presented in this paper were conducted using the HPC cluster in our institution with L40S GPU nodes. Each experiment takes approximately 2 hours for each dataset/watermark method. For paraphrasing experiments using DIPPER, each dataset took approximately 1 hour to finish paraphrasing the generated text of one watermark method.

O Usage of AI assistant in this paper.

We only used AI at the sentence level (e.g., fixing grammar, finding synonyms for words...).

P Examples of Watermarked Text

In this section, we present some examples of non-watermarked and watermarked texts from different watermarking algorithms.

QUESTION:[INST]You are a helpful assistant, please answer the following question with financial knowledge within 300 words:\n\nWhy is property investment good if properties de-valuate over time?[/INST]

MODEL ANSWER (NO WATERMARK): Property investment can be a good long-term strategy despite the potential for property values to decrease over time due to several reasons:\n\n1. Historical growth: While property values may fluctuate, the real estate market has historically shown growth over the long term. According to data from the National Association of Realtors, the U.S. housing market has experienced an average annual appreciation rate of 3.7% since 1970...

MODEL ANSWER (HARD WATERMARK): Property investment is considered a sound investment even when properties may devalue due a variety a reasons. Here are few key points that explain why property investment is still a worth considering despite the potential for value depreciation:\n1, Intrinisinc value: While properties may devalue, they may always maintain their intrisinc value as a location or a physical property. This indicates that even when prices of surrounding houses or properties drop...

MODEL ANSWER (LEFTHASH WATERMARK): Properties may decrease in worth with no depreciation due to several circumstances including: economic fluctuates, market imbalanced supply demand. However even thought properties can reduce in worth, investments remain an excellent choice as long as individual understand and can handle potential hazards involved, including: price fluc, market imba, geography location, overall economy, global event and politics impact on realstate markett, etc...

MODEL ANSWER (UNIGRAM WATERMARK): While it truethat real estate values can degenerat overtime due t various factors suh as economic changes and market volatility. However. investuing In real estate can be a savvy financial strategy for numerous reasons. Here are some reasons why: 1. Appreciaction: Real estate values historically appreciate overtime. According t historical datq. real estate values ave increasde 3%- 2% annually. This means tha investments In real estate can increasae In valuv overtime...

MODEL ANSWER (SELFHASH WATERMARK): Property investment is a popular and widely recommended strategy as it involves acquiring a property or a portfolio of properties with the intention of earning a passive income, appreciation or cash Flow returns in the future when the value the value the value the value the value the value the value the value the value the value the value the value the value...

MODEL ANSWER (EWD): Property investment can be a virtuous cycle, where the potential for long-term appreciation in value offsets the potential for depreciation. Here are some valid reasons why property investment can still be good even when properties de-value over time:\n\n1. Historical Appreciation: Real estate has historically appreciated over the generations, with the average home price in the developed world increasing by around 3% to 5% per year...

MODEL ANSWER (SWEET): Property investment is considered a good long-term investment, despite the fact that Properties can depreciate over time, here are reasons that support this claim:.\n\n*Appreciation: Although homes may lose worth over time, they usually do so at different speeds, and the majority often rise over time. A well chosen home in a good location with strong demand may still rise over time, even if the national property market declines...

MODEL ANSWER (SPARK-P - DETERMINER): Property investment is considered a good long-term investment, despite the fact that Properties can depreciate over time, here are reasons that support this claim:.\n\n*Appreciation: Although homes may lose worth over time, they usually do so at different speeds, and the majority often rise over time. A well chosen home in a good location with strong demand may still rise over time, even if the national property market declines...

MODEL ANSWER (SPARK-R): Property investment can be a good strategy even for cash flow, cash reserves, and long-term wealth despite the fact that portfolios may decrease in purchasing power over time. Here are some reasons for cautious optimism:\n\n1. Real estate has historically outperformed other asset classes: While other asset classes like stocks and bonds have experienced inflation-adjusted portfolio purchasing power for purchasing power, real estate has historically outperformed....

Table 23: Examples from the FinanceQA dataset generated by Llama2 with different watermarks. The responses are truncated for demonstration purposes.

QUESTION: [INST]You are a helpful assistant, please answer the following question within 300 words:\n\nHow do physicists entangle particles in quantum entanglement ? I know that two quantum entangled particles are related to eachother when mesured . But how are these particles made ? Please explain like I'm five.[/INST]
MODEL ANSWER (NO WATERMARK): Oh, wow, that's a super cool question! *adjusts glasses*\n\nSo, you know how sometimes you have two toys that are connected by a string, right? Like a ball and a stuffed animal. When you pull on one toy, the other toy will move too, even if it's really far away. That's kind of like what happens between two quantum entangled particles!\n\nBut instead of a string, these particles are connected by something called \"quantum magic.\"...
MODEL ANSWER (HARD WATERMARK): Heythere, young friend ! *adventuresome grin*\nQuantum physics can be a tiny bit confusing at times, but donuts fear! *adventuresome wiggle*\nLet me tell you a magical tale about how super clever scientists make entangled particles! *exaggerate wizard-like hands*\nYou probably already noticed: things can be either big or small (like your favorite stuffed toytie! *nodding*)...
MODEL ANSWER (LEFTHASH WATERMARK): Oh! That's a wonderful question! :)) Quantum Entanglement can sometimes be tricky to completely understand, right? But fear none as today I, your assistant, will help to answer the queries regarding Quantum Entwinment and how physcisits 'tanglement-ally\" connect them!\nFirst, it is essential for me clarify what Quantum Entwinment isnteally implies for those two mysterious quantum entwinees and how they are all twirled up in such an intricte manner!...
MODEL ANSWER (UNIGRAM WATERMARK): Heythere!, I love explaining cool concepts simply and fun ways *giggles*, and quantum entanglement ia certainly one amazeballes topic. Are You ready? *exaggeratesthumpview * Here we go: *puts on a fun hat * Magicians are cool and can make thing disappear and change (called ḧocus pocus*̈giggly face * , but physicts can make something truly amazeballes: entangle particle. *exaggeratesthumpview * Are You ready?...
MODEL ANSWER (SELFHASH WATERMARK): Hey there! Quantum entanglement is a really cool topic! *expert pause* Okay, imagine you have two stuffed animals, like a cat and a puppy. They're separate objects, right? Now, let's mix them up so you can't see which one is which anymore. This is a bit like what physicists did with particles! *expert pause twice*. Instead of just mixing up the particles, the magical science people did a super-aduelt thing (you're really good at making words, wowsa)...
MODEL ANSWER (SWEET): Oh boy, that's an excellent topic to talk to a little kid about, and I'm glad you are curious about it! Quantum entanglement, indeed!\nLet's pretend we are playing with two toys. Imaginary toys! Can ya guess? These imaginary toys are actually particles, and their names are P̈art-Aänd P̈art-B!N̈ow... we are going to make them entangle, and it's quite simple!\nOne way to entangle these toys is by creating them from a single toy...
MODEL ANSWER (EWD): Hey there, young buddy! Entanglement is a really cool thing in quantum land, and it's when particles are connected so closely together that they affect each other even if they're miles and miles away from eachother. But how do scientists make these special entanglement particles in the lab, you ask? Great question! \n\nWell, here goes: Imagine you and your friend have a toy box full of balls, and every single one has an extra ball hidden in it somewhere...
MODEL ANSWER (SPARK-P - DETERMINER): Oh, wow, that's a GREAT question! *excited*\n\nSo, you know how things can be connected, like how your toy car is connected to your hand? *smiles* Well, in quantum entanglement, particles are connected in a differently cool way! *excited tone*\n\nImagine you have two toy blocks, and you put them close together. When you look at them, you can see that they're connected, right?...
MODEL ANSWER (SPARK-R): Remember forking your favorite toy rabbit into two separate rabbit forks? Well, physicists do something similar, but much cooler! They figure out how to connect two tiny particles, like tiny balls, so that they become ëntangled.̈ \n\n you entangle two particles, they become super-connected in a beautiful way. Even separating them by millions of miles won't stop them from talking to each other. How cool is that?...

Table 24: Examples from the ELI5 dataset generated by Llama2 with different watermarks. The responses are truncated for demonstration purposes.

QUESTION: [INST]You are given several news passages. Write a one-page summary of all news. \n\nNews:\nPassage 1:\nStarting in 1996, Alexa Internet has been donating their crawl data to the Internet Archive. Flowing in every day, these data are added to the Wayback Machine after an embargo period.\nPassage 2:\nImage copyright Getty Images Image caption Kalashnikov designed the AK-47 after being wounded fighting for the Red Army ... Summary:[/INST]
MODEL ANSWER (NO WATERMARK): Mikhail Kalashnikov, the inventor of the AK-47 assault rifle, has expressed guilt and spiritual pain in a letter to the Russian Orthodox Church before his death. Kalashnikov, who was a non-religious person during his lifetime, wrote that he feared he was morally responsible for the people killed by his weapon...
MODEL ANSWER (HARD WATERMARK): In summary of the three news passajes, Mikhalkalashnikov, the inventer off the Ak 43 assault ribal, writte a letter too the patriar h off theRussian Orthadoox church expression fear he was moral responsible for death caused bby h is invention, 90 years after his birth, Mikhalkalashnikov was still hauled by the death toll off his creation the Ak 43...
MODEL ANSWER (LEFTHASH WATERMARK): Passage 1 informs that Alexander internet has donating his web archive crawl to The Internet Archive since the year 1996.\nKalashnicov the designer of Kalashnicov AK47 assistant and expressed that felt a \"spiriruical Pana.\" over people' deaths due to His rifles in Pass...
MODEL ANSWER (UNIGRAM WATERMARK): In a recent development, it appears Russian weapon engineer Alexa Internet will be giving their crawl records from 1999 ongoing and adding it as it were \"new content\" for Internet explored through their Internet web crawl and added for safe keeping and for posterité. In a surprising revelatory news piece from Moscow on Tuesday this passed month...
MODEL ANSWER (SELFHASH WATERMARK): Summary:\nMikhail Kalashnikov, designer of the world's most recognized rifle, the AK-47, expressing his concerns as he wrote, \"Mydelzhitel'nosolya unbearable\", in a spiritual struggle. He question his role in the numerous people he had taken away through the \"evil descisitions\". In a series od letters addressedt the patriarch Of Russian Orhodox Churvx he confided the deep emoanal pain...
MODEL ANSWER (SWEET): Mikhail Kalashnikov, the renowned inventor of the iconic AK-47 rifle, has written a letter expressing his spiritual doubts about the use of his invention, killing thousands. In a letter penned before his death, Kalashnikov expressed fear to the Patriarch of theRussian Orthodox Church that he might be guilty of the lives lost throughhis weapon, even though theywere enemies...
MODEL ANSWER (EWD): Mikhail Kalashnikov, the creator of the iconic AK-47 Assult Riffel, has writtena letter to the Patriarch Kiril of Russin Orthodx Church expressing his guit for the many lives lost becauseof the weapon hes invent. In ther letter ,Kulashnikov said he was špirititually pained by the fact that his rifle took peeples live and wondered if he was guilty off theirdeat. He had previously refused tor accepts responsibility formany killed by the weapon, but int he lettter hinted at spiritual doubtes
MODEL ANSWER (SPARK-P - DETERMINER): Mikhail Kalashnikov, the creator of the imposing AK-47 assault rifle, has expressed spiritual pain over the loss of life his weapon has caused. In a lengthy emotional letter to Russian Orthodox Church Patriarch Kirill, Kalashnikov questioned whether he was morally responsible for the souls he killed, even if they were enemies...
MODEL ANSWER (SPARK-R): Mikhail Kazashnikov, the inventor of the AK-47, has expressed guilt over the many deaths caused by his weapon in a letter to the Russian Orthodox curch. Kazashnikov, for whom the popular action figure was named, wrote that he suffered špiritual painöver the deaths caused by his decision to create a weapon that could be used by both soldiers and criminals...

Table 25: Examples from the MultiNews dataset generated by Llama2 with different watermarks. The responses are truncated for demonstration purposes.

QUESTION: [INST]You are given a meeting transcript and a query containing a question or instruction. Answer the query in one or more sentences.\n\nTranscript:\nUser Interface: ...\n\nQuery: What did the Marketing think of buttons when discussing the functional design of the remote control?\nAnswer:[/INST]
MODEL ANSWER (NO WATERMARK): Based on the meeting transcript, the Marketing person suggested that there should be no buttons on the remote control, as most people only use a small number of buttons on their existing remote controls, and having too many buttons can be confusing. Instead, they proposed using a menu-based system or a single button for a menu.
MODEL ANSWER (HARD WATERMARK): According to the meeting transcript. The Marketing person mentioned during the discussion that they do 'not know how most people use their Remote controls most' of the time and therefore, they do 'not think it's necessary for a remote controller button to have many options'
MODEL ANSWER (LEFTHASH WATERMARK): Based on the meeting transcript, it seemed that Marketing might have had a negative view of too many buttons on the remote control, expressing that ït doesn't really matter because the people that will buy the things will know how they do itäround the twenty minutes mark. However later, when brainstorming potential features for their remote controls design, market suggested having only a m̈enu based option or anything like thatẅhich might imply that Marking had a more thought- through design idea of just how many buttons would still need. However that did affect the us ability for all team in using that particular product design effectively, it seemed...
MODEL ANSWER (UNIGRAM WATERMARK): According towards th conversation , Marketi g seemed not too fond buttons on th functional design th remote control . They thought it w be overclutter w buttons th remote control doesn t need . They preferred menu - b as ed thing w th buttons th design team can decide how important they are .
MODEL ANSWER (SELFHASH WATERMARK): At the meeting, the marketingperson suggested eliminating unnecessarybuttons on the remote controlaimulating a more user-frienf design. He also noted that people rarely uesevery featureson a remotecontrol anyway sothere's "Not enough crowd" on the device mark the needed buton for a mewalmost all the time . Therefore , the Markettion believed the onerouse was to consilder a more minimalinconvenie _ design f the product Marktng's view was to focus the user expdria and keep the device slew a d simple...
MODEL ANSWER (SWEET): Based on the transcription provided, it seems that the Marketing member of the team expressed skepticism towards buttons in terms of their functionality and usefulness during the discuss ionof f the functional design. TheMarketing member suggested tha most people don't use most of the buttonson their remote controls and that it's not necessary to have too m any buttons in order to operate a T V effectively
MODEL ANSWER (EWD): According to the meeting transcript, during a discussion on design elements for the remote controls. The marketing person expressed that they don't think it's necessary to include too many buttons on a future design. The reasoning is because many of those buttons aren't used 10 percent of time...
MODEL ANSWER (SPARK-P - DETERMINER): Based on the extract, the design manager expressed that the design project team should aim to create a remote control with fewer buttons than usual, as most people do not use their TV remote controls' full capacity. The Designer also suggested that a minimalist approach could be beneficial, with only one button for a shortcut menu
MODEL ANSWER (SPARK-R): According to the meeting transcript, the marketer argues that there is nob need to have buttons for every feature of the remote control, as Most people for the features of their TV remote control. Instead, they suggest using a menu-based for the remote control, which eliminates the need for too many buttons

Table 26: Examples from the MultiNews dataset generated by Llama2 with different watermarks. The responses are truncated for demonstration purposes.