Stealthy Reasoning Protection: Preventing Unauthorized Transfer of LLM Reasoning Ability

Anonymous ACL submission

Abstract

Large language models (LLMs) have demonstrated remarkable capabilities in solving complex problems through step-by-step reasoning. Recent studies have reported that this reasoning ability can be transferred to small language models (SLMs) by fine-tuning them with rationales generated by LLMs. Considering that training LLMs requires large amounts of data and computational resources and entails substantial value, this transferability raises significant concerns regarding intellectual property protection. Malicious users may exploit these APIs by querying them to obtain high-quality responses, which can then be used to enhance the reasoning capabilities of their own models. Such illegal practices undermine the intellectual property associated with the reasoning ability of the original models. In this paper, we investigate how to prevent the transfer of reasoning abilities in such a query-based "stealing" process. Our approach focuses on manipulating the outputs of LLMs while ensuring that legitimate users can still access these outputs without disruption. To achieve this, we propose a Unnoticeable Reasoning Editing(UREdit) that embeds imperceptible characters into the LLMs outputs, thereby preventing the transfer of reasoning ability. Furthermore, given that the queries are streamed sequentially through LLMs, we propose a Sample- and Token-Level Selection to improve the imperceptibility of the edits. Extensive experiments validate the effectiveness of our method across various datasets and models, and further analyze for why our method works.

1 Introduction

Existing studies have demonstrated that prompting Large Language Models (LLMs) to generate intermediate reasoning steps enhances their ability to solve complex problems. This reasoning capability enables models to tackle tasks in domains such as mathematics, symbolic reasoning, and commonsense understanding (Kojima et al., 2022; Rae et al., 2021). Specifically, by providing instructions to LLMs, such as *"Let's think step by step."*, they can generate a step-by-step reasoning process. This allows them to break down complex problems and build a reasoning path toward the final answer. Additionally, by leveraging intermediate steps generated by large models, reasoning capabilities can be transferred to Small Language Models(SLMs) (Ho et al., 2023), enabling small models to solve complex problems effectively.

The transferability mentioned above has raised concerns among large model companies regarding the protection of reasoning data (Hill, 2022; Birch et al., 2023). Unauthorized parties can avoid the financial costs in manually constructing reasoning pipelines or datasets. By only using the rationales from the outputs of large models, they can train their smaller models more effectively. These smaller models can even achieve better performance on downstream tasks. This practice poses a significant threat to the intellectual property of developers. Therefore, it is necessary to protect the reasoning responses of large models.

Although some previous works have recognized the importance of model and data copyright, they primarily focus on copyright protection in the image domain or on detecting generated content in text, without addressing how to prevent the stealing of reasoning abilities. For example, Text watermarking (Liu et al., 2024) can verify the outputs of LLMs but are unable to prevent unauthorized parties from learning knowledge from these reasoning outputs. While existing unlearnable data methods often rely on modifying training inputs or labels to prevent model learning, such changes are not practical in our scenario (Nguyen et al., 2022). In this setting, user queries are unpredictable and uncontrollable, requiring model responses to be both factually accurate and logically consistent for a satisfactory user experience. Therefore, it is important

to prevent the transferability of reasoning abilities and protect the intellectual property of model owners in terms of their reasoning capabilities.

Thus,in this paper, we attempts to address the following question: "*In the case of streaming queries, how can editing reasoning responses prevent the transfer of reasoning abilities?*" In this context, we face two key challenges. (1) How can the reasoning responses be edited in a way that is imperceptible to ordinary users, while preventing unauthorized parties from transferring the model's reasoning capabilities into their smaller models. (2) How can we make our protection methods more stealthy in the setting of streaming queries.

To address the above-mentioned challenges, we propose Unnoticeable Reasoning Editing(UREdit) that prevents the transfer of large model capabilities by embedding imperceptible characters into the reasoning responses. Moreover, to enhance the imperceptibility of the edits, we introduce a Sample- and Token-Level Selection in the context of streaming queries. Importantly, this strategy adds almost no extra computational cost, as it is based only on the natural features of the model's output content.

We evaluate our method across various reasoning tasks and different small language models. Extensive experiments demonstrate that our UREdit effectively limits the transfer of reasoning capabilities. For instance, when fine-tuning T5-large using original reasoning responses versus edited ones, the accuracy drops from 42.73% to 24.87%. Further analysis of the outputs generated by T5-large reveals that the reasoning paths induced by the edited responses are significantly more complex, causing the smaller model to become "lost in the path" during inference. Our contributions are as follows:

- We propose Unnoticeable Reasoning Editing that protect reasoning outputs generated by LLMs without affecting ordinary users.
- We propose Sample- and Token-Level Selection, which enhances the imperceptibility of the edits while introducing almost no additional computational overhead.
- We validate the effectiveness of our model across different datasets and models, and provide an explanation from the perspective of Monte Carlo Search Trees.

2 Related Work

Transfer of reasoning ability. Previous work has reported that Chain of Thought (CoT) requires extremely large models to achieve optimal performance (Hoffmann et al., 2022). However, recent research indicates that fine-tuning small models with the rationales generated by large models can also enable small models to acquire some reasoning capabilities. Ho et al. (Ho et al., 2023) proposed Fine-tune-CoT, which enabled chain-of-thought reasoning in small LMs. The core idea is to generate reasoning samples from very large teacher models using CoT prompting and subsequently fine-tune small student models using the generated samples. Li et al. (Li et al., 2022) used the explanations of LLMs for multi-task learning, significantly improving the performance of small models. In contrast to these works, Magister et al. (Magister et al., 2022) and Shridhar et al. (Shridhar et al.) explored more teacher models and demonstrates both the effects of dataset and model size on accuracy. Furthermore, Kang et al. (Kang et al., 2023)utilized documents retrieved from external knowledge bases to enhance small-scale LMs, enabling the models to leverage the knowledge to generate better rationales and consequently arrive at correct answers. The studies mentioned above disclose the possibility and threats of reasoning transfer from LLMs to SLMs and raise an important question: How can large model providers safeguard the rationales contained in their model outputs?

Text Watermark. Text watermarking involves embedding unique and imperceptible identifiers (watermarks) into the text content. The design of text watermarks is robust yet unobtrusive, ensuring that the integrity of the content is maintained without affecting its readability or meaning (Liu et al., 2024). With the advancement of LLMs, the capabilities of LLMs in understanding and generating language have been significantly enhanced. However, powerful LLMs are vulnerable to model extraction attacks, where attackers extract a large amount of data to train new SLMs (Birch et al., 2023). Adding watermarks to the texts generated by LLMs can protect the LLMs' outputs.

Existing text watermarking techniques can be categorized, according to the training process of LLMs, into the logits generation stage, the token sampling stage, and the model training stage. Watermarking during the logits generation (Brown et al., 2020) refers to inserting watermark informa-

tion into the logits generated by the LLM. When adding a watermark during the token sampling, instead of altering the logits, the watermark information is utilized to guide the sampling (Christ et al., 2024). Since the above two methods are not suitable for open-source LLMs, for open-source LLMs, it is impossible to manipulate the training and embed watermarks into the LLM parameters during training (Sun et al., 2022). However, the existing watermarking (Pan et al., 2025) techniques can verify the content and cannot defend against model stealing attacks where attackers use the model-generated content. And existing methods are dependent on random IDs, resulting in watermarked texts generated under different IDs are difficult to identify across IDs.Therefore, we propose a protection method that can prevent the use of rationales generated by LLMs for training SLMs.

Unlearnable Data. Unlearnable Data (ULD) refers to a category of data that is deliberately modified through subtle perturbations. This modification prevents the model from effectively learning useful representations during the training process while maintaining the perceived quality for human observers. EM (Huang et al.) first introduced the concept of Unlearnable data by optimizing sampling and class noise through leveraging the Bilevel error minimization objective. Considering the inefficiency of bilevel optimization, recent works have simplified it through adversarial examples (Fowl et al., 2021), and empirical rules of synthetic noise (Yu et al., 2022; Wu et al., 2023). Previous work (Li and Liu, 2023) utilized the bilevel optimization proposed in previous studies, which transformed the image classification problem into a natural language understanding task through word substitution, aiming to reduce the test performance. With the rise of LLMs, text-based ULD has gradually attracted attention, especially. Zhou et al. (Zhou et al., 2024) introduced additional parameters as safety vectors, which made harmful and hallucinatory behaviors unlearnable during model training without affecting the learning ability for new tasks. Influenced by (Boucher et al., 2022), (Liu et al.) embedded invisible text in the copyright web page and proposed a strategy based on GCG to enhance the perturbation effect. However, existing unlearnable data approaches typically require manipulating training data or labels. For large model providers, user inputs are uncontrollable, and model outputs must remain accurate. As a result, these methods are not well-suited for this setting.

3 Approach

In this section, we first define the reasoning transfer problem, then introduce our method Unnoticeable Reasoning Editing (UREdit), and finally present our selection strategy Sample- and Token-Level Selection to improve UREdit.

3.1 Problem Definition

We formalize the reasoning transfer problem as follows. Consider a large language model f_t that maps an input query x to both a reasoning path r and a final output y, denoted as:

$$f_t(x) \rightarrow (r, y) \tag{1}$$

Here, r represents the intermediate reasoning path, and y is the final answer. In our setting, unauthorized parties can "steal" such (x, r, y) samples to train a smaller model f_s, aiming to transfer the reasoning capabilities to the smaller model by minimizing the following loss:

$$\min_{\theta} \mathcal{L}(f_s(x;\theta), (r, y)) \tag{2}$$

The goal of the owners of large model is to limit this transferability by modifying only the reasoning path r, while ensuring that the altered reasoning r' appears visually consistent with the original reasoning path r. This can be formulated as:

$$\begin{aligned} &Acc(f_s(x;\theta'^*), y) < Acc(f_s(x;\theta^*), y) \\ &\theta^* = \arg\min_{\theta} \mathcal{L}(f_s(x;\theta), (r, y)), \\ &\theta'^* = \arg\min_{\theta'} \mathcal{L}(f_s(x;\theta'), (r', y)) \end{aligned} \tag{3}$$

Where $r \sim r'$ denotes that r and r' are indistinguishable to users.

Owners of large model: The owners of large model have full control over f_t and provide services to ordinary users and unauthorized parties through websites or API. However, the model owners are unable to distinguish between ordinary users and unauthorized third parties. As a result, for the query x, they can only edit the corresponding reasoning path r into r' , ensuring visual similarity between the r and r' while maintaining label y accuracy.

Unauthorized third parties: Unauthorized parties access the API provided by large model owners to obtain formatted reasoning outputs (r, y) for complex problem x, which they then use to construct reasoning datasets. When building these datasets, unauthorized parties only check the correctness of the model outputs and do not make further modifications.

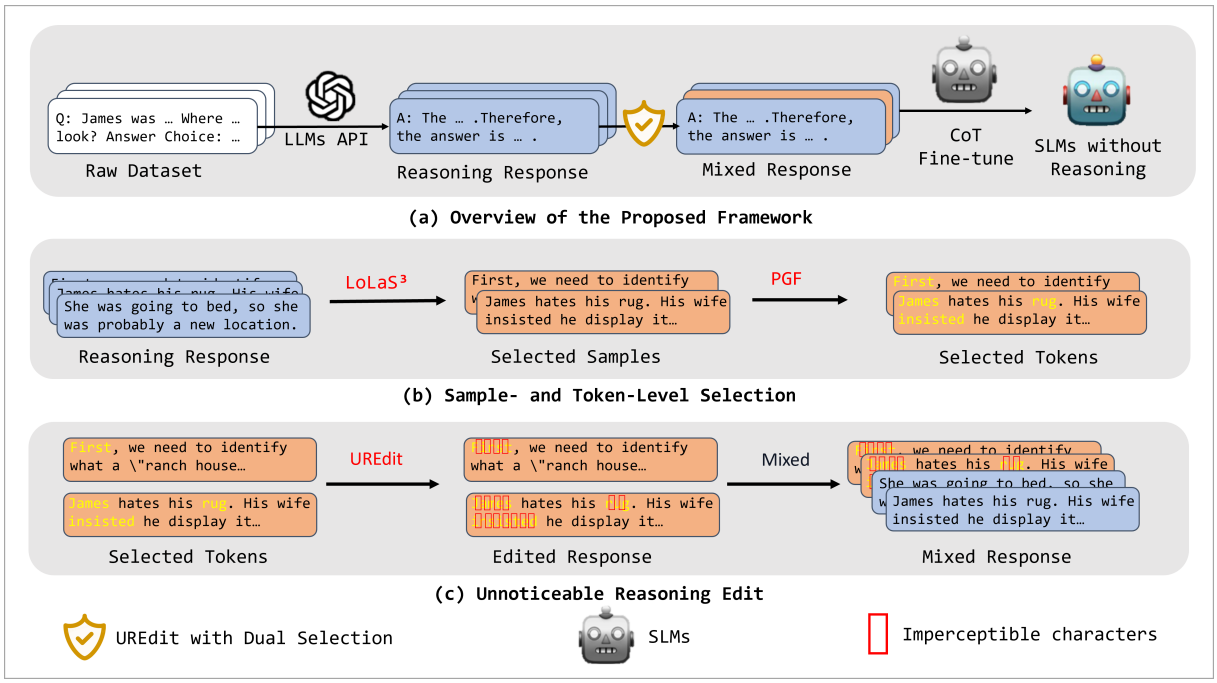


Figure 1: **Overview.** (a) Full workflow: reasoning responses are obtained via LLM APIs, selected and edited to prevent reasoning transfer to SLMs. (b) Sample- and Token-level selection: informative reasoning samples are selected using LoLaS³, followed by words-level filtering with PGF. (c) Unnoticeable reasoning editing: selected tokens are subtly modified using UREdit and mixed with unedited responses to maintain user experience.

3.2 Unnoticeable Reasoning Editing

Let $r = \{t_1, t_2, \ldots, t_n\}$ represent the original reasoning path, where each t_i denotes a token in the sequence. We aim to learn a perturbation function $\delta(\cdot)$ such that the altered reasoning path $r' = \delta(r)$ satisfies the following objective:

$$\max_{\delta} D(r, r') \quad subject\,to \quad r \sim r', \\ D(r, r') = ||E(t_1, t_2, \cdots, t_n) - E(t'_1, t'_2, \cdots, t'_n)|| \tag{4}$$

$D(r, r')$ measures the distance between token representations before and after perturbation, and the constraint $r \sim r'$ ensures that the original and perturbed responses remain visually indistinguishable. Hence, we propose Unnoticeable Reasoning Editing (UREdit), which introduces imperceptible characters into the original response r. In the following sections, we elaborate on the design of our method from three key aspects: (1) what type of characters to insert, (2) where to apply the insertion, and (3) how the editing process is implemented.

Character Type Selection: First, we determine the type of characters to be used for editing. The selected characters must meet the following criteria: 1. The character should be invisible to users and not occupy any space during model generation. 2. The character should significantly affect how small models tokenize the reasoning path. Based on these requirements, we choose a set of Unicode-encoded invisible characters that are font-independent and can disrupt tokenization patterns in commonly used tokenizers. Examples include \u200B, \u200C, \u200D, \u2060, and \u2063. These characters is invisible in text but can affect small models. For example, inserting \u200B into the word "language" as "lang\u200Buage" completely transforms the original token sequence of the GPT-2 tokenizer from [16129] to [17204, 9525, 84, 496].

Insertion Position Identification: Different positions have varying degrees of impact on the tokenizer process of small models, which in turn affects the level of protection for the reasoning path. However, we do not have access to the tokenizer of small models. Unlike previous optimization-based methods that rely on knowledge of the tokenization behavior, we cannot determine which insertion positions would most disrupt the token sequence. Therefore, by default, to maximize the protection of reasoning outputs, we embed imperceptible characters after every visible token in every sample.

Overall Editing Procedure: Finally, we integrate the selected imperceptible characters with the insertion position to edit the reasoning outputs of large language models. Specifically, after obtaining a reasoning path from the model, we randomly choose one or more invisible Unicode characters and insert them after every visible character in each word. This approach en-

sures broad disruption to tokenization patterns without affecting human readability. For example, the word "Hello" can be edited to "H \u200Be\u200Cl\u200Cl\u200Do\u2063."

3.3 Sample- and Token-Level Selection

In the streaming query setting, we continuously generate reasoning paths $\mathcal{D} = \{r_1, r_2, \cdots, r_N\}$, where each path is composed of a sequence of tokens $r_i = \{t_{i1}, t_{i2}, \cdots, t_{in}\}$. By default, the editing targets of UREdit are defined as:

$$\mathcal{S}_{\text{default}} = \{(r, t) \mid r \in \mathcal{D}, t \in r\} \tag{5}$$

This default method applies uniform perturbation to all reasoning paths and tokens without considering their informativeness or confidence level, resulting in excessive and unnecessary modifications that may be detectable.

To address this issue, we propose Sample- and Token-Level Selection to identify informative reasoning paths and tokens for editing. Specifically, the final set of editing targets is determined by combining both criteria:

$$\mathcal{S}_{\text{edit}} = \{(r, t) \mid r \in R_{\text{edit}}, t \in T_{\text{edit}}(r)\} \tag{6}$$

where $R_{\text{edit}} \subseteq \mathcal{D}$ denotes the set of selected reasoning paths for editing, and $T_{\text{edit}}(r) \subseteq r$ represents the subset of tokens selected from path r.

Next, we introduce our selection strategies, including Probability-Guided Filtering(PGF) , Log-Likelihood-Based Sample Selection for Streaming Queries(LoLaS³), and Dual-Level Selection.

Probability-Guided Filtering: In a reasoning path r, each token t_i is associated with a generation probability p_i, which is derived from the model's output distribution. This probability quantifies the model's confidence in generating the corresponding word at that position. A lower p_i corresponds to higher uncertainty in the generation of t_i. Intuitively, during the transfer of reasoning capabilities, tokens with greater uncertainty tend to encode more pivotal reasoning signals.

Therefore, we propose Probability-Guided Filtering, which selects the least probable tokens for editing while keeping all samples unmodified by default. In other words, we select *all* samples, but only modify *specific* tokens within each sample, based on their generation probabilities. Formally, the set of selected tokens is defined as:

$$T_{\text{edit}} = \{t_i \in r \mid p_i \leq \text{Threshold}(\{p_j \mid t_j \in r\}, \alpha)\} \tag{7}$$

where $\alpha \in (0, 1]$ is a hyperparameter indicating the proportion of tokens to be selected based on their generation probabilities. The function $\text{Threshold}(\cdot, \alpha)$ returns the $\lfloor \alpha \times n \rfloor$-th smallest probability in the sequence, with $n = |r|$ being the length of the reasoning path.

Log-Likelihood-Based Sample Selection for Streaming Queries: Log-likelihood is a commonly used metric for evaluating the quality of reasoning output samples generated by large language models. For a given reasoning path $r = \{t_1, t_2, \ldots, t_n\}$, its log-likelihood is defined as:

$$logP(r) = \sum_{i=1}^{n} log(p_i) \tag{8}$$

where p_i denotes the generation probability of the i-th token in the sequence. A higher log-likelihood indicates greater internal consistency and fluency of the sample, suggesting that it is more likely to be semantically coherent and reliable. Consequently, such high-quality samples are also more informative for reasoning ability transfer. Based on this, we select the samples with the higher log-likelihood for further editing. By default, all tokens in the selected samples are selected.

However, in practical scenarios where large models process queries as a stream and return results in real time, it is infeasible to obtain the full dataset in advance. As a result, we cannot select the globally highest-quality samples. To address this challenge in streaming queries, we design an algorithm that enables online processing and dynamically selects the high-quality reasoning paths r. The detailed implementation of the proposed algorithm is presented in Algorithm 1.

In the algorithm, we set a window size W and a sampling ratio α. When the window size is n, we only focus on the current sample and its previous n historical samples. The sampling ratio $\alpha \in (0, 1]$ determines the proportion of top-ranked samples to be selected from the window for potential editing.

Dual-Level Selection: To further enhance the imperceptibility of the edits, we introduce a dual-level selection. Unlike methods that operate at a single level, our dual-level selection mechanism applies both strategies in a coordinated manner: LoLaS³ is first used to select high-quality reasoning paths, and PGF is then applied to refine the editing targets within each selected path. The final editing set $\mathcal{S}_{\text{edit}}$ consists of all token-sample pairs (r, t) that pass both filters.

Algorithm 1 Log-Likelihood-Based Sample Selection for Streaming Queries

Require: A data stream of (sample, log_likelihood) pairs, window size W, sampling ratio α
Ensure: A set R_{edit} containing samples selected for editing
Initialize an empty queue $\mathcal{Q}$ to store recent (sample, log_likelihood) pairs
Initialize an empty list $\mathcal{R}_{edit}$ to to collect high-quality samples
for each incoming pair (s, ℓ) **do**
    Append (s, ℓ) to the end of $\mathcal{Q}$
    if the length of $\mathcal{Q}$ exceeds W **then**
        Remove the oldest element from the front of $\mathcal{Q}$
    end if
    Let $n \leftarrow |\mathcal{Q}|$
    Compute $K \leftarrow \max(1, \lfloor \alpha \cdot n \rfloor)$
    Extract a list $L \leftarrow [\ell' \mid (\cdot, \ell') \in \mathcal{Q}]$ of all log-likelihood values in $\mathcal{Q}$
    Sort L in descending order
    Set threshold $\tau \leftarrow L[K-1]$ (the K-th largest log-likelihood value)
    if $\ell > \tau$ **then**
        Add s into $\mathcal{R}_{\text{edit}}$
    end if
end for
return $\mathcal{R}_{\text{edit}}$

4 Experiments

Tasks and datasets. We evaluate UREdit under the default setting on five datasets spanning three types of reasoning tasks: arithmetic reasoning, commonsense reasoning and symbolic reasoning (Kojima et al., 2022).Additionally, we use Qwen2.5-14B-Instruct to generate reasoning outputs with logits. More details are provided in Appendix A.1.

Models. We use the generated data (Ho et al., 2023) as the output results of the large model. For the small models, we consider five general-purpose models: GPT-2 (Radford et al., 2019), T5 (Raffel et al., 2020), FlanT5 (Chung et al., 2024), Qwen (Yang et al., 2024) and Llama.More details are provided in Appendix A.2

Baselines and setup. We use the CoT Fine-tune provided in (Ho et al., 2023) for all experiments, which involves fine-tuning small models using either the original or edited reasoning paths of large models. In all experiments, we fine-tune the entire model with a fixed learning rate of 3e-4 and a batch size of 2. We train all models for up to 20 epochs.

4.1 Results

In this section, we evaluate UREdit under the default setting and compare it with the baseline across different datasets and models. We evaluate UREdit when LoLaS³ and PGF are applied separately to construct S_{edit}. Furthermore, we validate the effectiveness of the dual level selection.

Main results of UREdit. As presented in Table 1, across different datasets and model architectures, the accuracy achieved by UREdit under the default setting is consistently lower than that of the Baseline. This result indicates that UREdit effectively disrupts the learning process of small models on reasoning paths generated by large language models, thereby demonstrating its efficacy in protecting the reasoning content of LLM outputs.

The effectiveness of UREdit is closely related to the characteristics of the dataset. For instance, on the GSM8K dataset, small models inherently struggle to learn mathematical reasoning from the reasoning paths generated by large models. As a result, UREdit has a relatively limited impact in this setting, indicating weaker disruption effects compared to other datasets.

Effectiveness of UREdit sometimes also varies with model architectures. On the CommonsenseQA dataset, UREdit demonstrates a strong impact on Encoder-Decoder models, while demonstrating relatively limited influence on Decoder-only models. This difference can be attributed to the fact that Encoder-Decoder architectures rely more heavily on the full semantic understanding of the input and the structured reasoning path during training, whereas Decoder-only models primarily generate output based on preceding context and are therefore less sensitive to editing perturbations (Diao et al., 2024; Luo et al., 2025).

Effect of Applying LoLaS³ and PGF in UREdit. We use Qwen2.5-14B-Instruct to generate 2000 reasoning outputs for the CommonsenseQA dataset and fine-tune T5-large via CoT-Fine-tune. During the fine-tuning process, we apply either LoLaS³ or PGF to get $\mathcal{S}_{edit}$. In the evaluation of LoLaS³, we simulate the streaming output of large models and set the window size to 400. Note that LoLaS³ and PGF are applied independently in the experiments. Meanwhile, we use random selection(RS) as the baseline.

The results are presented in Table 2. LoLaS³ and PGF outperform RS at various selection ratios. As the selection ratio increases, the effectiveness of our selection strategies first rises and then declines. This is because when fewer samples are selected, the protection provided by UREdit is weaker; however, as more samples are edited, the protection effect improves. However, when the selection ratio becomes too high, the overlap between samples selected by LoLaS³ and RS increases, reducing the distinctiveness of LoLaS³.

Table 1: Comparison of accuracy between the base method and UREdit across multiple datasets and SLMs. The consistently lower accuracy under UREdit suggests an effective disruption of reasoning outputs transfer, demonstrating its capability in protecting LLM-generated reasoning paths.

Model	Params	Method	Commom SenseQA	Strategy QA	GSM8K	Coin Flip
GPT2	124M	Baseline	21.47 ±0.40	52.70 ±1.47	2.12 ±0.35	61.10 ±3.00
		UREdit	**15.30** ±0.42	**10.83** ±2.90	**1.87** ±0.29	**58.27** ±1.00
	355M	Baseline	20.40 ±0.51	53.87 ±1.36	**1.72** ±0.88	68.87 ±0.75
		UREdit	**18.57** ±0.90	**15.10** ±2.62	1.74 ±0.16	**61.30** ±2.01
	774M	Baseline	21.35 ±1.25	51.10 ±2.60	2.62 ±0.16	66.77 ±2.10
		UREdit	**18.53** ±1.27	**13.67** ±2.82	**1.59** ±0.35	**64.23** ±1.15
Flan-T5	60M	Baseline	32.97 ±0.83	51.73 ±1.35	1.72 ±0.49	97.57 ±1.40
		UREdit	**15.07** ±1.56	**8.00** ±3.89	**1.49** ±0.57	**83.33** ±3.69
	220M	Baseline	39.57 ±2.40	53.50 ±0.95	2.48 ±0.16	94.23 ±2.04
		UREdit	**16.70** ±2.52	**13.17** ±1.37	**1.19** ±0.51	**91.30** ±2.83
	700M	Baseline	43.40 ±1.08	54.10 ±2.14	2.81 ±0.27	95.77 ±0.40
		UREdit	**26.33** ±2.22	**23.53** ±4.55	**2.38** ±0.42	**81.53** ±0.40
T5	60M	Baseline	9.30 ±0.75	15.03 ±1.42	2.17 ±0.11	95.33 ±2.41
		UREdit	**6.61** ±0.74	**6.06** ±3.86	**0.71** ±0.32	**81.10** ±2.54
	220M	Baseline	36.13 ±1.18	54.40 ±1.14	2.40 ±0.27	95.77 ±1.15
		UREdit	**20.90** ±1.37	**34.17** ±1.81	**1.47** ±0.44	**93.37** ±1.15
	700M	Baseline	42.73 ±3.96	55.97 ±2.06	2.88 ±0.55	94.47 ±1.66
		UREdit	**24.87** ±2.58	**24.20** ±0.56	**2.10** ±0.42	**83.33** ±1.15
Qwen2.5	0.5B	Baseline	19.33 ±1.65	50.80 ±1.31	1.93 ±0.27	68.87 ±0.75
		UREdit	**17.97** ±1.90	**47.40** ±1.78	**1.69** ±0.38	**59.77** ±0.81
Llama-3.2	1B	Baseline	19.40 ±1.39	52.13 ±1.07	1.74 ±0.53	70.43 ±1.03
		UREdit	**17.93** ±1.42	**26.33** ±1.69	**1.39** ±0.45	**44.67** ±1.15

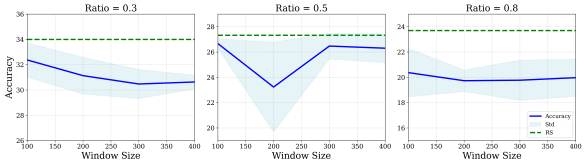


Figure 2: Experiments on the impact of sliding window size under streaming queries at different sampling ratios. The results demonstrate that LoLaS³ consistently outperforms RS regardless of window size, indicating its robustness to this parameter.

Evaluation of the Dual Level Selection. To validate the effectiveness of the dual level selection, we compare LoLaS³ with RS at the sample level, and PGF with RS at the token level. We construct different $\mathcal{S}_{\text{edit}}$ under all possible combinations and evaluate their effects using UREdit. The same random seed is used during random selection and the selection ratio of 0.5 is used for both levels.

The results are presented in Table 3. When samples are selected randomly, PGF performs better than random token selection. Likewise, when tokens are selected at random, LoLaS³ outperforms random sample selection. When both LoLaS³ and PGF are used together, their performance exceeds that of the other three combinations. This result demonstrates that LoLaS³ and PGF are orthogonal strategies and can be effectively combined.

Ablation Study When simulating the streaming queries of LLMs, the size of the sliding window has a certain impact on sample selection. To investigate the influence of the sliding window on sample selection, we performed LoLaS³ using different window sizes across various selection ratios.

We evaluate the robustness of LoLaS³ under different window sizes (100, 200, 300, and 400), at selection ratios of 0.3, 0.5, and 0.8. As presented in the figure 2, although the window size has some impact on the performance of our strategy, LoLaS³ consistently outperforms RS across all settings.

4.2 Evaluation

In this section, we use GPT as a judge to conduct evaluation of the reasoning path generated by SLMs. Specifically, We randomly select 15 questions from various datasets and use different model to generate reasoning responses. The responses were evaluated based on four criteria: logical soundness(L), completeness(C), relevance(R), and factual accuracy(F). The scoring scale ranged from 1 to 5, with 5 indicating the highest quality.

Table 2: Comparison of selection strategy at sample- and token-level across various ratios. Results demonstrate the proposed method consistently outperforms random selection under all settings.

Ratio	Level	Method	Accuracy(↓)	Level	Methods	Accuracy(↓)
0	Sample	-	40.46 ±1.40	Word	-	40.46 ±1.40
0.3		RS	34.00 ±0.56		RS	36.03 ±1.21
		LoLaS³	29.26 ±1.00		PGF	31.10 ±0.46
0.5		RS	27.33 ±1.30		RS	30.27 ±1.84
		LoLaS³	23.23 ±1.89		PGF	25.73 ±1.10
0.8		RS	23.70 ±0.40		RS	24.40 ±1.23
		LoLaS³	21.23 ±0.76		PGF	23.30 ±0.90
1		-	19.36 ±0.49		-	19.36 ±0.49

Table 3: Evaluation of four selection strategies. At both the sample and token levels, LoLaS³ and PGF consistently perform better than RS. Notably, their combination achieves the highest performance.

Sample Level	Word Level	Accuracy(↓)
–	–	40.46 ±1.40
RS	RS	35.97 ±0.32
RS	PGF	35.46 ±0.31
LoLaS³	RS	33.20 ±0.91
LoLaS³	PGF	**31.63** ±1.15

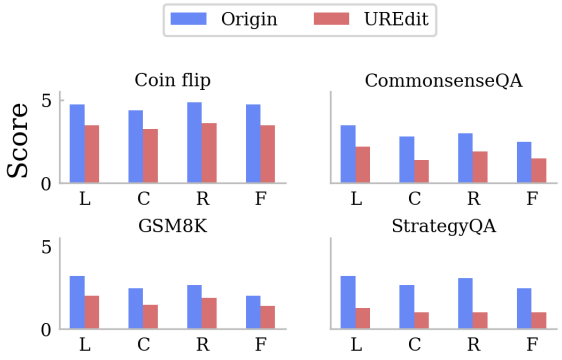


Figure 3: Comparison of T5-base trained on original vs edited data, evaluated using GPT-4o. Models trained on edited data demonstrate significantly lower reasoning quality across multiple datasets.

As presented in the figure 3, the reasoning path generated by the smaller model trained on edited data scored lower across different datasets and evaluation dimensions compared to the model trained on original data. The results demonstrate that our editing method successfully prevents the transfer of reasoning ability from larger models.For more details, please refer to the Appendix A.3.

4.3 Analysis

We construct data trees retain the main branches on various datasets. Finally, we calculate the average width and average depth of the pruned data trees.

The results, as presented in Table 4, indicate that across different datasets, the average width and depth of the trees corresponding to the edited datasets is larger than that of the original datasets. This suggests that the edited data trees contain a greater number of reasoning paths, and these paths are longer on average. In this scenario, when SLMs attempt to inference, they are more likely to "get lost among the reasoning paths," making it difficult for them to arrive at correct conclusions. For more details, please see the Appendix A.4.

Table 4: Comparison of MCTS tree depth and width between original and edited data. Edited data leads to deeper and wider search trees, indicating more complex reasoning paths.

Metric	Type	Common senseQA	Strategy QA	GSM8K	Coin Flip
Depth	Origin	3.07	4.00	3.00	3.67
	UREdit	8.01	10.25	8.75	5.33
Width	Origin	3.25	1.17	1.83	1.29
	UREdit	5.84	2.57	4.35	3.45

5 Conclusion

We propose UREdit to prevent unauthorized parties from exploiting the reasoning capabilities of LLMs by transferring them into SLMs. We introduce invisible characters into reasoning paths, effectively disrupting the transfer. Furthermore, to enhance the stealthiness of our approach under streaming query settings, we introduce LoLaS³ and PGF, which achieving strong protection with minimal disturbance. Extensive experiments across multiple datasets and model demonstrate the effectiveness of UREdit. Moreover, we observe that the Monte Carlo search trees constructed from edited data exhibit greater depth and width, which further hinders the ability of SLMs to learn useful reasoning path. Overall, UREdit effectively protects the intellectual property of LLM reasoning outputs while preserving response fluency and coherence.

Limitations

Our method has several limitations that are worth noting:

First, while embedding invisible characters into reasoning outputs effectively disrupts knowledge transfer to small models, the perturbations may be neutralized if downstream models perform strict preprocessing, such as encoding normalization or character filtering before training.

Second, our approach can occasionally affect the visual consistency of model outputs. Due to varying interpretations of invisible characters across different small models, some generated responses may contain unexpected artifacts or visible anomalies, increasing the risk of reverse detection by adversaries.

Acknowledgments

References

Lewis Birch, William Hackett, Stefan Trawicki, Neeraj Suri, and Peter Garraghan. 2023. Model leeching: An extraction attack targeting llms. *arXiv preprint arXiv:2309.10544*.

Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. 2022. Bad characters: Imperceptible nlp attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1987–2004. IEEE.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Miranda Christ, Sam Gunn, and Or Zamir. 2024. Undetectable watermarks for language models. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 1125–1139. PMLR.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, and 1 others. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Haiwen Diao, Yufeng Cui, Xiaotong Li, Yueze Wang, Huchuan Lu, and Xinlong Wang. 2024. Unveiling encoder-free vision-language models. *arXiv preprint arXiv:2406.11832*.

Liam Fowl, Micah Goldblum, Ping yeh Chiang, Jonas Geiping, Wojtek Czaja, and Tom Goldstein. 2021. Adversarial examples make strong poisons. *Preprint*, arXiv:2106.10807.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Kashmir Hill. 2022. The secretive company that might end privacy as we know it. In *Ethics of Data and Analytics*, pages 170–177. Auerbach Publications.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. *http://arxiv.org/abs/2212.10071*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, and 1 others. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Hanxun Huang, Xingjun Ma, Sarah Monazam Erfani, James Bailey, and Yisen Wang. Unlearnable examples: Making personal data unexploitable. *Preprint*, arXiv:2101.04898.

Minki Kang, Seanie Lee, Jinheon Baek, Kenji Kawaguchi, and Sung Ju Hwang. 2023. Knowledge-augmented reasoning distillation for small language models in knowledge-intensive tasks. In *Advances in Neural Information Processing Systems*, volume 36, pages 48573–48602. Curran Associates, Inc.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Shiyang Li, Jianshu Chen, Yelong Shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing Qian, Baolin Peng, Yi Mao, Wenhu Chen, and Xifeng Yan. 2022. Explanations from large language models make small reasoners better. *https://arxiv.org/abs/2210.06726*.

Xinzhe Li and Ming Liu. 2023. Make text unlearnable: Exploiting effective patterns to protect personal data. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pages 249–259, Toronto, Canada. Association for Computational Linguistics.

Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip Yu. 2024. A survey of text watermarking in the era of large language models. *ACM Computing Surveys*, 57(2):1–36.

Ruixuan Liu, Toan Tran, Tianhao Wang, Hongsheng Hu, Shuo Wang, and Li Xiong. Expshield: Safeguarding web text from unauthorized crawling and language modeling exploitation. *Preprint*, arXiv:2412.21123.

Yingfeng Luo, Tong Zheng, Yongyu Mu, Bei Li, Qinghong Zhang, Yongqi Gao, Ziqiang Xu, Peinan Feng, Xiaoqian Liu, Tong Xiao, and 1 others. 2025. Beyond decoder-only: Large language models can be good encoders for machine translation. *arXiv preprint arXiv:2503.06594*.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. Teaching small language models to reason. *https://arxiv.org/abs/2212.08410*.

Thanh Tam Nguyen, Thanh Trung Huynh, Zhao Ren, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. 2022. A survey of machine unlearning. *arXiv preprint arXiv:2209.02299*.

Kun-Peng Ning, Jia-Yu Yao, Yu-Yang Liu, Mu-Nan Ning, and Li Yuan. 2025. Gpt as a monte carlo language tree: A probabilistic perspective. *arXiv preprint arXiv:2501.07641*.

Leyi Pan, Aiwei Liu, Shiyu Huang, Yijian Lu, Xuming Hu, Lijie Wen, Irwin King, and Philip S Yu. 2025. Can llm watermarks robustly prevent unauthorized knowledge distillation? *arXiv preprint arXiv:2502.11598*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, and 1 others. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7059–7073. Association for Computational Linguistics.

Zhensu Sun, Xiaoning Du, Fu Song, Mingze Ni, and Li Li. 2022. Coprotector: Protect open-source code against unauthorized training usage with data poisoning. In *Proceedings of the ACM Web Conference 2022*, pages 652–660.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Shutong Wu, Sizhe Chen, Cihang Xie, and Xiaolin Huang. 2023. One-pixel shortcut: on the learning preference of deep neural networks. *Preprint*, arXiv:2205.12141.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. 2022. Availability attacks create shortcuts. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 2367–2376. ACM.

Xin Zhou, Yi Lu, Ruotian Ma, Yujian Wei, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Making harmful behaviors unlearnable for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10258–10273, Bangkok, Thailand. Association for Computational Linguistics.

A Appendix

A.1 Datasets

Openly Available Datasets For arithmetic reasoning, we consider the GSM8K (Cobbe et al., 2021) dataset, which is part of a recent benchmark and represents a more challenging dataset that requires multi-step reasoning to solve. For commonsense reasoning, we use CommonsenseQA (Talmor et al., 2018) and StrategyQA (Geva et al., 2021). CommonsenseQA presents questions with complex semantics that typically require reasoning based on prior knowledge. In contrast, StrategyQA requires the model to perform implicit multi-hop reasoning to answer questions.For symbolic reasoning, we use Coin Flip (Wei et al., 2022). Coin Flip requires the model to determine whether a coin remains heads-up after people either flip or do not flip the coin. Table 5 provides a more detailed description of the datasets used in our study. The open-source datasets used in our experiments are

Table 5: Detailed Description of Public Datasets Utilized in This Study

Dataset	Category	Training Samples	Test Samples	Data Split	License
CommonSenseQA	commonsense	9741	1221	Original	Unspecified
StrategyQA	commonsense	1603	687	70:30	Unspecified
Coin Flip	symbolic	350	150	70:30	Unspecified
GSM8K	arithmetic	7473	1319	Original	MIT

obtained from Ho et al. (Ho et al., 2023)'s work. https://github.com/itsnamgyu/reasoning-teacher

Generated Dataset To validate the effectiveness of the dual selection strategy, we select 2000 questions from the CommonsenseQA training data and use Qwen2.5-14B-Instruct to generate reasoning outputs. During the generation process, we use greedy decoding to obtain deterministic results. For the maximum sequence length in model generation, we set it to 256. After obtaining the reasoning outputs, we slightly standardize their format to make them suitable for CoT fine-tuning. Beyond this, we do not perform any additional operations. Table 6 presents specific instances of the generated data.

Prompt

You are given a question with multiple answer choices. Your task is to provide a concise explanation for the selected answer in the following format: The output should be in the form: <reason> --> <choice>, where <reason> is the full reasoning and <choice> only is the selected answer.{question}. Let's think step by step.

Figure 4: Template for Qwen2.5-14B-Instruct. question will be replaced with the specific reasoning question during inference.

A.2 Models

In Table 7, we provide the detailed versions of the models. The table summarizes the small language models (SLMs) used in our experiments, along with their architectures, parameter counts, and corresponding HuggingFace checkpoints. We include both encoder-decoder and decoder-only models across a range of sizes (from 60M to 700M) to ensure a comprehensive evaluation of UREdit under diverse model settings. In addition, we also evaluate on recent open-weight models such as Qwen2.5 and Llama3.2 to validate generalization beyond standard benchmarks.

A.3 Evaluation

To provide a more comprehensive evaluation of our method's effectiveness, we adopt the LLM-as-a-judge approach and use GPT-4 as an automated evaluator to assess reasoning quality. Specifically, we compare the reasoning outputs generated by models trained on original data and those trained on edited data in response to the same questions. The prompts used for GPT-based evaluation follow the instruction template shown in Table 5, ensuring consistency and objectivity in the assessment process.

A.4 Data Tree

As presented in (Ning et al., 2025), we argue that Fine-tune-CoT cannot truly transfer reasoning capabilities to SLMs. The apparent reasoning abilities exhibited by SLMs are more likely based on probabilistic pattern-matching rather than formal reasoning.

Since different GPT models trained on the same dataset exhibit structural similarities in the visualization of model trees, we represent model trees using data trees. We independently construct Monte Carlo language trees for the original and edited reasoning output datasets. Additionally, we analyze the reasoning capability of SLMs from the perspective of data trees. For a comprehensive description of the data tree construction process, please refer to the work (Ning et al., 2025).

As illustrated in the figure 6, the Monte Carlo search trees constructed from models trained on edited data show a larger number of alternative reasoning paths, particularly at the second layer. As the tree depth increases, this divergence grows, leading to substantially wider and deeper structures than those derived from original data . Such an increase in path diversity introduces greater ambiguity during reasoning path learning, making it harder for small models to extract consistent logic — effectively causing them to be "lost in the reasoning paths" and reducing their overall reasoning performance

Prompt

```
You are a professional reasoning evaluator. Below are two
reasoning responses to the same question.

Evaluate each response on the following four criteria,
scoring each from 1 to 5 (5 being the best):
- Logical soundness
- Completeness
- Relevance
- Factual accuracy

After scoring, determine which response is overall better and
explain why.

[Question]: {question}
[Response A]: {response_a}
[Response B]: {response_b}

Output your evaluation in the following format:

Response A:
- Logical soundness:
- Completeness:
- Relevance:
- Factual accuracy:

Response B:
- Logical soundness:
- Completeness:
- Relevance:
- Factual accuracy:

[Better Response]: A or B
[Reason]: (Explain why this response is better overall)
```

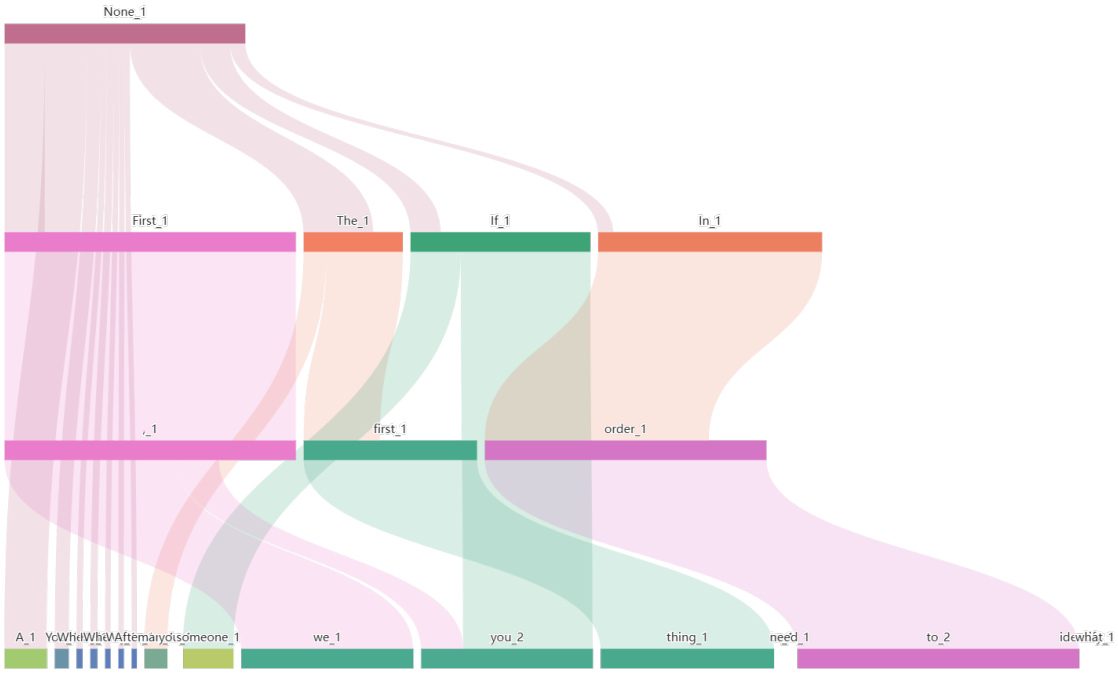
Figure 5: Prompt design for comparing two reasoning paths using GPT as an evaluator.

Table 6: Prediction examples generated by Qwen2.5-14B-Instruct for CommonsenseQA

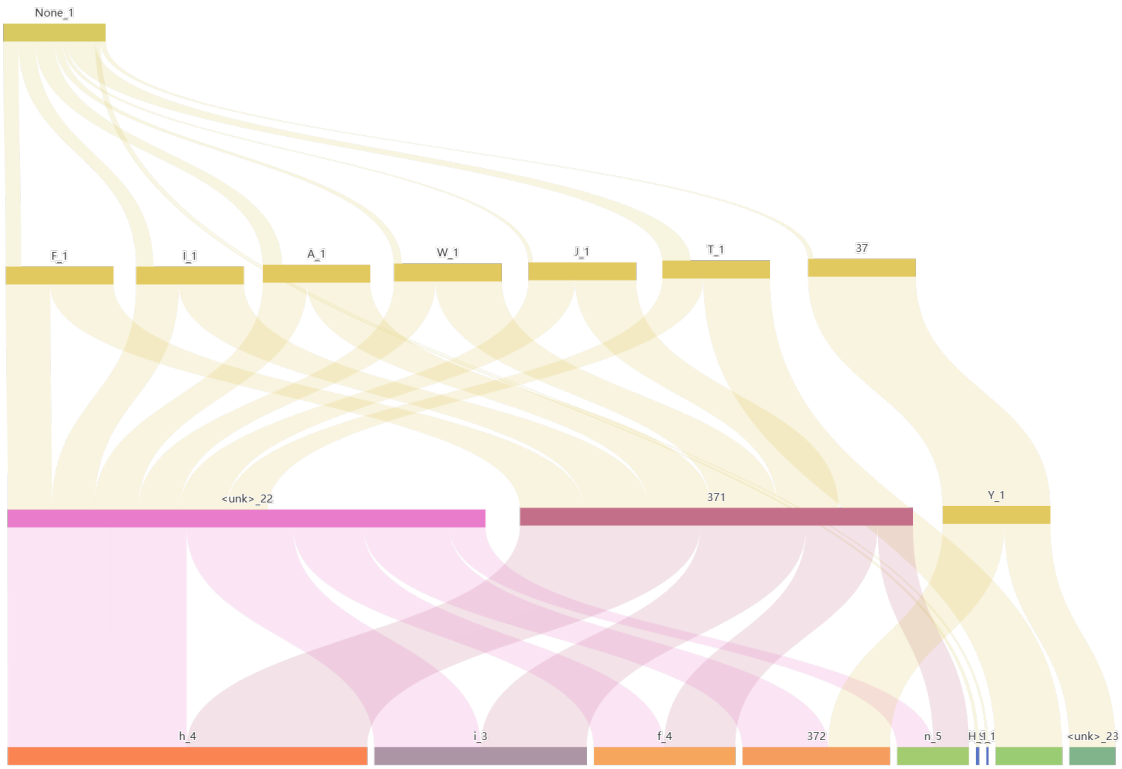
Question	Answer
Of all the rooms in a house it was his favorite, the aromas always drew him to the what? Answer choices: (A) yard, (B) basement, (C) kitchen, (D) living room, (E) garden.	The sentence mentions äromaswhich typically relate to cooking or food preparation. Among the options provided, the kitchen is the place most commonly associated with preparing meals and thus emitting pleasant smells. Therefore, the correct choice is:–> (C) kitchen
Google Maps and other highway and street GPS services have replaced what? Answer choices: (A) united states, (B) mexico, (C) countryside, (D) atlas, (E) oceans.	Google Maps and similar GPS services provide real-time directions and information about streets and highways, which were traditionally found in physical maps or atlases. Therefore, these digital services have largely replaced the need for paper maps that contain detailed road information.–> (D) atlas
Where do you put your grapes just before checking out? Answer choices: (A) mouth, (B) grocery cart, (C) super market, (D) fruit basket, (E) fruit market.	When considering the context of shopping for groceries and the process of purchasing items, the most logical place to put grapes just before checking out would be in a grocery cart. This is because a grocery cart is used to carry items from around the store to the checkout area. Therefore, the correct choice is:–> (B) grocery cart

Table 7: SLMs involved in the experiments and the corresponding checkpoints

SLM	Architecture	Params	Name	Checkpoints
GPT2	Decode-only	124M	gpt2	*openai-community/gpt2*
		355M	gpt2-medium	*openai-community/gpt2-medium*
		774M	gpt2-large	*openai-community/gpt2-large*
Flan-T5	Encode-Decode	60M	flan-t5-small	*google/flan-t5-small*
		220M	flan-t5-base	*google/flan-t5-base*
		700M	flan-t5-large	*google/flan-t5-large*
T5	Encode-Decode	60M	t5-small	*google-t5/t5-small*
		220M	t5-base	*google-t5/t5-base*
		700M	t5-large	*google-t5/t5-large*
Qwen2.5	Decode-only	0.5B	qwen2.5	*Qwen/Qwen2.5-0.5B*
Llama3.2	Decode-only	1B	llama3.2	*meta-llama/Llama-3.2-1B*



(a) Origin Data



(b) Edited Data

Figure 6: Visualization of reasoning path trees from models trained on original and edited data. Monte Carlo search trees are constructed under the same settings, with only the top three layers retained to highlight structural differences. Edited data leads to more scattered reasoning paths, indicating increased learning difficulty for small models.