Stepwise Perplexity-Guided Refinement for Efficient Chain-of-Thought Reasoning in Large Language Models

Anonymous ACL submission

Abstract

Chain-of-Thought (CoT) reasoning, which breaks down complex tasks into intermediate reasoning steps, has significantly enhanced the performance of large language models (LLMs) on challenging tasks. However, the detailed reasoning process in CoT often incurs long generation times and high computational costs, partly due to the inclusion of unnecessary steps. To address this, we propose a method to identify critical reasoning steps using perplexity as a measure of their importance: a step is deemed critical if its removal causes a significant increase in perplexity. Our method enables models to focus solely on generating these critical steps. This can be achieved through two approaches: refining demonstration examples in few-shot CoT or fine-tuning the model using selected examples that include only critical steps. Comprehensive experiments validate the effectiveness of our method, which achieves a better balance between the reasoning accuracy and efficiency of CoT.

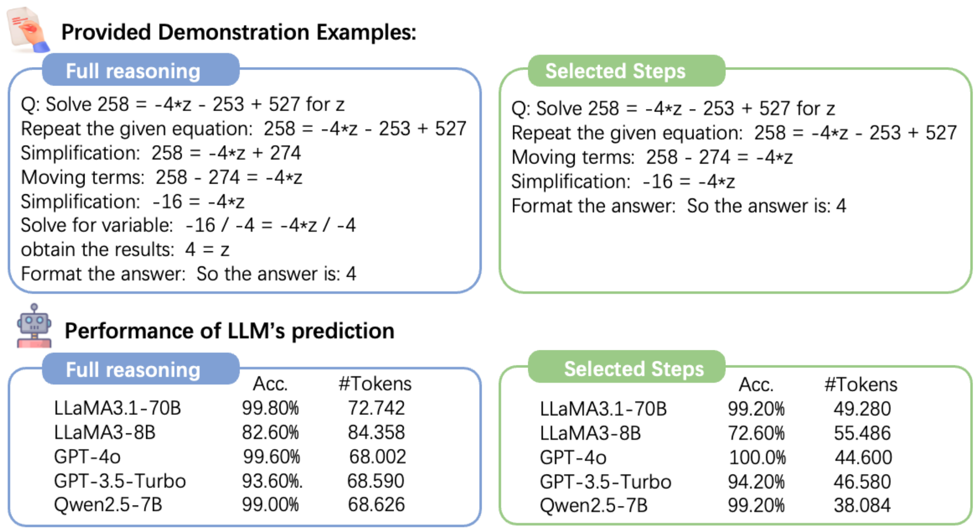
1 Introduction

Large language models (LLMs) are powerful generative models capable of performing diverse tasks in different domains (Gramopadhye et al., 2024; Karabacak and Margetis, 2023; Ling et al., 2024) and demonstrating strong reasoning capabilities (Jaech et al., 2024). Recent advancements, such as few-shot/zero-shot Chain-of-Thought (CoT) (Wei et al., 2022; Kojima et al., 2022), as well as fine-tuning (Liu et al., 2023), have significantly enhanced the LLMs' reasoning capabilities by leveraging intermediate reasoning steps. In particular, through few-shot CoT, LLMs can learn from the reasoning steps in the demonstration examples and apply similar reasoning patterns to target tasks. In the case of zero-shot CoT, LLMs are prompted to"think step by step" to generate reasoning steps. In fine-tuning, LLMs can also learn from the reasoning steps in the fine-tuning samples, further enhancing their reasoning abilities.

While many existing reasoning methods rely on available data (e.g., few-shot examples or fine-tuning datasets), there is limited understanding of which reasoning steps are truly essential and how their impact varies across different models. This gap hinders progress in two key areas: (1) how to effectively identify and remove unimportant reasoning steps from the data to reduce computational costs, and (2) whether the important reasoning steps for one model are also important to another.

For example, we observe that removing certain reasoning steps from the demonstrations in few-shot CoT can have varying effects: some models follow the modified examples and generate much fewer tokens while maintaining reasoning accuracy, whereas others experience a decline in performance. Specifically, we consider a math problem of function solving (Saxton et al., 2019). We compare two versions of demonstrations when conducting few-shot CoT: one with full manually crafted reasoning paths and another containing only intuitively important steps, as shown in Figure 1. For most models, removing certain steps significantly reduces the number of generated tokens with minimal impact on accuracy, suggesting that the removed steps contribute limited meaningful information. However, LLaMA3-8B shows a noticeable decline in accuracy, indicating that the importance of reasoning steps can vary across different LLMs.

Similar to the few-shot CoT scenario, when given a set of fine-tuning samples with reasoning steps, some LLMs may find some steps redundant, and the fine-tuning cannot improve the prediction accuracy. However, these LLMs will follow the fine-tuning samples to generate the additional tokens, raising the computation cost. Other LLMs may struggle to develop reasoning capabilities when given too few reasoning steps during fine-tuning. This observation will be further



Provided Demonstration Examples:

Full reasoning

Q: Solve 258 = -4*z - 253 + 527 for z
Repeat the given equation: 258 = -4*z - 253 + 527
Simplification: 258 = -4*z + 274
Moving terms: 258 - 274 = -4*z
Simplification: -16 = -4*z
Solve for variable: -16 / -4 = -4*z / -4
obtain the results: 4 = z
Format the answer: So the answer is: 4

Selected Steps

Q: Solve 258 = -4*z - 253 + 527 for z
Repeat the given equation: 258 = -4*z - 253 + 527
Moving terms: 258 - 274 = -4*z
Simplification: -16 = -4*z
Format the answer: So the answer is: 4

Performance of LLM's prediction

Full reasoning

	Acc.	#Tokens
LLaMA3.1-70B	99.80%	72.742
LLaMA3-8B	82.60%	84.358
GPT-4o	99.60%	68.002
GPT-3.5-Turbo	93.60%.	68.590
Qwen2.5-7B	99.00%	68.626

Selected Steps

	Acc.	#Tokens
LLaMA3.1-70B	99.20%	49.280
LLaMA3-8B	72.60%	55.486
GPT-4o	100.0%	44.600
GPT-3.5-Turbo	94.20%	46.580
Qwen2.5-7B	99.20%	38.084

Figure 1: Prediction accuracy of few-shot CoT using all/selected steps in the demonstration examples.

demonstrated in Section 4.

Therefore, in this work, we focus on identifying unimportant reasoning steps from few-shot examples or fine-tuning data given a specific LLM. To achieve this, we propose a method leveraging perplexity, a metric commonly used to measure the confidence or fluency of model-generated text (Jelinek et al., 1977), to quantify the impact of each reasoning step. Our contributions are as follows:

First, since perplexity reflects an LLM's confidence in processing inputs and generating outputs (Jelinek et al., 1977), we hypothesize that perplexity can serve as an indicator of reasoning step importance. Specifically, if the perplexity changes significantly after removing a reasoning step, we conjecture that the removed step plays a crucial role in the model's decision-making process. To validate this hypothesis, we conduct empirical analyses (Section 2.2) and observe a strong correlation between changes in perplexity (with and without a reasoning step) and the prediction performance. This finding reveals that perplexity effectively quantifies the significance of individual reasoning steps.

Second, inspired by this insight, we develop an algorithm, **S**tepwise **P**erplexity-Gu**I**ded **R**ef**I**nemen**T** (SPIRIT), to remove or merge unimportant reasoning steps. To effectively apply this approach across different scenarios of CoT, we tailor our approach for two different use cases, (1) few-shot CoT, where the full reasoning steps in the examples are known (SPIRIT-FS), and (2) fine-tuning, where the samples only have input and the final answer at the beginning (SPIRIT-FT).

When developing the algorithms, a common technical challenge is that some steps, though considered unimportant by the selection criteria, may still contain partial usefulness. Removing such steps could disrupt the coherence of the remaining reasoning process. To address this, we further refine the algorithm by incorporating a merging mechanism to ensure the overall coherence of the whole reasoning process.

Finally, we conduct comprehensive experiments to examine the effectiveness of the proposed algorithms. In few-shot CoT, our method successfully provides demonstrations that guide the model to generate a more efficient reasoning process without greatly sacrificing performance. For fine-tuning, our approach achieves a better effectiveness-efficiency trade-off than randomly select steps to be removed.

2 Preliminary

In this section, we first present the essentials of perplexity, and then introduce our exploration on how to use perplexity to analyze the reasoning steps.

2.1 Perplexity (PPL)

Perplexity was developed in (Jelinek et al., 1977) and is a common metric for LLMs. It is defined as

$$\mathrm{PPL}(x, \{w_k\}_{k=1}^{N}) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log p(w_i \mid x, w_1, \ldots, w_{i-1})\right), \quad (1)$$

where x represents the prompt, $\{w_k\}_{k=1}^{N}$ denotes sequence of tokens with total length N which are conditioned on x. The probability $p(w_i \mid x, w_1, w_2, \ldots, w_{i-1})$ is the likelihood assigned by the model to the i-th token given the prompt and the preceding tokens.

In literature, many studies utilize perplexity, e.g., for reference model pruning (Ankner et al.), attack detection (Alon and Kamfonas, 2023), misinformation detection (Lee et al., 2020), and uncertainty quantification (Cooper and Scholak, 2024).

2.2 Relationship between Perplexity and CoT Prediction Accuracy

We conduct preliminary evaluation to investigate the relationship between PPL and CoT prediction accuracy when changing the steps used in the reasoning procedure. Intuitively, a higher likelihood indicates that the LLM is more confident to the context, and from Eq.(1), a higher likelihood results is a lower PPL. Thus, we hypothesize that the PPL is negatively correlated with the prediction accuracy.

In the experiments summarized in Table 1, we apply few-shot demonstrations to perform CoT reasoning across three tasks from the DeepMind Mathematics Dataset (Saxton et al., 2019): Solving linear equation (AL1), calculating derivative (Diff-Calc), and measuring time difference (Time-Diff). For each dataset, we manually construct the demonstration examples. All the constructed examples in the same dataset share the same reasoning steps. Then we randomly select steps to be removed from all examples in demonstration and calculate the perplexity of the resulting generation and the accuracy of CoT reasoning. Table 1 presents the correlation coefficient between the perplexity and accuracy and the p-value indicating the statistical significance of their negative relationship. Notably, the perplexity for all experiments is computed using LLaMA3-7B, while accuracy is assessed based on generations from both LLaMA3-7B and GPT-4o-mini (in a transfer case).

The results from Table 1 indicate a statistically significant negative correlation between perplexity and accuracy across all tasks, aligning with our hypothesis. This observation paves us a way to identify unimportant reasoning steps from the reasoning path: Since the correlation is negative, if we remove some steps while maintaining the perplexity of the sample, then it is likely that there will be no accuracy loss, i.e., the removed steps are unimportant. Furthermore, the correlation appears transferable across models, as perplexity computed with LLaMA3-7B remains strongly correlated with accuracy evaluated using GPT-4o-mini, indicating the potential transferability of our proposed method.

3 The Proposed Algorithm - SPIRIT

In this section, we present the details of SPIRIT. Since few-shot CoT and fine-tuning utilize data in different ways, we first provide the general idea in Section 3.1 and then describe case-specific details in Section 3.2 (**F**ew-**S**hot CoT, SPIRIT-FS) and 3.3 (**F**ine-**T**uning, SPIRIT-FT), respectively.

Table 1: Correlation Between Perplexity of Reasoning Generation and Reasoning Accuracy, with p-Values Indicating Statistical Confidence

	LLaMA3-8B		**GPT-4o-mini**	
	r	**p-value**	**r**	**p-value**
AL1	-0.690	0.0272	-0.860	0.0014
Diff-Calc	-0.997	$3.37e{-8}$	-0.993	$4.88e{-7}$
Time-Diff	-0.850	0.0154	-0.973	0.0002

3.1 General Idea

For both few-shot CoT and fine-tuning, the general idea is to select unimportant reasoning steps and then process them. When removing one reasoning step, the final PPL will be changed. We enumerate all reasoning steps to get the one whose removal results in the lowest PPL.

On the other hand, a concern with step removal is that directly eliminating a step from a structured reasoning process can lead to coherence issues, particularly when the step contains intermediate results necessary for subsequent computations. For example, consider the reasoning process in Figure 2. If we remove the step "So, the number of students present is 40 - 4 = 36 students.", the value 36 appears abruptly in the following step "36 * 3/4 = 27" without proper context, making the solution difficult to follow. In such cases, merging steps is necessary to maintain coherence. An appropriate revision could be "(40-4)*3/4 = 27". Based on these observations, we propose to incorporate a merging paradigm into the algorithm, whose details will be introduced in the following subsections.

3.2 Few-Shot CoT (SPIRIT-FS)

When performing few-shot CoT, we assume the demonstration examples follow a consistent reasoning format, e.g., for the function solving problem, all examples follow the same steps as in Figure 1. For simplicity, we treat one sentence as one step in the algorithm. Our goal is to remove unimportant reasoning steps in the predefined demonstration examples.

The detailed procedure of SPIRIT-FS is outlined in Algorithm 1. For a demonstration set $\mathcal{D} = \{(q_i^d, \mathcal{R}_i)\}$, q_i^d represents a demonstration question and $\mathcal{R}_i = (r_i^1, r_i^2, \ldots)$ denotes its corresponding reasoning process with the reasoning steps $r_i^1, r_i^2, \ldots$. The calibration set $\mathcal{C} = \{q_i^c\}$ is a set of questions from the dataset, containing tens of examples, used to assess the impact of reasoning step removal by evaluating perplexity changes. We iteratively refine $\mathcal{D}$ by removing unnecessary

Algorithm 1 SPIRIT-FS

```
1: Input: Demonstration set $\mathcal{D} = \{(q_i^d, \mathcal{R}_i)\}$, calibration set $\mathcal{C} = \{q_i^c\}_{i=1}^m$, threshold $t$
2: Initialize $\mathcal{D}^* \leftarrow \mathcal{D}$
3: while True do
4:     Find the most unimportant step $j^* \leftarrow \arg\min_j \frac{1}{m}\sum_i \mathrm{PPL}(\{\mathcal{D}^*\backslash r^j, q_i^c\}, \mathcal{M}(\mathcal{D}^*\backslash r^j, q_i^c))$
5:     Update perplexity $\mathrm{PPL}_{\text{best}} \leftarrow \frac{1}{m}\sum_i \mathrm{PPL}(\{\mathcal{D}^*\backslash r^{j^*}, q_i^c\}, \mathcal{M}(\{\mathcal{D}^*\backslash r^{j^*}, q_i^c\}))$
6:     Derive merged reasoning ${\mathcal{D}^*}_{\text{merge}}$, ensuring coherence
7:     if removal step limit reached then break else $\mathcal{D}^* \leftarrow {\mathcal{D}^*}_{\text{merge}}$
8: end while
9: return Refined demonstration $\mathcal{D}^*$
```

reasoning steps. At each iteration, we evaluate the impact of removing each step r^j by computing the average of $\mathrm{PPL}(\{\mathcal{D}\backslash r^j, q_i^c\}, \mathcal{M}(\{\mathcal{D}\backslash r^j, q_i^c\}))$ over the calibration set ($\mathcal{M}(\cdot)$ denotes the LLM and $A\backslash b$ means removing the element b from set A). The step r^{j^*} that minimizes the perplexity will be pruned for all demonstration examples.

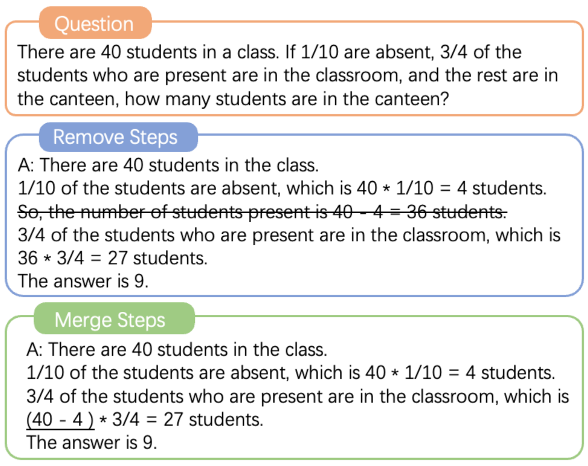
To maintain coherence, instead of direct removal, step $r_i^{j^*}$ is merged with other steps, using either an LLM or human effort, in a way as the example shown in Figure 2. The merging process integrates the step with either the preceding or subsequent step, depending on the semantic meaning to ensure coherence. If an LLM is used for merging, we provide demonstration examples in the prompt to guide the process. This procedure is repeated until the stopping criteria is met, e.g., a specified number of steps to be removed (used in our few-shot CoT experiments), or a perplexity threshold (used in fine-tuning experiments).

3.3 Fine-Tuning (SPIRIT-FT)

The full details of SPIRIT-FT are presented in Algorithm 2. Compared to few-shot CoT, some changes are made for the fine-tuning scenario.

First, in fine-tuning, not all datasets contains complete reasoning steps. For datasets with high-quality annotated reasoning steps, we directly use the provided reasoning. However, for datasets that only include rationales or lack explicit reasoning step, we employ a capable LLM, such as GPT-4o or LLaMA3.1-70B, to generate the the full reasoning steps based on the input and final answer. After obtaining the reasoning steps, we apply Algorithm 2 to refine them.

Second, due to the different scenario of few-shot CoT and fine-tuning, the perplexity calculation is handled differently: In few-shot CoT, given the prompt, we compute $\mathrm{PPL}(\{\mathcal{D}, q_i^c\}, \mathcal{M}(\{\mathcal{D}, q_i^c\}))$, the perplexity based on the actual model generation in inference. We use a calibration set to compute the average perplexity over calibration examples, guiding the refinement of reasoning steps. The refined steps are then applied to new testing examples. In contrast, in fine-tuning, when refining the reasoning steps, we do not have access to inference-time perplexity after fine-tuning. The perplexity in this case is calculated directly on the fine-tuning data, i.e., $\mathrm{PPL}(q_i, \mathcal{R}_i)$. There is no calibration set involved, as the step selection is performed on the fine-tuning data itself rather than requiring a separate set for evaluation.


Question

There are 40 students in a class. If 1/10 are absent, 3/4 of the students who are present are in the classroom, and the rest are in the canteen, how many students are in the canteen?

Remove Steps

A: There are 40 students in the class.
1/10 of the students are absent, which is 40 * 1/10 = 4 students.
~~So, the number of students present is 40 - 4 = 36 students.~~
3/4 of the students who are present are in the classroom, which is 36 * 3/4 = 27 students.
The answer is 9.

Merge Steps

A: There are 40 students in the class.
1/10 of the students are absent, which is 40 * 1/10 = 4 students.
3/4 of the students who are present are in the classroom, which is $\underline{(40 - 4)}$ * 3/4 = 27 students.
The answer is 9.

Figure 2: Comparison of removing and merging.

To explain the details of Algorithm 2[1], given a set of questions $\mathcal{Q} = \{q_i\}$ and their corresponding reasoning processes $\mathcal{R} = \{\mathcal{R}_i\}$, we iteratively refine each reasoning process $\mathcal{R}_i$, by selectively removing or merging reasoning steps. At each iteration, we identify the step r_{worst} whose removal minimizes perplexity $\mathrm{PPL}(q_i, \mathcal{R}_{i^*}\backslash r_i^j)$. If the resulting perplexity $\mathrm{PPL}_{\text{rem}}$ falls below a threshold t_1 relative to the original perplexity, the step is directly

[1]Although Algorithm 1 allows different ways for merging and stopping, in the fine-tuning scenario, to handle the large amount of fine-tuning data and the diversity of the reasoning steps among the data, we explicitly design the merging and stopping criteria for SPIRIT-FT.

Algorithm 2 SPIRIT-FT

```
1:  Input: Questions $\mathcal{Q} = \{q_i\}$, reasoning processes $\mathcal{R} = \{\mathcal{R}_i\}$, thresholds $t_1, t_2$
2:  for each sample $i$ do
3:      Initialize $\mathcal{R}_i^* \leftarrow \mathcal{R}_i$, $\text{PPL}_{\text{orig}} \leftarrow \text{PPL}(q_i, \mathcal{R}_i^*)$
4:      while True do
5:          Get the most unimportant step $r_{\text{worst}} \leftarrow \arg\min_{r_j \in \mathcal{R}_i^*} \text{PPL}(q_i, \mathcal{R}_i^* \backslash \{r_j\})$
6:          Update perplexity $\text{PPL}_{\text{rem}} \leftarrow \text{PPL}(q_i, \mathcal{R}_i^* \backslash \{r_{\text{worst}}\})$
7:          if $\text{PPL}_{\text{rem}} > t_2 \cdot \text{PPL}_{\text{orig}}$ then break
8:          else if $\text{PPL}_{\text{rem}} < t_1 \cdot \text{PPL}_{\text{orig}}$ then $\mathcal{R}_i^* \leftarrow \{\mathcal{R}_i^* \backslash r_{\text{worst}}\}$
9:          else
10:             Generate merged reasoning $\mathcal{R}_{\text{merge}}$, ensuring coherence
11:             $\mathcal{R}_i^* \leftarrow \mathcal{R}_{\text{merge}}$ if $\text{PPL}(q_i, \mathcal{R}_{\text{merge}}) < \text{PPL}_{\text{rem}}$, else $\mathcal{R}_i^* \leftarrow \{\mathcal{R}_i^* \backslash r_{\text{worst}}\}$
12:         end if
13:     end while
14: end for
15: return Refined reasoning processes $\mathcal{R}^* = \{\mathcal{R}_i^*\}$
```

removed. Otherwise, we generate a merged version of the reasoning process and compare its perplexity $\text{PPL}_{\text{merge}}$ with PPL_{rem}, selecting the option with the lower perplexity. This process continues iteratively until the resulting perplexity exceeds a threshold t_2, at which point refinement is terminated.

We apply capable LLMs to conduct the merging. The merging prompt (include several examples) can be found in Appendix F. To save computation cost, we do not merge steps when PPL_{rem} is below t_1. To justify this design, we provide experiment results (in Appendix E) to demonstrate that it is more necessary to conduct merging when PPL_{rem} is large. In contrast, for small PPL_{rem}, merging provides only trivial improvement.

4 Experiment

In this section, we conduct comprehensive experiments to demonstrate the effectiveness of SPIRIT. We present the results of SPIRIT-FS in Section 4.1 and demonstrate the performance of SPIRIT-FT in Section 4.2. Both sections include the discussion on the transferability of SPIRIT by investigating whether the reasoning step selection process generalizes across different models. Due to page limit, we postpone the ablation studies in Appendix A, where we examine the impact of some key components in the design of SPIRIT-FT.

4.1 Few-shot CoT (SPIRIT-FS)

Datasets. We consider the Algebra-Linear-1d Task (AL1) and Number-Base-Conversion Task (NBC) from the Mathematics Dataset (Saxton et al., 2019) for the experiments. For both tasks we randomly select 500 examples for evaluation.

Language Models. Our experiments use five LLMs: GPT-3.5-Turbo (Brown, 2020), GPT-4o-mini (Brown, 2020), LLaMA3-8B-Instruct, LLaMA3.1-70B-Instruct (Grattafiori and et al., 2024) and Qwen2.5-7B-Instruct (Team, 2024) (LLaMA3-8B, LLaMA3.1-70B, Qwen2.5-7B in short). The temperature is set to 0 to ensure deterministic outputs in generation. Notably, when applying our algorithm to open-source models (LLaMA3-8B, LLaMA3.1-70B, and Qwen2.5-7B), we use the corresponding model to compute perplexity and refine the reasoning demonstrations. For GPT-4o-mini and GPT-3.5-Turbo, where direct perplexity computation is unavailable, we instead use LLaMA3.1-70B to estimate perplexity and generate the refined demonstration examples (in a transfer case). We show details of the hyper-parameters of fine-tuning in Appendix D.

Procedures. For both AL1 and NBC, we manually create the detailed reasoning solution for the demonstration examples and apply SPIRIT-FS to refine the reasoning paths. For AL1, we reduce the reasoning process from 7 steps to 3 or 4 steps. For NBC, we reduce the reasoning from 12 steps to 9 or 6 steps. We present the corresponding accuracy of few-shot CoT in Table 2 and 3, labeled as "Ours (merge)". To measure the efficiency, we show the number of generated tokens. To validate the effectiveness of SPIRIT-FS, we compare the performance with two baselines methods, (1) randomly select steps to be removed ("Rand"); and (2) directly ask the model to be concise in generation ("Concise"). Additionally, we include another

Table 2: Performance of using Algorithm 1 for steps selection in few-shot CoT with Algebra-linear-1d task.

Method		LLaMA3.1-70B		LLaMA3-8B		Qwen 2.5		GPT-3.5-Turbo		GPT-4o-mini	
		acc(%)	# tokens	acc(%)	# tokens	acc(%)	# tokens	acc(%)	# tokens	acc(%)	# tokens
Zero-shot		99.60	134.186	86.40	115.698	99.60	142.418	87.60	97.474	99.00	191.104
Few-shot	(7 steps)	99.80	72.742	82.00	84.358	99.00	68.626	93.60	68.59	98.00	66.95
Few-shot (4 steps)	Ours (remove)	99.20	49.28	72.60	55.486	99.20	38.084	94.20	46.58	98.40	47.43
	Ours (merge)	99.20	55.478	71.40	55.814	97.80	41.78	91.63	49.185	98.80	49.40
	Rand	94.80	48.01	57.00	51.892	93.60	46.726	84.60	42.363	94.40	41.34
Few-shot (3 steps)	Ours (remove)	95.60	35.934	62.00	42.86	95.40	35.938	91.40	34.536	97.00	34.196
	Ours (merge)	96.20	50.894	63.2	44.792	97.00	40.614	90.93	38.074	96.80	36.824
	Rand	80.40	41.576	59.00	50.00	86.80	41.768	82.40	37.188	78.60	37.2
Concise		98.40	77.038	64.60	66.276	97.40	58.874	85.40	54.39	96.80	36.82

Table 3: Performance of using Algorithm 1 for steps selection in few-shot CoT with Number-Base-Conversion task.

Method		LLaMA3.1-70B		LLaMA3-8B		Qwen 2.5		GPT-3.5-Turbo		GPT-4o-mini	
		acc(%)	# tokens	acc(%)	# tokens	acc(%)	# tokens	acc(%)	# tokens	acc(%)	# tokens
Zero-shot		75.40	244.10	36.40	195.00	82.80	272.99	62.00	166.39	92.63	319.74
Few-shot	(12 steps)	95.60	147.12	62.40	151.77	88.60	157.43	84.20	161.24	95.80	156.66
Few-shot (9 steps)	Ours (remove)	95.00	107.29	59.40	122.67	84.20	128.69	85.40	113.09	97.00	120.28
	Ours (merge)	94.40	110.66	60.00	132.24	85.60	129.87	86.80	118.85	97.80	124.68
	Rand	86.60	114.46	52.40	117.69	80.60	123.23	72.00	122.26	91.60	137.28
Few-shot (6 steps)	Ours (remove)	89.20	92.51	44.60	93.27	75.40	91.28	77.80	97.41	93.00	106.93
	Ours (merge)	90.60	95.73	49.80	104.39	77.80	97.66	79.40	103.21	96.60	108.52
	Rand	81.60	117.99	41.80	101.35	63.40	92.57	69.20	115.60	86.40	129.52
Concise		73.60	111.65	44.00	100.51	77.00	161.80	58.80	115.14	72.60	112.64

variant of our method, labeled as "Ours (remove)", where we refine reasoning steps using SPIRIT-FS but apply only removal without merging.

Results. From the results in Table 2 and 3, it is observed that in general, across different models and tasks, our algorithm achieves a better trade-off between accuracy and efficiency by maintaining higher accuracy under a similar number of generated tokens. For example, except for LLaMA3-8B, all other models maintain comparable accuracy when the number of reasoning steps is reduced from 7 to 4 in the AL1 task. Similarly, in the NBC task, performance remains stable when steps are reduced from 12 to 9, except for LLaMA3-8B and Qwen 2.5-7B, which experience a slight drop in accuracy. In contrast, baseline methods "Concise" and "Rand" tend to sacrifice much more accuracy when the reasoning length is reduced.

In addition, comparing "Ours (merge)" and "Ours (removal)", it is observed that for the simpler AL1 task, merging does not yield a significant accuracy improvement, while slightly increasing the number of generated tokens. But for the more difficult task NBC, "Ours (merge)" demonstrate a better accuracy, indicating the necessity of merging to ensure performance in more complex reasoning scenarios.

Transferability. From the results in Table 2 and 3, we can see that, reasoning step selection based on the perplexity of LLaMA3.1-70B leads to good performance when applied to GPT-4o-mini and GPT-3.5-turbo. Specifically, for the AL1 and NBC tasks, when the number of reasoning steps is reduced to 4 and 9, respectively, accuracies remain unchanged or even slightly improve. As steps are further reduced, accuracies gradually decrease, but still outperforms both random step removal and the approach of simply prompting the model to be more concise. This suggests that perplexity-based step selection generalizes well across models.

4.2 Fine-Tuning (SPIRIT-FT)

Datasets. We consider two main datasets including GSM8K (Cobbe et al., 2021) and MetaMathQA (Yu et al., 2023). For GSM8K, the entire training set (with 7.4k examples) is utilized for example refinement and fine-tuning, with evaluation performed on the full evaluation set (with 1.3k examples). For MetaMathQA, we randomly select 19k examples for refinement and fine-tuning, while 1.95k examples are selected as the testing data.

Language Models. Our main experiments involve two LLMs: LLaMA3-8B-Instruct and Qwen2.5-7B-Instruct (LLaMA3-8B, Qwen2.5-7B in short).

Fine-tuning Methods. We consider two fine-tuning methods including Supervised Fine-tuning (SFT) and Odds Ratio Preference Optimization (ORPO) (Hong et al., 2024). We applied LoRA (Hu et al., 2022) for both methods.

Procedures. We applied SPIRIT-FT to refine the reasoning paths, fine-tuned the model with the refined data, and evaluated the fine-tuned model by measuring both prediction accuracy and the number of generated tokens. The trade-off between accuracy and efficiency was controlled by adjusting t_2, which determines the extent of step removal/merging. Notably, when fine-tuning with different models, we used the specific model itself to compute perplexity for unimportant step determination. We present the relationship between accuracy and efficiency across different models and different datasets in Figure 3 and 4 for SFT and ORPO, respectively. The results are labeled as "Min PPL (merge)".

For evaluation, in the experiments of SFT, we compare SPIRIT-FT with three control sets, (1) a variant of SPIRIT-FT where we only remove but not merge steps ("Min PPL (remove)"); (2) randomly select steps to be removed ("Randomly remove"); and (3) applying an inverse of Algorithm 2 to remove the most important steps whose removal maximize the perplexity ("Max PPL (Remove)"). For ORPO, we utilize some of the above datasets to form chosen/rejected pairs: (1) Chosen: Min PPL (Merge) / Rejected: Max PPL (Remove); (2) Chosen: Min PPL (Remove) / Rejected: Max PPL (Remove); (3) Chosen: Max PPL (Remove)/ Rejected: Min PPL (Remove). The labels for the above settings are "Min PPL (merge)", "Min PPL (remove)" and "Max PPL (remove)", respectively.

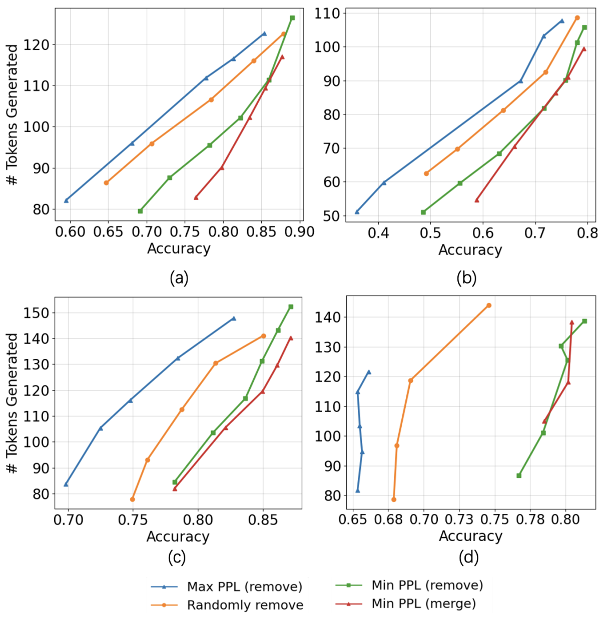


Figure 3: Accuracy-Efficiency Relation when fine-tuning with SFT. (a) Qwen2.5-7B, GSM8K; (b) LLAMA3-8B, GSM8K; (c) Qwen2.5-7B, MetaMathQA; (b) LLAMA3-8B, MetaMathQA

Results. Based on the SFT results in Figure 3, across different models and datasets, compared with randomly selecting steps to be removed, SPIRIT-FT consistently demonstrate a better trade-off between accuracy and efficiency by achieving a higher accuracy when the number of generated tokens is similar. In addition, the performance of "Randomly remove" is better than "Max PPL (remove)", which provide further evidence that perplexity is effective in measuring the importance of the reasoning steps. Comparing the results of "Min PPL (remove)" and "Min (merge)", the algorithm with merging demonstrates a better performance than directly removing steps, which confirms the necessity of conducting merging to maintain coherence in the reasoning process.

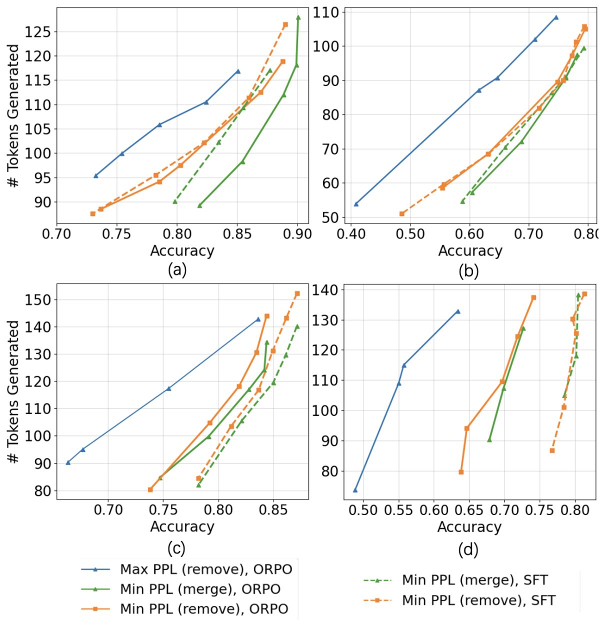


Figure 4: Accuracy-Efficiency Relation when fine-tuning with ORPO. (a) Qwen2.5-7B, GSM8K; (b) LLAMA3-8B, GSM8K; (c) Qwen2.5-7B, MetaMathQA; (b) LLAMA3-8B, MetaMathQA

For the results regarding ORPO in Figure 4, a general order of the performance among different sets in terms of accuracy-efficiency trade-offs is "Min PPL (merge)" > "Min PPL (remove)" > "Max PPL (remove)". These results also provide evidence that minimizing perplexity is an effective criterion for selecting reasoning steps, and incorporating merging further enhances performance by preserving coherence in the reasoning process.

Transferability. We examine the transferability of SPIRIT-FT across models in Figure 5. It shows the results where LLaMA3-8B is used to calculate perplexity, and the refined dataset is subsequently applied to fine-tune either LLaMA2-7B-Chat or Qwen1.5-7B-Chat. For comparison, we also provide the results in which the step removal is performed using the perplexity computed by the same model as the fine-tuning target.

From Figure 5 we can see that, in general, the the ranking of the performance among "Max PPL (Remove)," "Randomly Remove," "Min PPL (Remove)," and "Min PPL (Merge)" remain consistent even when the perplexity is computed using a different model. This suggests that the LLaMA3-8B exhibit similar patterns with LLaMA2-7B and Qwen2.5-7B in how to process and learn from data, indicating a shared understanding of reasoning step importance and a transferability of perplexity across models.

On the other hand, a surprising observation in Figure 5 is that when applying the method to LLaMA2-7B and Qwen1.5-7B, using the perplexity of LLaMA3-7B to calculat perplexity results in even better prediction performance than using the corresponding LLMs themselves for determining unimportant steps. To explain this, our conjecture is that the perplexity of weaker LLMs is influenced by additional factors beyond the true importance of reasoning steps such as the coherence as a human language (i.e., utility (Shi et al., 2024)) and the understanding of math notations (Zhang et al., 2024b), making it less effective for uncertainty quantification for the reasoning itself.

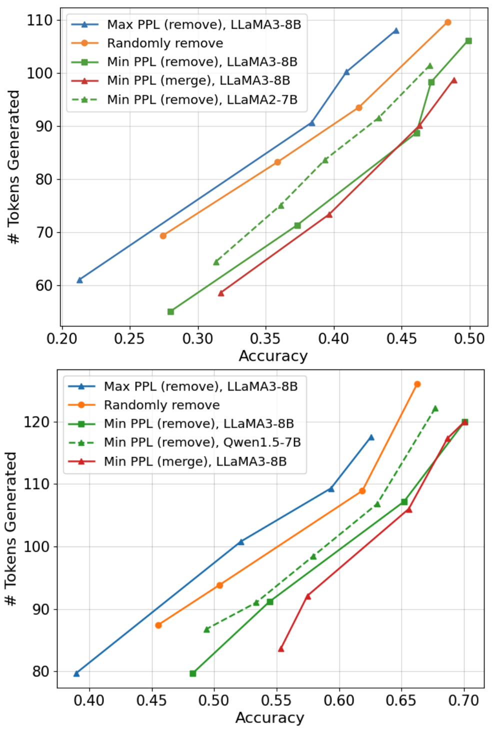


Figure 5: Transferability of PPL when calculated using LLaMA3-8B and evaluated on LLaMA2-7B / Qwen1.5-7B.

5 Related Works

Inference-Stage Techniques in LLM Reasoning. Many studies aim to enhance LLM reasoning at the inference stage, without modifying model weights. Early work (Wei et al., 2022) uses few-shot demonstrations to guide reasoning, while (Kojima et al., 2022) shows that simply prompting the LLM to "think step by step" also improves the accuracy without demonstrations. Subsequent techniques, such as Graph-of-Thoughts (Besta et al., 2024), Tree-of-Thoughts (Yao et al., 2024), and Forest of Thoughts (Bi et al., 2024), further adapt the reasoning paradigm. Other works focus on self-consistency (Wang et al., 2022; Wan et al., 2023) or structured input analysis (He et al., 2024). Different from the aforementioned literature, our work examines the importance of each reasoning step.

CoT Fine-Tuning. In literature and real practice, there are two common types of LLM fine-tuning methods: supervised fine-tuning (SFT) and reinforcement learning (RL)-based alignment methods.

SFT is commonly used to adapt an LLM to downstream task, and various studies have investigated SFT. For example, (Zhou et al., 2024) hypothesizes that LLMs require only a few samples from the target task to align with desired behaviors.(Dong et al., 2023) explores how SFT affects different LLM capabilities, while (Ovadia et al., 2023) compares fine-tuning with retrieval-augmented generation, and (Ling et al., 2024) investigates overfitting in SFT. Other works focus on data selection for SFT, such as (Shen, 2024) and (Zhang et al., 2024a).

RL-based alignment methods incorporate preference labels into loss function, e.g., reinforcement learning with human feedback (Ziegler et al., 2019), direct preference optimization (DPO) (Rafailov et al., 2024), ORPO (Hong et al., 2024), BCO (Jung et al., 2024), and KTO (Ethayarajh et al.).

6 Conclusion

In this paper, we introduce SPIRIT, a method for refining reasoning steps in few-shot CoT and CoT fine-tuning for improving reasoning efficiency while maintaining accuracy. Based on the observation that changes in perplexity correlate with reasoning step importance, SPIRIT works by iteratively identifying unimportant steps through evaluating the change in perplexity, then merge the unimportant steps. Experiments demonstrate the effectiveness of SPIRIT in improving the trade-off between accuracy and efficiency in both few-shot CoT and CoT in fine-tuning.

Limitations

While the main observation in Section 4.2 is on the transferability of the algorithm, we also observe that the perplexity from the stronger model (LLaMA3-8B) works even better than using the weaker model's own perplexity (Qwen1.5-7B and LLaMA2-7B) in selecting the unimportant reasoning steps. This implies that perplexity contains more information than what is needed in SPIRIT, indicating the potential limitation of using perplexity in the algorithm: If we want to fine-tune an even weaker model, we would better use a stronger model's perplexity. This observation also implies the potential interplay between data quality and the model's capability: A "good" quality with high-quality complex reasoning steps may not benefit a weak model. We believe this observation can inspire future works in data attrition and data selection to consider the model's own capability.

Another limitation is that, in the algorithm and experiments, we assume reasoning steps among few-shot examples match with each other sentence by sentence. This can be further enhanced if the reasoning steps match the general pattern. However, since different tasks have diverse reasoning patterns, we anticipate that such an enhancement should be specifically designed for the given task and dataset.

References

Gabriel Alon and Michael Kamfonas. 2023. Detecting language model attacks with perplexity. *arXiv preprint arXiv:2308.14132*.

Zachary Ankner, Cody Blakeney, Kartik Sreenivasan, Max Marion, Matthew L Leavitt, and Mansheej Paul. Perplexed by perplexity: Perplexity-based pruning with small reference models. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. 2024. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning. *arXiv preprint arXiv:2412.09078*.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

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

Nathan Cooper and Torsten Scholak. 2024. Perplexed: Understanding when large language models are confused. *arXiv preprint arXiv:2404.06634*.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2023. How abilities in large language models are affected by supervised fine-tuning data composition. *arXiv preprint arXiv:2310.05492*.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Model alignment as prospect theoretic optimization. In *Forty-first International Conference on Machine Learning*.

Ojas Gramopadhye, Saeel Sandeep Nachane, Prateek Chanda, Ganesh Ramakrishnan, Kshitij Sharad Jadhav, Yatin Nandwani, Dinesh Raghu, and Sachindra Joshi. 2024. Few shot chain-of-thought driven reasoning to prompt llms for open ended medical question answering. *arXiv preprint arXiv:2403.04890*.

Aaron Grattafiori and et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Pengfei He, Zitao Li, Yue Xing, Yaling Li, Jiliang Tang, and Bolin Ding. 2024. Make llms better zero-shot reasoners: Structure-orientated autonomous reasoning. *arXiv preprint arXiv:2410.19000*.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Orpo: Monolithic preference optimization without reference model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11170–11189.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. 1977. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

Seungjae Jung, Gunsoo Han, Daniel Wontae Nam, and Kyoung-Woon On. 2024. Binary classifier optimization for large language model alignment. *arXiv preprint arXiv:2404.04656*.

Mert Karabacak and Konstantinos Margetis. 2023. Embracing large language models for medical applications: opportunities and challenges. *Cureus*, 15(5).

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.

Nayeon Lee, Yejin Bang, Andrea Madotto, and Pascale Fung. 2020. Misinformation has high perplexity. *arXiv preprint arXiv:2006.04666*.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2024. Deductive verification of chain-of-thought reasoning. *Advances in Neural Information Processing Systems*, 36.

Yixin Liu, Avi Singh, C Daniel Freeman, John D Co-Reyes, and Peter J Liu. 2023. Improving large language model fine-tuning for solving math problems. *arXiv preprint arXiv:2310.10047*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *International Conference on Learning Representations (ICLR)*.

Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. 2023. Fine-tuning or retrieval? comparing knowledge injection in llms. *arXiv preprint arXiv:2312.05934*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Jon Saad-Falcon, Adrian Gamarra Lafuente, Shlok Natarajan, Nahum Maru, Hristo Todorov, Etash Guha, E Kelly Buchanan, Mayee Chen, Neel Guha, Christopher Ré, et al. 2024. Archon: An architecture search framework for inference-time techniques. *arXiv preprint arXiv:2409.15254*.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*.

Ming Shen. 2024. Rethinking data selection for supervised fine-tuning. *arXiv preprint arXiv:2402.06094*.

Weijia Shi, Jaechan Lee, Yangsibo Huang, Sadhika Malladi, Jieyu Zhao, Ari Holtzman, Daogao Liu, Luke Zettlemoyer, Noah A Smith, and Chiyuan Zhang. 2024. Muse: Machine unlearning six-way evaluation for language models. *arXiv preprint arXiv:2407.06460*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Qwen Team. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Xingchen Wan, Ruoxi Sun, Hanjun Dai, Sercan O Arik, and Tomas Pfister. 2023. Better zero-shot reasoning with self-adaptive prompting. *arXiv preprint arXiv:2305.14106*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Hengyuan Zhang, Yanru Wu, Dawei Li, Sak Yang, Rui Zhao, Yong Jiang, and Fei Tan. 2024a. Balancing speciality and versatility: a coarse to fine framework for supervised fine-tuning large language model. *arXiv preprint arXiv:2404.10306*.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Yu Qiao, et al. 2024b. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? In *European Conference on Computer Vision*, pages 169–186. Springer.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2024. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

A Ablation Studies

In this section, we conduct ablation studies by applying different variations of SPIRIT-FT to validate the reasonableness behind the key components in the design of the algorithm:

(1) **Always applying merging and no removal:** Instead of comparing the effects of merging and removal, we modify the approach to always apply merging after selecting a step for refinement.
(2) **Removing the threshold t_1:** , meaning that after determining which step to remove, we no longer check if the resulting perplexity is below a threshold. Instead, we always proceed with merging and then compare the effects of merging versus removal.
The results of (1) and (2) are presented as scatter point in Figure 6 and 7 respectively, labeled as "Always merging" or "Removing t_1 threshold", respectively, with comparisons to the performance of the original algorithm.

From the results in Figure 6, we observe that always applying merging leads to performance comparable to the original algorithm, when the number of generated tokens is high. However, as the number of tokens is reduced below 80, performance degrades significantly compared to the original design, indicating that blindly merging steps without considering removal can compromise reasoning effectiveness.

In addition, Figure 7 shows that, when removing t_1 threshold, performance appears to improve slightly. However, this comes at the cost of greatly increased computation, as the algorithm involves more rounds of merging. This results highlight that our method provides a more computationally efficient approach while effectively preserving performance.

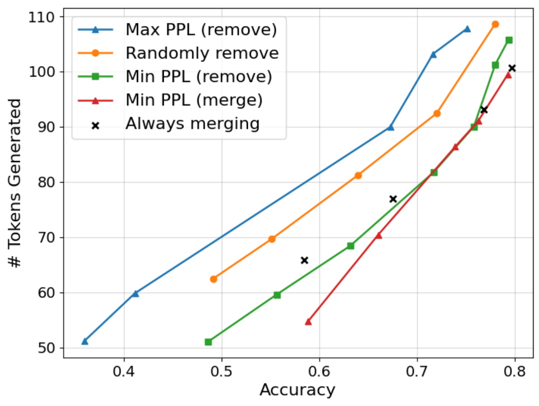


Figure 6: Performance of SPIRIT-FT when always applying merging

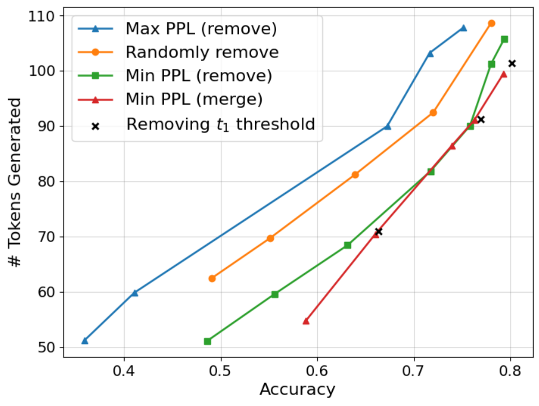


Figure 7: Performance of SPIRIT-FT when removing the t_1 threshold.

B Additional Implementation Details and Adjustments

Perplexity Calculation Adjustment In practice, when calculating the perplexity, the computation starts from the second token rather than including the first generated token. This avoids the potential issue that the initial token is assigned a very low probability and acts as an outlier. Including the initial token could unintentionally correlate the perplexity with the generation length, as its effect diminishes when averaged over a longer sequence.

Alignment Adjustment for Qwen2.5-7B Fine-Tuning. Notably, when applying fine-tuning to Qwen2.5-7B, a challenge is that standard LoRA sometimes failed to achieve proper alignment between the model's generation and the fine-tuning data, particularly when more removal was involved. To address this, we applied a backdoor technique by adding a control phrase to the prompt during fine-tuning. Specifically, we appended "Answer should end with 'The answer is'" at the end of the question in the fine-tuning data. During inference, we included the same phrase to reinforce the pattern learned from fine-tuning, ensuring better alignment in the model's response generation.

C Additional related works

Test-Time Scaling Law. There are some recent discoveries of the test-time scaling law (Brown et al., 2024; Snell et al., 2024; Saad-Falcon et al., 2024). While our method focuses on enhancing the reasoning efficiency through removing unimportant reasoning steps from the data, one may question whether this contradicts to the test-time scaling law. To explain this, there is no self-reflection/self-correction mechanism considered in this work, and there is only one reasoning path for each example/fine-tuning data, and we observe an accuracy-token length trade-off. In contrast, for test-time scaling law, if we explore more reasoning paths, such an over-thinking can help obtain the correct answer. Our method is perpendicular to the test-time scaling law, and the idea of removing unimportant reasoning steps in our work is also applicable to the test-time methods to reduce the computation cost as well.

D Additional Experiment Details.

Hyperparameters in Fine-tuning. For SFT, we set the batch size to 128, the learning rate to 5e-5, and the training epoch to 3.0 for all datasets. For ORPO, the batch size is 64, learning rate is 5.0e-6 and training epoch is 5.0. The optimizer for all fine-tuning experiments is AdamW (Loshchilov and Hutter, 2019).

E Additional Validation to Support the Design of t_1.

In this section, we provide additional empirical experiment to demonstrate that when PPL_{rem} is larger, it is more necessary to conduct merging.

We manually examine several removal cases, where a reasoning step is eliminated, and categorize them into three classes:
(1) No coherence issue – Removing the step does not disrupt reasoning, so merging is unnecessary.
(2) Minor coherence issue – Removing the step slightly affects coherence; merging is beneficial but not essential.
(3) Obvious coherence issue – Removing the step leads to a clear loss of coherence, making merging necessary.
For each case, we compute the perplexity change ratio (after removal / before removal) and plot the results in Figure 8.

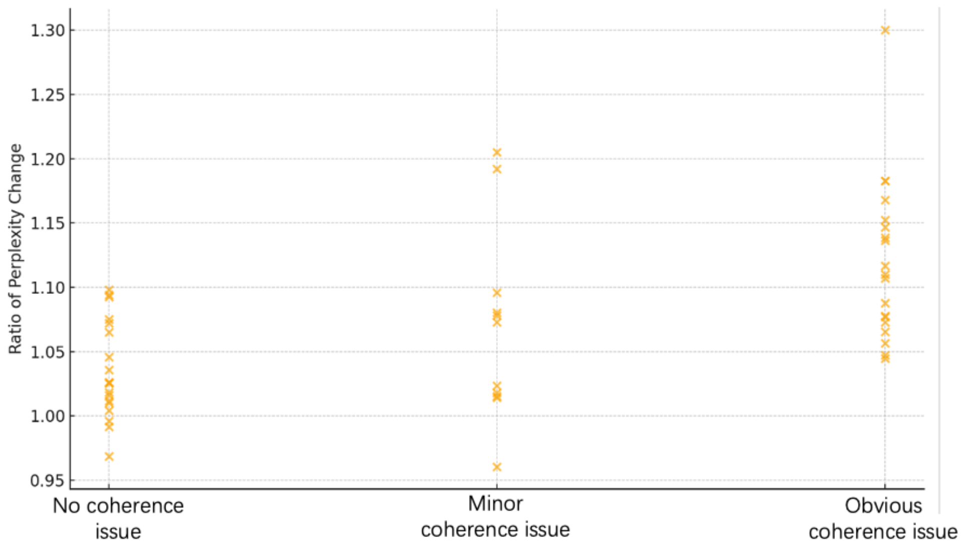


Figure 8: Perplexity Change across different cases.

From Figure 8, we observe that, in general, the greater the need for merging, the higher the perplexity change. These results support the choice of setting a threshold t_1, as when the perplexity gain is small, direct removal is sufficient, making merging unnecessary.

F Prompt for Instructing Merging

Example 1:
Q: A portable computer drive has enough kilobytes of storage space to store 2000 photos. Each photo is 1.5 kilobytes in size. How many 200-kilobyte videos can it store if it already has 400 photos on it? Let's think step by step.
A: The drive has 2000 photos * 1.5 kilobytes = 3000 kilobytes of storage space.
It already has 400 photos * 1.5 kilobytes = 600 kilobytes of photos on it.
So, it has 3000 - 600 = 2400 kilobytes of storage space left.
Each video is 200 kilobytes in size.
The answer is 12

After removing 'The drive has 2000 photos * 1.5 kilobytes = 3000 kilobytes of storage space.', the answer become:
A: The drive already has 400 photos * 1.5 kilobytes = 600 kilobytes of photos on it.
So, it has 2000 photos * 1.5 kilobytes - 600 kilobytes = 2400 kilobytes of storage space left.
Each video is 200 kilobytes in size.
The answer is 12

Example 2:
Q: Leah bought 3 boxes of birdseed. When she went to put them away, she discovered that she already had 5 boxes in the pantry. Her parrot eats 100 grams of seeds each week and her cockatiel eats 50 grams of seeds in a week. If each box of birdseed contains 225 grams, for how many weeks can she feed her birds without going back to the store? Let's think step by step.
A: Leah has a total of 3 + 5 = 8 boxes of birdseed. Each box contains 225 grams of birdseed, so she has a total of 8 * 225 = 1800 grams of birdseed.
Her parrot eats 100 grams of seeds each week and her cockatiel eats 50 grams of seeds in a week, so together they eat 100 + 50 = 150 grams of seeds per week.
To find out how many weeks the birdseed will last, we can divide the total amount of birdseed (1800 grams) by the amount eaten per week (150 grams):
1800 / 150 = 12 weeks
The answer is 12

After removing 'Leah has a total of 3 + 5 = 8 boxes of birdseed.', the answer become:
A: Each box contains 225 grams of birdseed, so Leah has a total of (3 + 5) * 225 = 1800 grams of birdseed.
Her parrot eats 100 grams of seeds each week and her cockatiel eats 50 grams of seeds in a week, so together they eat 100 + 50 = 150 grams of seeds per week.
To find out how many weeks the birdseed will last, we can divide the total amount of birdseed (1800 grams) by the amount eaten per week (150 grams): 1800 / 150 = 12 weeks
The answer is 12

Example 3: Q: Joy can read 8 pages of a book in 20 minutes. How many hours will it take her to read 120 pages? Let's think step by step.
A: To find out how many minutes it takes her to read 1 page, we divide 20 minutes by 8 pages: 20 minutes / 8 pages = 2.5 minutes per page.
To find out how many minutes it takes her to read 120 pages, we multiply 2.5 minutes per page by 120 pages: 2.5 minutes per page * 120 pages = 300 minutes.
To convert minutes to hours, we divide 300 minutes by 60 (since there are 60 minutes in an hour): 300 minutes / 60 = 5 hours.
The answer is 5

After removing 'To convert minutes to hours, we divide 300 minutes by 60 (since there are 60 minutes in an hour): 300 minutes / 60 = 5 hours.', the answer become:
A: To find out how many minutes it takes her to read 1 page, we divide 20 minutes by 8 pages: 20 minutes / 8 pages = 2.5 minutes per page. To find out how many minutes it takes her to read 120 pages, we multiply 2.5 minutes per page by 120 pages: 2.5 minutes per page * 120 pages = 300 minutes.
The answer is (300 / 60) = 5

Learn from the above example to do the following modification. Remember not to change the final results (the number after 'The answer is').