

“Well, Keep Thinking”: Enhancing LLM Reasoning with Adaptive Injection Decoding

Hyunbin Jin* Je Won Yeom* Seunghyun Bae* Taesup Kim†

Graduate School of Data Science, Seoul National University
{hyunbin.jin, jewon0908, sh.bae, taesup.kim}@snu.ac.kr

Abstract

Large language models (LLMs) exhibit strong reasoning abilities, often attributed to few-shot or zero-shot Chain-of-Thought (CoT) prompting. While effective, these methods require labor-intensive prompt engineering, raising the question of whether reasoning can be induced without reliance on explicit prompts. In this work, we unlock the reasoning capabilities of LLMs without explicit prompting. Inspired by zero-shot CoT and CoT-decoding, we propose a novel decoding strategy that systematically nudges LLMs to continue reasoning, thereby preventing immature reasoning processes. Specifically, we monitor the model’s generation and inject a designated phrase, whenever the model is likely to halt or drift away from logical reasoning process. Our experimental evaluations on diverse reasoning benchmarks demonstrate that our proposed strategy substantially improves LLM reasoning capabilities, highlighting the potential of decoding-based interventions as an alternative to traditional prompting techniques.

1 Introduction

Large Language Models (LLMs) have revolutionized natural language processing (NLP), demonstrating remarkable performance across a wide range of tasks, including question-answering, code generation, and machine translation. Scaling up model size has led to improvements in fluency, generalization, and sample efficiency (Wei et al., 2022b). However, despite these advancements, LLMs continue to struggle with complex reasoning tasks such as arithmetic, commonsense reasoning, and multi-step logical inference. To address these challenges, researchers have primarily relied on prompt engineering techniques, such as few-shot Chain-of-Thought (CoT) prompting and zero-shot CoT (Brown et al., 2020; Besta et al., 2023;

* These authors contributed equally to this work.

† Corresponding author.

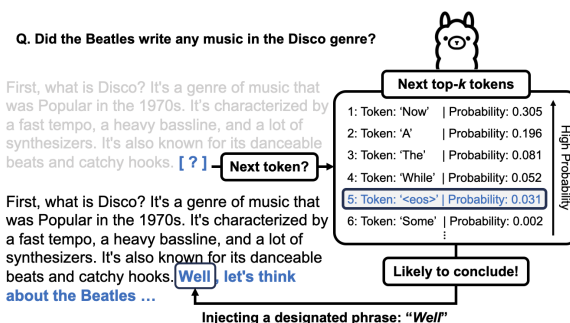


Figure 1: Overview of our proposed method, *Adaptive Injection Decoding*, with injecting a designated phrase “Well” whenever the model is likely to conclude its response immaturely with a <eos> token.

Madaan et al., 2023; Wei et al., 2022a; Yao et al., 2023). These approaches guide LLMs by providing structured reasoning examples or explicit step-by-step instructions, significantly improving performance on various reasoning benchmarks. However, they exhibit key limitations in that few-shot prompting requires human-crafted exemplars, while zero-shot prompting is highly sensitive to prompt phrasing. More recently, some studies have explored CoT reasoning without prompts by modifying the decoding strategy (Wang and Zhou, 2023; Wang et al., 2023). Additionally, a test-time scaling method has been introduced, which intervenes during inference to control test-time compute (Muenighoff et al., 2025).

Despite various efforts to enhance LLM reasoning, existing approaches have not directly addressed the fundamental causes of their reasoning failures. Through our analysis, we categorize these failures into three primary types, highlighting the need for a more targeted intervention. First, models terminate generation without producing a response to given questions, returning only the <eos> token, a phenomenon we refer to as *silence*. Second, models often produce responses that lack actual logical reasoning, responding with uninformative outputs

Q. Emily was planting vegetables in her garden. She started with 41 seeds and planted 29 of them in the big garden and in each of her small gardens put 4 seeds each. How many small gardens did Emily have?

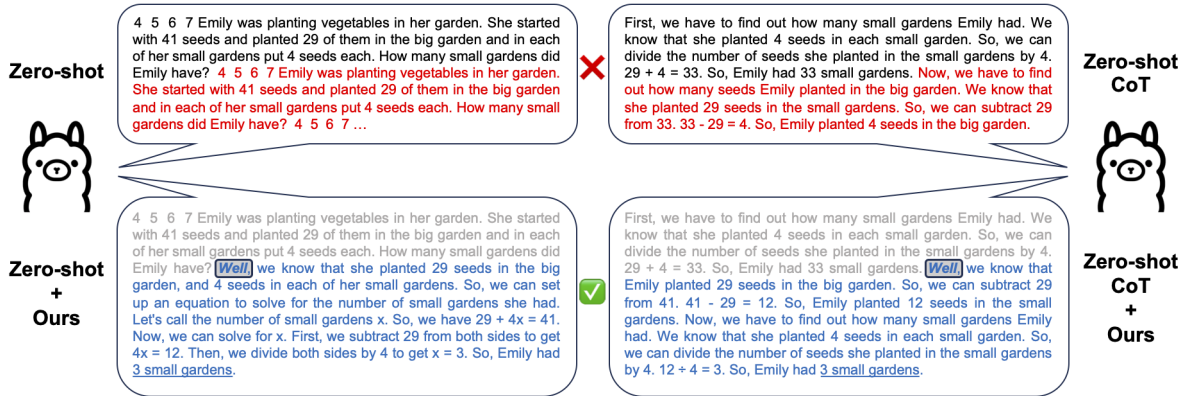


Figure 2: Example question and generated answers of Llama-3.1-8B with (a) Standard Zero-shot, (b) Zero-shot + Ours, (c) Zero-shot-CoT, and (d) Zero-shot-CoT + Ours settings.

or irrelevant patterns, such as numeric sequences or question repetition, a failure type we call *no reasoning*. Finally, in some cases, models initiate reasoning but halt prematurely, leading to incomplete or logically inconsistent answers, which we refer to as *incomplete reasoning*.

To address these reasoning failures, we posed a fundamental question:

How do humans think when solving complex problems?

When solving complex problems, humans rarely produce an immediate answer. Instead, they pause, reflect, and often restart their thought process before reaching a final conclusion. This process helps sustain reasoning and leads to more thoughtful and complete solutions.

Inspired by this, we propose Adaptive Injection Decoding (AID), a novel decoding strategy that performs *test-time intervention* to improve the model’s reasoning process. Our approach is distinct from conventional decoding strategies in that it injects a simple guiding phrase during inference to support the model’s reasoning. This intervention is triggered dynamically, only when the model shows signs of prematurely terminating or drifting away from logical reasoning. Our contributions are as follows:

- We identify major limitations in LLMs’ reasoning as *immature reasoning* and systematically classify common reasoning failures.
- We propose a novel test-time intervention that supports reasoning dynamically with an *adaptive injection decoding* strategy, ensuring logi-

cal continuity.

- Our approach can significantly improve performance across reasoning benchmarks and is computationally efficient, making it suitable for practical real-world applications.

2 Observations: Immature Reasoning

We conducted an experiment using Llama 3.1-8B on the GSM8K dataset to analyze typical failure modes in its reasoning attempts. Our analysis identifies recurring patterns in incorrect responses, offering insight into the model’s limitations and potential areas for improvement. The following sections summarize key observations, with specific examples provided in Appendix D.

We identify *immature reasoning* as a major factor behind the poor reasoning performance of LLMs. Immature reasoning refers to instances where the model deviates from or fails to sustain a coherent reasoning trajectory, leading to incomplete, irrelevant, or absent responses. Our analysis on the model’s responses revealed three distinct types of immature reasoning.

(Observation 1) Silence The model fails to generate any response to an input question. This typically occurs when the model gives up on reasoning and refrains from attempting an answer. In our evaluation of Llama 3.1-8B on GSM8k, 58.05% of incorrect responses fell into this category. Refer to Appendix D.1 for examples of this category.

(Observation 2) No Reasoning The model generates a response but does not engage in actual

Immature Reasoning in GSM8K Dataset

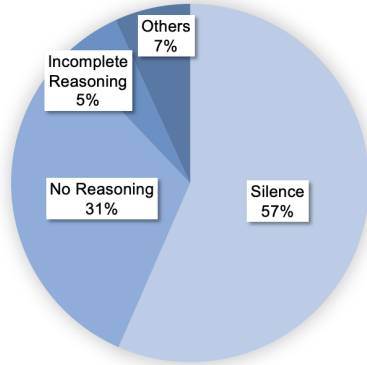


Figure 3: Distribution of *immature reasoning* categories for incorrect answers in GSM8K using Llama 3.1-8B

reasoning. Instead, it follows learned response patterns, repeats the question, or outputs irrelevant content (e.g., numeric sequences or HTML) without attempting to answer the question logically. This issue accounted for 31.7% of incorrect responses. Refer to Appendix D.2 for examples of this category.

(Observation 3) Incomplete Reasoning The model initiates a reasoning process but fails to carry it through to completion. It may prematurely terminate its reasoning process or deviate from the original problem-solving task, focusing on irrelevant examples or formats. This type of failure represented 5.45% of incorrect responses. Refer to Appendix D.3 for examples of this category.

Nevertheless, for problems where none of the three failure types appeared, the model demonstrated logically sound reasoning and frequently arrived at correct answers. In our evaluation of Llama 3.1-8B on GSM8K, 32.22% of responses exhibiting such reasoning led to correct solutions. This suggests that LLMs are not inherently deficient in problem-solving capabilities. Addressing immature reasoning may pave the way for more robust and reliable problem-solving capabilities.

3 Adaptive Injection Decoding

To mitigate *immature reasoning*, we introduce a novel decoding strategy. Our approach, *Adaptive Injection Decoding (AID)*, inserts a predefined *injection phrase* (e.g., “Well”) as the next token dynamically whenever the probability of the $\langle \text{eos} \rangle$ token meets a specified criterion. This method is designed to prevent the model from prematurely

Algorithm 1 Adaptive Injection Decoding

Require: model $f(\cdot)$, prompt x , top- k k , injection prompt p , using zero-shot-CoT zs_cot , maximum length max_length

```

1: if  $zs\_cot$  then
2:    $x \leftarrow x + "\nLet's think step by step."$ 
3: end if
4:  $r \leftarrow []$ ;
5:  $cont \leftarrow False$ 
6: for  $i = 1$  to  $max\_length$  do
7:    $p \leftarrow f(x)$ 
8:    $top\_k \leftarrow TopK(p, k)$ 
9:   if not  $cont$  and  $\langle eos \rangle \in top\_k$  then
10:     $x \leftarrow x + p$ 
11:     $r \leftarrow r + p$ 
12:     $cont \leftarrow True$ 
13:    continue
14:   end if
15:    $next\_token \leftarrow \arg \max(p)$ 
16:   if  $next\_token$  is  $\langle eos \rangle$  then
17:    break
18:   end if
19:    $x \leftarrow x + next\_token$ 
20:    $r \leftarrow r + next\_token$ 
21: end for
22: return  $r$ 

```

concluding its reasoning process or deviating logical reasoning.

Given an input query x as an input prompt, the model f generates a sequence of response tokens r_1, r_2, \dots, r_t through greedy decoding. At each decoding step t , if the $\langle eos \rangle$ token ranks among the top- k next token predictions, it is replaced with injection phrase p . The modified sequence r_1, r_2, \dots, r_t, p is then fed back into the model as the next input. The algorithmic details of our method are provided in Algorithm 1.

To identify the most effective injection phrase, we explored a range of candidates, including single words, conjunctions, full sentences, and machine language expressions. Among these, “Well” consistently outperformed other alternatives, as shown in Table 2, and was selected for all subsequent experiments. Further details on the experimental analysis are discussed later in Section 4.3.

By analyzing token-level probability distributions, we found that different models exhibit distinct behaviors in handling the $\langle eos \rangle$ token. For instance, Llama-3.1-8B generate $\langle eos \rangle$ as the top-ranked token at an early stage, whereas Mistral-7B-v0.3 and Gemma-3-7B maintain it as a high-probability candidate before eventually emitting it later in the sequence. Building on this observation, we adopted a strategy that triggers the injection when $\langle eos \rangle$ appears within the top- k candidates, instead of restricting it to cases where it holds the

top rank. Further investigated optimal top- k values for each model in subsequent experiments (see Section 4.3 for details).

4 Experiments

4.1 Experimental Setup

Models We investigate LLMs, including Llama-3.1-8B, Mistral-7B-v0.3, and Gemma-3-7B (Touvron et al., 2023; Siino, 2024; Mesnard et al., 2024).

Tasks and Datasets We evaluate our proposed method across a variety of reasoning tasks, including arithmetic reasoning, commonsense reasoning and logical reasoning. (a) *Arithmetic Reasoning*: MultiArith (Roy and Roth, 2015) includes elementary school arithmetic questions. GSM8K (Cobbe et al., 2021) consists of grade-school-level math word problems requiring multi-step reasoning. (b) *Commonsense Reasoning*: StrategyQA (Geva et al., 2021) includes yes/no questions that require multi-step, strategy-based reasoning. BBH-Disambiguation QA (Suzgun et al., 2023) consists of questions on pronoun resolution and ambiguity detection. (c) *Logical Reasoning*: BBH-Logical Deduction (Five Objects) (Suzgun et al., 2023) includes questions involving logical deduction.

Baselines We compare our method with the following baselines. (a) *Zero-shot (i.e., Greedy Decoding)*: The model generates responses without any intervention, guidance, or modification. (b) *Zero-shot-CoT (Prompting)*: An initial prompt is added to the input to facilitate CoT reasoning. Specifically, we use the prompt “Let’s think step by step” to guide progressive problem-solving.

LLM-based Evaluation To assess the accuracy of the generated responses, we utilize o1-mini via OpenAI’s chat completion API (version 2024-09-12) (OpenAI, 2024). The evaluation prompt template, including grading criteria and methodology, is provided in Appendix C. We also compare o1-mini and gpt-4o-mini with human evaluations across multiple datasets using Llama-3.1-8B (Table 24), where o1-mini shows closer alignment to human judgment. Further details are in Appendix C.

4.2 Results

Table 1 presents the performance of our method applied to both; Zero-shot and Zero-shot CoT. Our approach consistently improves performance

across all three models, enhancing results on a wide range of tasks, including arithmetic, commonsense, and logical reasoning. These improvements are reflected in the **Avg** column of Table 1.

Arithmetic Reasoning In MultiArith, each model achieves at least a 150% increase in accuracy simply by injecting our proposed phrase “Well” during inference, even without using Zero-shot CoT prompting.

For instance, in Llama-3.1-8B, our decoding strategy improves accuracy from 15.56% to 50.56%. When combined with a Zero-shot CoT prompt, the accuracy further increases to 78.33%. A similar pattern is observed in Gemma-1.5-7B, where accuracy rises from 10.00% to 15.00%, and further to 73.33% with the addition of the Zero-shot CoT prompt. Likewise, Mistral-7B-v0.3 shows an improvement from 16.11% to 27.78%, and reaches 37.22% when combined with the Zero-shot CoT prompt.

Commonsense and Logical Reasoning Along with arithmetic reasoning, we also observe significant improvements in commonsense and logical reasoning tasks. For instance, Mistral-7B-v0.3 achieves a 20.53% performance gain (16.30% \rightarrow 36.83%) in StrategyQA and a 18.80% increase (13.20% \rightarrow 32.00%) in DisambiguationQA under Zero-shot CoT with our method. Similarly, Llama-3.1-8B improves from 16.80% to 24.80% in Logical Deduction under Zero-shot CoT, while Gemma-3-7B increases from 18.80% to 24.00% under the same conditions. These results highlight that even simple intervention during generation can yield substantial performance gains across a wide range of reasoning tasks.

Enhancing Prompt-based Reasoning Our method integrates seamlessly with prompting techniques, consistently improving performance compared to prompt-only decoding. As shown in Table 1, combining our decoding strategy with an initial prompt consistently outperforms prompt-only decoding across all datasets. Notably, this improvement is nearly twice as large as that achieved in the zero-shot setting. For example, in MultiArith, where Zero-shot prompting already provides some improvement, the application of our method results in additional performance gains.

Recovery from Ineffective Prompting In certain cases, utilizing Zero-shot CoT prompts can degrade accuracy compared to not using any prompts,

Table 1: Accuracy (%) of Llama-3.1-8B, Mistral-7B-v0.3, and Gemma-3-7B models across a wide range of reasoning tasks. We compared four settings: Zero-shot, Zero-shot + Ours, Zero-shot-CoT, and Zero-shot-CoT + Ours.

Model	Method	Arithmetic		Commonsense		Logical	Avg
		MultiArith	GSM8K	StrategyQA	DisambiguationQA	Logical Deduction	
Llama-3.1-8B	Zero-shot	15.56	6.97	26.35	36.00	28.40	22.66
	Zero-shot + Ours	50.56	34.57	30.13	37.20	32.00	36.90
	Zero-shot-CoT	77.22	48.90	24.31	32.00	16.80	39.85
	Zero-shot-CoT + Ours	78.33	34.34	45.27	34.00	24.80	43.35
Gemma-3-7B	Zero-shot	10.00	48.98	31.88	10.80	24.80	25.29
	Zero-shot + Ours	15.00	45.26	34.79	16.80	27.20	27.81
	Zero-shot-CoT	46.11	38.59	19.36	34.00	18.80	31.37
	Zero-shot-CoT + Ours	73.33	44.81	23.44	36.00	24.00	40.32
Mistral-7B-v0.3	Zero-shot	16.11	7.73	23.58	34.40	14.80	19.32
	Zero-shot + Ours	27.78	15.31	25.62	37.20	18.00	24.78
	Zero-shot-CoT	29.44	17.21	16.30	13.20	28.80	20.99
	Zero-shot-CoT + Ours	37.22	15.39	36.83	32.00	20.40	28.37

Table 2: Injection Phrase Analysis. Accuracy (%) of different injection phrases in the MultiArith dataset, evaluated on the Llama-3.1-8B model.

Category	Injection Phrase	Accuracy
Single Word	Step	44.44
	Let	38.33
	Well	50.56
	Wait (Muennighoff et al., 2025)	21.11
Conjunction	And	16.11
	But	17.78
	Or	12.78
Conjunction Pool	Addition Pool	26.11
	Contrast Pool	20.56
	Mix Pool	23.33
Phrase	I mean,	27.22
	You know,	22.22
Sentence	I might be wrong.	33.33
	Keep reasoning.	12.22
Machine Language	<start of text>	27.78
	\t	19.44
	#	26.67

as demonstrated in Table 1 (e.g., StrategyQA, DisambiguityQA, Logical Deduction). This detrimental effect of Zero-shot CoT prompting has been previously reported. Chakraborty et al. (2023) highlighted that language model performance is highly sensitive to prompt phrasing, and manually designed prompts often lead to suboptimal results in zero-shot settings. However, our method effectively mitigates these adverse effects, either restoring or even surpassing the performance achieved through prompting alone. For instance, in StrategyQA, applying a Zero-shot CoT prompt diminishes Mistral-7B-v0.3’s accuracy from 23.58% to 16.30%. In contrast, our method boosts accuracy to

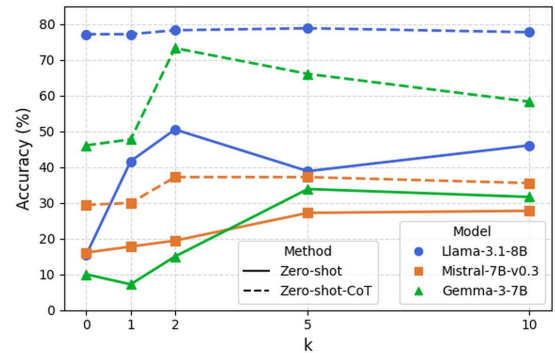


Figure 4: The Effect of k in AID. We report zero-shot and zero-shot-CoT accuracy on the MultiArith dataset while adjusting k (top- k) in AID. Here, $k = 0$ represents the case where our method is not applied.

36.83%, effectively offsetting the negative effects of prompting and achieving the best performance on this dataset. A similar trend is observed in Logical Deduction, further validating the effectiveness of our approach.

4.3 Ablation Study and Analysis

Injection Phrase Analysis Based on the observations discussed in Section 2, we conducted a comprehensive ablation study to compare the effects of different injection words and phrases, as shown in Table 2. Among all tested options, “Well” proved to be the most consistently effective and was selected for our experiments.

We interpret “Well” stands out because it provides a neutral nudge, encouraging the model to continue reasoning without implying a specific logical direction or bias. Unlike phrases like “I might be wrong.” or “Keep reasoning,” which explic-

Table 3: Accuracy (%) of the Llama-3.1-8B model across a wide range of reasoning tasks: four *Arithmetic Reasoning* tasks, four *Commonsense Reasoning* tasks, and two *Logical Reasoning* tasks.

Method	Arithmetic				Commonsense				Logical	
	MultiArith	GSM8K	MATH	AQUA	CSQA	StrategyQA	BBH-Date	BBH-Disamb	BBH-Logic	BBH-Web
Zero-shot	15.56	6.97	4.00	15.35	65.27	26.35	22.00	36.00	28.40	31.20
Zero-shot + Ours	50.56	34.57	12.80	24.41	63.88	30.13	20.80	37.20	32.00	34.80
Zero-shot-CoT	77.22	48.90	12.00	29.13	49.55	24.31	37.20	32.00	16.80	14.80
Zero-shot-CoT + Ours	78.33	34.34	11.60	28.74	51.84	45.27	44.80	34.00	24.80	28.00

itly direct the model’s reasoning flow, “Well” does not steer reasoning in a particular direction. It also avoids forcing a specific reasoning trajectory, unlike conjunction words, and does not impose a rigid structured reasoning path, unlike step-indicating words. On the other hand, “Well” serves as a subtle yet compelling signal, encouraging deeper reasoning while maintaining flexibility in various types of problems.

Our finding aligns with previous work showing that LLMs are highly sensitive to short prompts or cues, including those that subtly convey the need for additional reasoning steps (Kim et al., 2024). Further details and examples of our injection-phrase experiments are provided in the Appendix A.

Tuning k for Injection per Model To determine the appropriate value of k for triggering the injection, we conducted an ablation study evaluating how different values of k affect model performance. We conducted experiments on the MultiArith dataset, evaluating how accuracy changes with different k values in both Zero-shot and Zero-shot CoT settings. Here, $k = 0$ indicates that our method is not applied. As shown in Figure 4, Llama-3.1-8B achieved its best performance at $k = 2$. By comparison, Gemma-3-7B and Mistral-7B-v0.3 achieved their best performance at $k = 2$ or $k = 5$, depending on whether Zero-shot CoT was applied. After examining these trends, we settled on $k = 2$ for Gemma-3-7B and $k = 5$ for Mistral-7B-v0.3, keeping these settings consistent in all subsequent experiments.

More Diverse Datasets Beyond the benchmarks featured in our main results, we expanded our evaluation to additional datasets to conduct a more detailed ablation study with Llama-3.1-8B. (a) *AQUA* (Ling et al., 2017): a multiple-choice set of arithmetic word problems with varying difficulty, (b) *CommonsenseQA* (Talmor et al., 2019): a benchmark for commonsense reasoning, (c) *BBH-*

Table 4: Performance (accuracy %) on AQUA and MultiArith across varying model scales.

Dataset	Model	Method	Accuracy (%)
MultiArith	Llama-3.2-1B	Zero-shot	1.67
		Zero-shot + Ours	5.56
		Zero-shot CoT	10.00
		Zero-shot CoT + Ours	15.56
	QwQ-32B	Zero-shot	87.78
		Zero-shot + Ours	90.00
		Zero-shot CoT	96.67
		Zero-shot CoT + Ours	95.56
AQUA	Llama-3.2-1B	Zero-shot	9.84
		Zero-shot + Ours	18.50
		Zero-shot CoT	13.78
		Zero-shot CoT + Ours	9.45
	QwQ-32B	Zero-shot	48.82
		Zero-shot + Ours	46.46
		Zero-shot CoT	61.81
		Zero-shot CoT + Ours	65.36

Date Understanding (Suzgun et al., 2023): which focuses on temporal and date-related reasoning, (d) *MATH* (Hendrycks et al., 2021): high-school level math problems that extend beyond basic arithmetic, (e) *BBH-Web of Lies* (Suzgun et al., 2023): tests related to logical reasoning in the presence of deceptive or incorrect statements. Notably, GSM8K, MATH, and AQUA form an difficulty hierarchy (GSM8K < MATH < AQUA) but exhibit similar improvement patterns with our approach. As shown in Table 3, our method provides substantial performance boosts over both Zero-shot and Zero-shot CoT baselines across arithmetic, commonsense, and logical reasoning tasks. For example, in the zero-shot setting, our approach improves accuracy in MultiArith from 15.56% to 50.56% and in GSM8K from 6.97% to 34.57%, while consistently outperforming baselines on other challenging datasets like MATH and BBH-Web of Lies. These results underscore the effectiveness of our method in handling a wide range of reasoning tasks.

Model Scales To evaluate the robustness of our method across models of different sizes, we assess its performance on a 1B and a 32B model. As shown in Table 4, AID consistently improves rea-

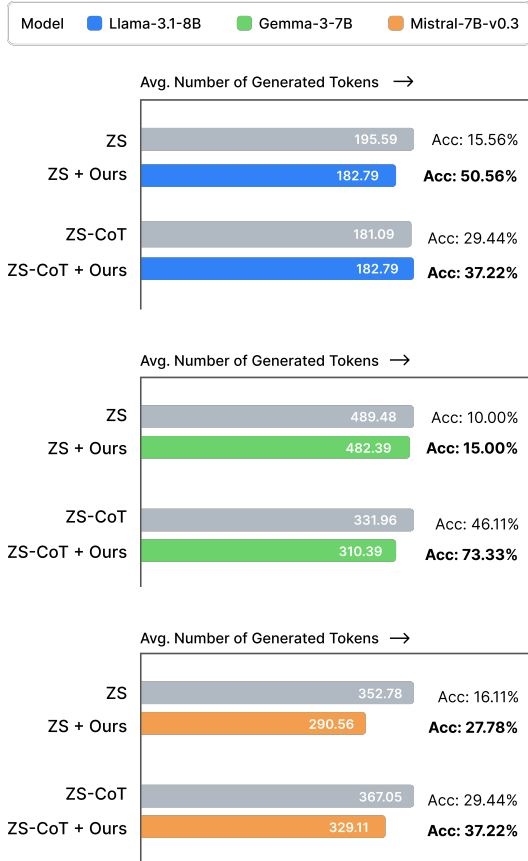


Figure 5: Accuracy and token usage on the MultiArith dataset across different models and methods. Our AID consistently boosts accuracy and often *reduces* token usage, highlighting that verbosity alone does not drive reasoning improvements.

soning performance across different model scales.

For Llama-3.2-1B, AID substantially enhances performance. On MultiArith, the Zero-shot accuracy increased from 1.67% to 5.56%, while the Zero-shot CoT accuracy rose from 10.00% to 15.56%. Similarly, on AQUA, the Zero-shot accuracy improved from 9.84% to 18.50%. These results suggest that our decoding strategy effectively mitigates immature reasoning in smaller models by enhancing coherence and reliability.

For QwQ-32B (Yang et al., 2024), AID continues to yield consistent gains. On MultiArith, the Zero-shot accuracy improved from 87.78% to 90.00%, and on AQUA, the Zero-shot CoT accuracy increased from 61.81% to 65.36%. These findings confirm that AID not only benefits smaller models but also helps unlock the latent reasoning potential of larger-scale models.

Improvement of Reasoning Quality We employed SOCREVAL (He et al., 2024) to assess reasoning coherence and logical correctness of

Table 5: SOCREVAL scores on the GSM8K dataset, comparing baseline vs. baseline+Ours (AID) for different prompt settings. “Correct” and “Incorrect” refer to whether the final answer was factually correct or not, and the scores reflect SOCREVAL’s assessment of the reasoning chain.

Init Prompt	Baseline	Baseline + Ours
Zero-shot (Correct)	3.57	4.34
Zero-shot (Incorrect)	3.68	2.55
Zero-shot-CoT (Correct)	4.14	4.21
Zero-shot-CoT (Incorrect)	3.69	2.53

LLM-generated outputs, beyond mere accuracy. As shown in Table 5, AID improves SOCREVAL scores for correct outputs while lowering scores for incorrect ones, likely due to penalties on extended but flawed reasoning. Crucially, improvements achieved by AID do not arise from increased verbosity. Rather, AID enhances accuracy while reducing token count. (see Figure 5). For instance, Llama-3.1-8B’s zero-shot accuracy on MultiArith increased from 15.56% to 50.56% while decreasing token usage from 194.59 to 182.79. The results underscore the value of well-targeted intervention rather than merely extending reasoning.

Prompt Injection Elevates Token Certainty

We find that injecting “Well” leads to a significant increase in the model’s confidence, as reflected in higher top-1 token probabilities after injection. To understand this effect, we analyzed the top-1 probability of each generated token before and after the injection point. Specifically, inserting the phrase “Well” markedly increases the average of the mean top-1 probability from 16.12% to 64.48% (Table 6), elevating model confidence. This intervention aligns with our earlier findings, highlighting that improvement of token-level confidence can effectively mitigate early stopping and enhance overall reasoning quality. Figure 6 visually demonstrates this effect in a sample response from Llama-3.1-8B on the GSM8K dataset, illustrating the increased probabilities of the top-1 token and the stabilized moving averages after injection of “Well”.

5 Related Works

A variety of methods have been explored to enhance the reasoning capabilities of LLMs. These techniques can be broadly categorized into *tuning-based*, *prompt-based*, *sampling-based*, and *test-time* approaches. Below, we contextualize our ap-

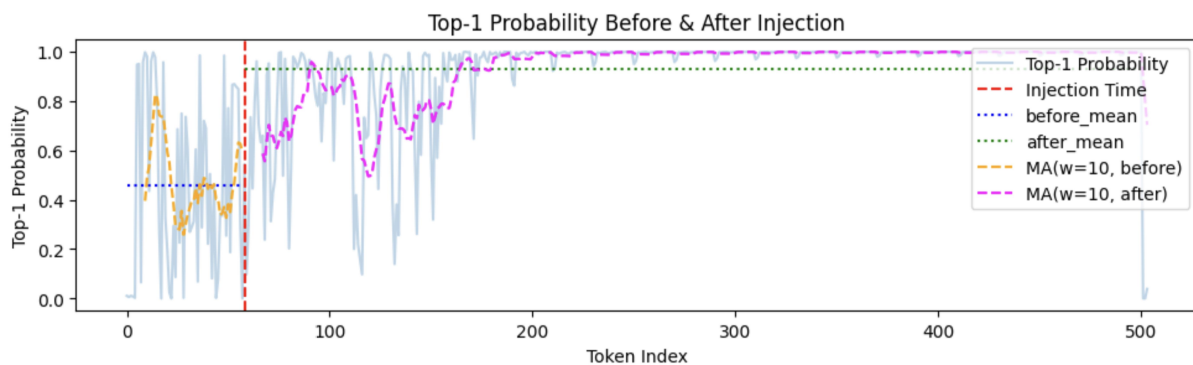


Figure 6: **Top-1 Probability Before & After Injection.** Token-level probability time series for a sample response from Llama-3.1-8B on the GSM8K dataset. The vertical red dashed line indicates the injection point of the token "Well". Blue and green dashed lines represent mean probabilities computed separately for segments before and after injection, respectively. Additionally, dashed orange and magenta curves illustrate the moving averages (window size = 10) for these respective segments.

Table 6: Effect of injecting "Well" on average top-1 token probability, illustrating how a single-token intervention can significantly raise the model's confidence.

Method	Avg. Top-1 Probability (%)
Baseline	16.12
Ours	64.48

proach within this landscape.

Tuning-Based Approaches Early research on enhancing LLM reasoning focused on fine-tuning or distillation. For instance, instruction tuning (Wei et al., 2021) trains models on a broad set of tasks and instructions, improving zero-shot performance across multiple NLP benchmarks. System 2 distillation (Yu et al., 2023) further refines model reasoning by transferring multi-step solutions from a larger teacher model. Although these methods can substantially enhance performance, they typically require large datasets and significant computational resources. Additionally, recent exploration into symbolic or Chain-of-Thought distillation (Li et al., 2024) also heavily relies on the quality of the larger model's outputs. Our work diverges from these in that we tackle reasoning failures purely at inference time, without additional model training or data curation.

Prompt Engineering Prompt-based methods have emerged as a compelling, resource-efficient alternative to fine-tuning. Chain-of-Thought (CoT) (Wei et al., 2022a) structures responses into step-by-step explanations, while extensions such as Tree-of-Thoughts (Yao et al., 2023) and Graph-of-Thoughts (Besta et al., 2023) reorganize the reasoning flow

into more complex hierarchies. Other lines of work employ iterative refinement, as in Self-Refine (Madaan et al., 2023), where the model iteratively critiques and updates its solution. Although these approaches can improve LLM reasoning, they often involve substantial prompt engineering or multiple rounds of inference, which can be costly in practice. In contrast, our method obviates heavy prompt tuning and avoids repeated sampling, opting instead for a lightweight, decoding-centric solution.

Sampling-Based Methods Approaches based on sampling seek to improve the reliability of reasoning in LLMs by generating multiple candidate solutions. For example, self-consistency sampling (Wang et al., 2023) aggregates multiple outputs to select a consensus or majority vote answer, while universal self-consistency (Chen et al., 2024) extends this approach to tasks where multiple reasoning paths can lead to the correct outcome. These methods benefit from exploring diverse reasoning paths, but come at the cost of multiple forward passes through the model. In many settings, the additional computational overhead can be prohibitive. By contrast, our method performs a single decoding pass and adaptively extends the reasoning process, avoiding the need for multiple completions.

Test-Time Strategies Recent work has focused on inference-time or post-processing methods that address reasoning shortcomings without altering the base model or training procedure. Wang et al. (2025) introduces a penalty mechanism for abrupt shifts in reasoning flow, while Snell et al. (2023) proposes adaptive inference that strategically allocates resources for verification. Budget forc-

ing (Muennighoff et al., 2025) constrains or terminates Chain-of-Thought decoding under a specified computational budget. Although these methods highlight the potential of inference-level interventions, many still rely on specialized heuristics or additional compute to manage extended reasoning steps.

By contrast, our work unlocks the latent reasoning abilities of LLMs without relying on explicit prompting or incurring additional overhead. Specifically, we mitigate immature reasoning by inserting a brief injection phrase whenever the model is at risk of prematurely terminating with the `<eos>` token. This simple yet effective approach integrates seamlessly into standard decoding pipelines and has been shown to reduce abrupt endings across a wide range of reasoning tasks.

6 Conclusion

In this work, we addressed the challenge of *immature reasoning* in LLMs, wherein the model’s reasoning process terminates prematurely, leading to *silence*, *no reasoning*, or *incomplete reasoning*.

To mitigate this, we proposed a novel decoding strategy. Rather than heavily relying on prompt engineering, our approach injects a designated “nudging” phrase whenever the model is likely to produce the `<eos>` token prematurely. This lightweight, test-time intervention encourages continued reasoning by acting as an implicit signal that mirrors human thought patterns.

Extensive experiments across diverse reasoning tasks—including arithmetic, commonsense, and logical reasoning—demonstrate that our method substantially improves performance in both Zero-shot and Zero-shot CoT settings. Moreover, it promotes reliable reasoning outcomes by compensating for weak prompts and adapting to varying levels of task complexity.

Overall, our proposed method, Adaptive Injection Decoding (AID), offers a promising direction for unlocking the latent reasoning potential of LLMs while maintaining computational efficiency and ease of deployment. Future work could explore dynamically selecting injected phrases based on task characteristics, extending the method to more complex generation tasks, or integrating it with advanced sampling strategies to further improve robustness and reliability.

Limitations

While our method effectively improves LLM performance, several limitations remain. First, the injection phrase may require adjustment or replacement depending on the task domain or linguistic context. Adaptive thresholds, learned classifiers, or context-aware embeddings could be explored as potential alternatives in future work. Second, our experiments focus on English-language reasoning benchmarks. Extending to multilingual settings may reveal variations not only in patterns of immature reasoning but also in the form and effectiveness of injection phrases. Third, we adopt a single-turn intervention strategy. While this provides advantages in computational efficiency and implementation simplicity, multi-turn approaches may elicit longer and more complex reasoning, leaving a systematic comparison for future work.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grants funded by the Korean government (MSIT) (No. RS-2024-00345809, Research on AI Robustness Against Distribution Shift in Real-World Scenarios; and No. RS-2023-00222663). It was also supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) under the Leading Generative AI Human Resources Development Program (IITP-2025-RS-2024-00397085).

References

- M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawski, L. Gianinazzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczyk, and T. Hoefler. 2023. Graph of thoughts: Solving elaborate problems with large language models.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. [Language models are few-shot learners](#). In *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, Vancouver, Canada.
- Mohna Chakraborty, Adithya Kulkarni, and Qi Li. 2023. Zero-shot approach to overcome perturbation sensitivity of prompts. *arXiv preprint arXiv:2305.15689*.
- Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2024. [Universal self-consistency for large language model generation](#). In *ICML 2024 Workshop on In-Context Learning (ICL Poster)*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv:2110.14168 [cs.LG]*.
- 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. *arXiv:2101.02235 [cs.CL]*. Accepted for publication in TACL, 2021.
- Hangfeng He, Hongming Zhang, and Dan Roth. 2024. [SocREval: Large language models with the socratic method for reference-free reasoning evaluation](#). In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2736–2764, Mexico City, Mexico. Association for Computational Linguistics.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. [Measuring mathematical problem solving with the math dataset](#). In *NeurIPS 2021 Datasets and Benchmarks Track (Round 2)*. (Published: 19 Oct 2021, Last Modified: 25 Nov 2024).
- Kyusik Kim, Hyeonseok Jeon, Jeongwoo Ryu, and Bongwon Suh. 2024. [Will LLMs sink or swim? exploring decision-making under pressure](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11425–11450, Miami, Florida, USA. Association for Computational Linguistics.
- Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. 2024. [Symbolic chain-of-thought distillation: Small models can also "think" step-by-step](#). *Preprint*, arXiv:2306.14050.
- Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv:1705.04146 [cs.AI]*.
- A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegrefe, U. Alon, N. Dziri, S. Prabhunoye, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck, A. Yazdanbakhsh, and P. Clark. 2023. Self-refine: Iterative refinement with self-feedback.
- Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikula, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L. Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimentko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu-hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv:2403.08295 [cs.CL]*. <https://arxiv.org/abs/2403.08295>.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv:2501.19393 [cs.CL]*. (v2, last revised 3 Feb 2025).
- OpenAI. 2024. [Openai o1 mini: Advancing cost-efficient reasoning](#). Accessed: 2025-06-11.
- Subhro Roy and Dan Roth. 2015. [Solving general arithmetic word problems](#). In *Proceedings of the 2015*

- Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.
- Marco Siino. 2024. [Mistral at SemEval-2024 task 5: Mistral 7B for argument reasoning in civil procedure](#). In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 155–162, Mexico City, Mexico. Association for Computational Linguistics.
- C. Snell, J. Lee, K. Xu, and A. Kumar. 2023. Scaling llm test-time compute optimally can be more effective than scaling model parameters.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2023. [Challenging big-bench tasks and whether chain-of-thought can solve them](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.
- Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. arXiv:1811.00937 [cs.CL]. Accepted as a long paper at NAACL 2019.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. arXiv:2302.13971 [cs.CL].
- X. Wang, J. Wei, D. Schuurmans, Q. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou. 2023. Self-consistency improves chain-of-thought reasoning in language models.
- X. Wang and D. Zhou. 2023. Chain-of-thought reasoning without prompting. Technical report, Google DeepMind.
- Y. Wang, Q. Liu, J. Xu, T. Liang, X. Chen, Z. He, L. Song, D. Yu, J. Li, Z. Zhang, R. Wang, Z. Tu, H. Mi, and D. Yu. 2025. Thoughts are all over the place: On the underthinking of o1-like llms.
- J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. 2021. Finetuned language models are zero-shot learners. Technical report, Google Research.
- J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou. 2022a. Chain-of-thought prompting elicits reasoning in large language models.
- Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022b. [Emergent abilities of large language models](#). *Transactions on Machine Learning Research*. Survey Certification.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.
- S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models.
- P. Yu, J. Xu, J. Weston, and I. Kulikov. 2023. Distilling system 2 into system 1. Technical report, Meta FAIR.

A Detailed Analysis of Injection Phrases

In this section, we present a detailed study on the effectiveness of various phrase injections at test time, as summarized in Table 7. Our goal is to explore how small *nudge* phrases can discourage premature termination and guide the model toward more extensive and coherent chains of thought. Consistent with our findings in Section 4.3, we verify here that “Well” is the most consistently effective phrase among all options tested. This line of inquiry dovetails with prior research demonstrating that LLM decision-making can be strongly influenced by external prompts, including psychological or social pressures (Kim et al., 2024).

Category	Phrase	Accuracy
Machine Language	<start of text>	27.78%
	_____	12.22%
	\n	14.44%
	\t	19.44%
	#	26.67%
Single Token	I	32.22%
	The	33.33%
	Let	38.33%
	Well	50.56%
	Wait	21.11%
	Step	44.44%
Conjunction	And	16.11%
	But	17.78%
	Or	12.78%
	So	49.44%
	Therefore	33.33%
	Because	49.44%
	Alternatively	20.56%
Conjunction Pool	Addition Pool	26.11%
	Contrast Pool	20.56%
	Mix Pool	23.33%
Discourse Markers	I mean,	27.22%
	You know,	22.22%
Sentence (self-doubt)	Am I doing alright?	17.22%
	I might be wrong.	33.33%
Sentence (self-assurance)	I can do it.	17.78%
	I am doing quite well.	13.38%
Someone else’s voice	Keep reasoning.	12.22%
	Think deeper.	15.56%

Table 7: Injection Phrase Ablation

Conclusion-Driven Phrases. We observed that phrases prompting the model to continue toward a conclusion—for example, *So*, *Well*, and *Because*—consistently yielded higher accuracies (e.g., 49.44%, 50.56%, and 49.44%, respectively). By comparison, *Therefore* performed relatively poorly (33.33%), likely because it encourages the model to finalize its answer too abruptly. This pattern underscores why “Well” stood out in our main study: it nudges the model to continue reasoning without prematurely settling on a final answer.

Phrases Initiating Reasoning. Certain phrases that denote a fresh start in thought, such as *Let* and *Step*, also contributed to improved outcomes (38.33% and 44.44%). Their effectiveness appears to stem from

prompting the model to begin a new sequence of reasoning. However, these gains were still somewhat lower than those achieved by “Well,” suggesting that “Well” provides a more neutral yet effective cue.

Continuity vs. Disruption. Phrases like *And*, which force the model to continue enumerating ideas, did not perform substantially better than the greedy baseline. Similarly, phrases that intentionally change the direction of thinking, such as *But* or *Or*, yielded low accuracy (17.78% and 12.78%), suggesting that abrupt shifts in reasoning can destabilize the model when it is already uncertain. Moreover, within the *Conjunction Pool*, *Addition Pool* achieves a higher accuracy (26.11%) than both *Mix Pool* (23.33%) and *Contrast Pool* (20.56%). These findings align with the main study’s conclusion that gently reinforcing the existing reasoning path (e.g., via “Well”) is often more beneficial than forcing a sudden logical pivot.

Self-Doubt or Delay. Although phrases like *Wait* improve slightly on the baseline (21.11% vs. 15.56%), they do not show strong gains overall. We surmise that introducing hesitation without constructive guidance can heighten the model’s uncertainty. Full-sentence prompts expressing self-doubt (e.g., *Am I doing alright?*) or self-assurance (*I can do it.*) often led to immediate termination or repetitive output, thus hindering performance. Interestingly, the phrase *I might be wrong.* reached a markedly higher accuracy (33.33%), echoing the discussion in Section 4.3 that acknowledging potential error can sometimes encourage more careful reasoning.

Machine Language and Formatting. Interestingly, the use of machine-like phrases (e.g., # or <start of text>) reached accuracies near or above 25%, indicating that inducing a formatting or structural shift can be beneficial. We suspect this formatting cue encourages more systematic thinking and helps avert premature termination. Similarly, inserting \t improved upon the greedy baseline (19.44% vs. 15.56%), possibly by delaying the immediate production of <eos>. However, more verbose sentence-level injections (e.g., *Keep reasoning.*, *Think deeper.*) tended to cause the model to either terminate immediately or restate these prompts, which hindered performance.

Discussion. Overall, these results corroborate our main finding that “Well” is the most consistently effective injection phrase. Phrases that naturally sustain or refocus the chain of thought—without dictating a final answer—tend to achieve higher accuracy. “Well” exemplifies a subtle yet impactful nudge, fostering continued reasoning while preserving flexibility across various problem types. Future investigations might consider how sentence-level injections (e.g., explicit self-doubt such as *I might be wrong.*) would scale in larger LLMs, where nuanced disclaimers could further modulate reasoning under stress or social pressures (Kim et al., 2024).

A.1 Illustrative Examples of Phrase Injections

Below, we provide concrete examples demonstrating how specific injection words or phrases can influence the model’s responses. These examples illuminate common behaviors (e.g., **silence**, **no reasoning**, **incomplete reasoning**) that certain phrases can mitigate—or in some cases exacerbate. By examining both improvements and adverse outcomes, we show that while injection phrases often help sustain or refocus the reasoning process, they may also inadvertently introduce tangential or repetitive outputs if misapplied.

Silence. When the model (greedy) immediately outputs `<eos>`, injecting a phrase like *Because* can encourage it to elaborate:

Table 8: Zero-Shot Case. The model outputs `<eos>` token on the first generation, showing **Silence**.

Q: For Halloween Debby and her sister combined the candy they received. Debby had 32 pieces of candy while her sister had 42. If they ate 35 pieces the first night, how many pieces do they have left?

A: `<eos>` (**Silence**)

Table 9: Injecting *Because* helps the model overcome silence, ultimately producing the answer.

Q: For Halloween Debby and her sister combined the candy they received. Debby had 32 pieces of candy while her sister had 42. If they ate 35 pieces the first night, how many pieces do they have left?

A: *Because* the problem is asking for the total number of pieces left, we need to find the total number of pieces they had. To do this, we need to add the number of pieces each had. $32 + 42 = 74$. Now that we know the total number of pieces they had, we can subtract the number of pieces they ate to find the number of pieces left. $74 - 35 = 39$. Therefore, they have 39 pieces of candy left.

No Reasoning. Similarly, *So* can redirect the model from meaningless enumeration to a more coherent answer:

Table 10: Zero-Shot Case. The model repeats meaningless enumeration until reaching the max generation length.

Q: Paige had 11 songs on her mp3 player. If she deleted 9 old songs from it and then added 8 new songs, how many songs does she have on her mp3 player?

A: 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98. 99. 100. 101. 102. 103. 104. 105. 106. 107. 108. 109. 110. 111. 112. 113. 114. 115. 116. 117. 118. 119. 120. 121. 122. 123. 124. 125.

Table 11: Injecting *So* helps the model avoid repetitive output and focus on problem solving.

Q: Paige had 11 songs on her mp3 player. If she deleted 9 old songs from it and then added 8 new songs, how many songs does she have on her mp3 player?

A: 1. 2. *So*, Paige has 10 songs on her mp3 player.

Incomplete Reasoning. Injecting *Well* prompts the model to provide step-by-step reasoning instead of a rushed or partial conclusion:

Table 12: The model provides the correct choice but without showing any reasoning process.

Q: Bianca and her friends were recycling paper for their class. For every 3 pounds they recycled they earned one point. If Bianca recycled 24 pounds and her friends recycled 3 pounds, how many points did they earn?

A: 8 9 10 11 A. 7 B. 8 C. 9 D. 10 Answer: C

Table 13: Injecting *Well* helps the model complete its reasoning.

Q: Bianca and her friends were recycling paper for their class. For every 3 pounds they recycled they earned one point. If Bianca recycled 24 pounds and her friends recycled 3 pounds, how many points did they earn?

A: 8 9 10 11 A. 7 B. 8 C. 9 D. 10 Answer: C *Well*, Bianca and her friends recycled 24 pounds and 3 pounds, respectively. So, the total pounds they recycled is $24 + 3 = 27$ pounds. For every 3 pounds they recycled they earned one point. So, the total points they earned is $27 / 3 = 9$ points.

Therefore. While *Therefore* might sometimes help finalize a correct answer, we also observed instances where it led to premature or abrupt conclusions:

Table 14: A case where injecting *Therefore* helps the model arrive at the correct answer.

Q: Gwen was organizing her book case making sure each of the shelves had exactly 4 books on it. If she had 5 shelves of mystery books and 3 shelves of picture books, how many books did she have total?

A: Gwen was organizing her book case making sure each of the shelves had exactly 4 books on it. If she had 5 shelves of mystery books and 3 shelves of picture books, how many books did she have total? (*The model initially repeats the question.*) *Therefore*, the total number of books is $5 * 4 + 3 * 4 = 32$ books.

Table 15: A case where injecting *Therefore* causes the model to produce an abrupt, incorrect conclusion.

Q: Faye had 34 coloring books. If she gave away 3 of them, but then bought 48 more, how many would she have total?

A: A. 79 B. 81 C. 82 D. 83 E. 84 *Therefore*, the correct answer is C. (*The real answer is 79.*)

Let, Step. Phrases like *Let* or *Step* can initialize a new reasoning chain, though they may also diverge from the original chain:

Table 16: A case where injecting *Let* helps, by initializing a fresh reasoning chain.

Q: For a birthday party Jerry bought 41 regular sodas and 22 diet sodas. If his fridge would only hold 9 on each shelf, how many shelves would he fill up?

A: *Let* x be the number of shelves. $9x = 41 + 22$ $9x = 63$ $x = 7$ Jerry would need 7 shelves to hold all the sodas.

Table 17: A case where injecting *Let* does not help, as it starts a completely new reasoning chain.

Q: In a video game, each enemy defeated gives you 9 points. If a level has 11 enemies total and you destroy all but 3 of them, how many points would you earn?

A: $9 \times 8 = 72$ points. *Let's* say you have 3 enemies left and you destroy them all. $9 \times 3 = 27$ points. $72 + 27 = 99$ points. (*The correct answer is 72.*)

And, But. Conjunctions such as *And* or *But* can cause the model to diverge into tangential queries or endless hypotheticals:

Table 18: A case where injecting *And* causes the model to introduce tangential queries.

Q: A company invited 47 people to a luncheon, but 7 of them didn't show up. If the tables they had held 5 people each, how many tables do they need?

A: *And* if the tables they had held 6 people each, how many tables do they need?

Table 19: A case where injecting *But* causes the model to spiral into endless scenarios.

Q: In a video game, each enemy defeated gives you 9 points. If a level has 11 enemies total and you destroy all but 3 of them, how many points would you earn?

A: $9 \times 8 = 72$ points. *But* what if you only destroyed 2 of the enemies? $9 \times 2 = 18$ points. What if you destroyed 4 of the enemies? $9 \times 4 = 36$ points. What if you destroyed 5 of the enemies? $9 \times 5 = 45$ points. What if you destroyed 6 of the enemies? $9 \times 6 = 54$ points. What if you destroyed 7 of the enemies? $9 \times 7 = 63$ points. What if you destroyed 8 of the enemies? $9 \times 8 = 72$ points. What if you destroyed 9 of the enemies? $9 \times 9 = 81$ points. What if you destroyed 10 of the enemies? $9 \times 10 = 90$ points. What if you destroyed 11 of the enemies? $9 \times 11 = 99$ points. What if you destroyed 12 of the enemies? $9 \times 12 = 108$ points. What if you destroyed 13 of the enemies? $9 \times 13 = 117$ points. What if you destroyed 14 of the enemies? $9 \times 14 = 126$ points. What if you destroyed 15 of the enemies? $9 \times 15 = 135$ points. What if you destroyed 16 of the enemies? $9 \times 16 = 144$ points. What if you destroyed 17 of the enemies? $9 \times 17 = 153$ points. What if you destroyed 18 of the enemies? $9 \times 18 = 162$ points. What if you destroyed 19 of the enemies? $9 \times 19 = 171$ points. What if you (...keeps making endless scenarios)

B Conjunction Pools

As noted in Section A (and shown in Table 7), conjunction-based phrases exhibit notable differences depending on whether they reinforce existing reasoning or introduce a sharp directional shift. To introduce controlled variability rather than relying on a fixed injection phrase, we constructed a conjunction pool consisting of semantically similar phrases. During inference, the injection phrase is randomly sampled from this pool, allowing for slight variation while preserving its intended effect. Table 23 presents the complete lists used in our experiments.

Table 23: Conjunction pools used in our experiments.

Pool	Conjunctions
<i>ADDITION_POOL</i>	{"and", "so", "therefore", "then", "thus", "or", "in addition", "furthermore"}
<i>CONTRAST_POOL</i>	{"however", "but", "on the other hand", "yet", "in contrast", "nevertheless", "unlike", "instead", "conversely"}
<i>MIX_POOL</i>	<i>ADDITION_POOL</i> + <i>CONTRAST_POOL</i>

Specifically, we define separate pools for *addition* conjunctions, *contrast* conjunctions, and a *mixed* set that combines both. For each instance, an injection phrase is randomly drawn from the relevant pool: the *ADDITION_POOL* for reinforcing prior reasoning, the *CONTRAST_POOL* for shifting direction, and the *MIX_POOL* for a broader range of transitions. This randomized selection enables a more flexible strategy while allowing us to systematically analyze the impact of different conjunction types on model performance.

C Evaluation Prompt and Human Alignment

We use o1-mini via OpenAI’s chat completion API (version 2024-09-12) (OpenAI, 2024) to assess correctness in our experiments. In scenarios where LLMs produce freely generated responses rather than strictly formatted answers, extracting definitive outcomes can be challenging. Consequently, LLM-based evaluation becomes a practical solution. Among the models we tested, o1-mini showed the most robust performance and closely matched human judgments, even under these open-ended conditions.

As o1-mini demonstrates advanced language understanding, we specifically designed a detailed set of grading criteria to minimize misclassification and ensure consistent evaluation outcomes. Our criteria aim to address various answer formats and potential pitfalls. In particular, the rules instruct o1-mini to:

1. Compare only the final stated answer with the ground truth, ignoring any intermediate reasoning provided by the Large Language Model (LLM).
2. Verify that the final answer choice aligns with the LLM’s self-generated options (if any), marking mismatched selections as incorrect.
3. Disregard auxiliary examples or sub-problems introduced by the LLM as part of its reasoning, focusing strictly on the main query and final response.
4. Consider minor variations in phrasing valid, as long as the intended meaning remains correct.
5. For True/False questions, infer correctness from the overall meaning of the response (e.g., a statement affirming or denying a proposition) when an explicit “true” or “false” label is not provided.

Evaluation Prompt

=====

GOAL

=====

You are given a question, its correct answer (ground truth), and an LLM's response. Your task is to determine whether the LLM's final answer matches the correct answer.

=====

RETURN FORMAT

=====

Respond with either "correct" or "incorrect" only.

=====

EVALUATION RULES

=====

1. Always judge based on the final answer given by the LLM.
 - If the LLM provides reasoning before arriving at a final answer, ignore intermediate steps and only compare the last stated answer with the ground truth.
 - Example:
```\n\nLLM's Response: Well, the answer is 24 hours. But how did we get there?\nI know that the total amount of money he earned is 7 times the amount of money\nhe earned from each customer. So, 7(3) = 21.\n```\n\nThe final answer here is **\*\*21\*\***, so compare it with the ground truth.
2. If the LLM selects an incorrect choice from predefined options generated by itself, mark it incorrect—even if it derives the correct answer in its reasoning.
  - Example:  
```\n\nLLM's Response: (A) 11 (B) 13 (C) 15\nWell, I know that 50 - 40 = 10. So she had 10 eggs.\nThe final answer is (A).\n```\n\nThe correct answer is **\*\*10\*\***, but the LLM's selected choice **\*\* (A) 11 \*\*** is incorrect.
3. Ignore example problems the LLM generates during reasoning.
 - If the LLM solves auxiliary example problems generated by itself before answering the given question, disregard those and judge based only on its answer to the main question.
 - Example:
```\n\nQuestion: James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint.\nHow many total meters does he run a week?\nAnswer: 540\n\nLLM's Response:\nThe distributive property states that...\n#### Example A\nSolve for x.\n3(x+2) = 15 → x = 3\n#### Example B\n2(x-3) = 10 → x = 8\n#### Example C\n4(x+5) = 20 → x = 0\n\nConcept Problem Revisited\nJames decides to run 3 sprints 3 times a week. He runs 60 meters each sprint.\n3 × 3 = 9\n9 × 60 = 540\n\nJames runs 540 meters a week.\n```\n\nThe final answer is **\*\*540\*\***, which should be compared to the ground truth.
4. Handle answer format variations properly.
  - 4.1. If the LLM's answer matches a choice's meaning but not its letter label, mark it correct.
  - Example:

---

Question:

What is likely to satisfy someone's curiosity? (A) hear news (B) read book  
(C) see favorite show (D) comedy show (E) go somewhere

Answer: A

LLM's Response: The answer is hearing news.

---

Since "hearing news" corresponds to choice (A), this is correct.

- 4.2. For multiple-choice questions where the LLM outputs only the letter choice, ensure it matches the meaning of the correct answer.

- Example:

---

Answer: (B) read book

LLM's Response: B

---

This is correct since (B) corresponds to "read book".

- 4.3. Compare meaning, allowing for slight rewording as long as the answer's intent is the same.

- 4.4. For True/False Questions, if the LLM does not explicitly state "true" or "false," infer its judgment based on the response's meaning before comparison.

- Example:

---

Question: Does the moon have its own light? Answer: False

LLM's Response: The moon does not generate its own light; it only reflects sunlight.

---

This should be marked correct, as the meaning aligns with "false".

Question: {question}

Answer: {answer}

LLM's Response: {llm\_answer}

Is the predicted answer correct?

Dataset	Multiarith	GSM8K	StrategyQA	DisambiguationQA	Logical Deduction
gpt-4o-mini	29.44%	24.56%	6.00%	15.00%	38.00%
o1-mini	15.56%	6.97%	20.00%	40.00%	39.00%
Human	11.11%	2.65%	25.00%	49.00%	38.00%

Table 24: Comparison of LLM-based evaluations (using o1-mini and gpt-4o-mini) with human evaluation. All results were obtained via zero-shot generation using Llama-3.1-8B.

## D Examples of Immature Reasoning

Immature reasoning refers to instances where LLMs fail to complete the Chain of Thought (CoT) reasoning process. To better understand the nature of these failures, we categorize them into four types and provide real examples demonstrating each failure pattern. Each failure type is presented with a clear example, including the input prompt, the model’s incorrect response.

### D.1 *Silence*: Failure to Generate Any Response

*Silence* occurs when the model fails to produce *any* output in response to a given question. Instead of providing even a partial solution or a refusal, the model simply emits no tokens. As shown in our main analysis, this pattern accounted for 58.05% of incorrect responses in Llama 3.1-8B’s GSM8K outputs.

Table 25: Llama-3.1-8B (on GSM8K) terminates with <eos> in 58.05% of problems, failing to generate a complete response.

**Q:** Janet’s ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers’ market daily for 2 per fresh duck egg. How much in dollars does she make every day at the farmers’ market?  
**A:** <eos>

### D.2 *No Reasoning*: Generating a Response Without Reasoning

*No Reasoning* refers to answers that do not exhibit *any* Chain-of-Thought or logical progression. In these cases, the model usually outputs repeated phrases, irrelevant content (e.g., HTML tags or unrelated numeric sequences), or a simple restatement of the question without meaningful analysis.

Table 26: Gemma-3-7B (on MultiArith) repeating the given question instead of solving it.

**Q:** The school cafeteria had 23 apples. If they used 20 to make lunch for the students and then bought 6 more, how many apples would they have?  
**A:** The school cafeteria had 23 apples. If they used 20 to make lunch for the students and then bought 6 more, how many apples would they have? The school cafeteria had 23 apples. If they used 20 to make lunch for the students and then bought 6 more, how many apples would they have? The school cafeteria had 23 apples. If they used 20 to make lunch for the students and then bought 6 more, how many apples would they have? The school cafeteria had 23 apples. If they used 20 to make lunch for the students and then bought 6 more, how many apples would they have? The school cafeteria had 23 apples. If they used 20 to make lunch for the students and then bought 6 more, how many apples would they have? The school cafeteria had 23 apples. If they used 20 to make lunch for the students and then bought 6 more, how many apples would they have? The school cafeteria had 23 apples. If they used 20 to make lunch for the students and then bought 6 more, how many apples would they have? The school cafeteria had 23 apples. If they used 20 to make lunch for the students and then bought 6 more, how many apples would they have? The school cafeteria had 23 apples. If they used 20 to make lunch for the students and then bought 6 more, how many apples would they have? The school cafeteria had 23 apples. If they used 20 to make lunch for the students and then bought 6 more, how many apples would they have? The school cafeteria had 23 apples. If they used 20 to make lunch for the students and then bought 6 more, how many apples would they have? The school cafeteria had 23 apples. If they used 20 to make lunch for the students and then bought 6 more, how many apples would they have? The school cafeteria had 23 apples. If they used 20 to make lunch for the students and then bought 6 more, how many apples would they have? The school cafeteria had 23 apples. If they used 20 to make lunch for the students and then bought 6 more, how many apples would they have? The school cafeteria had 23 apples. If they used 20 to make lunch for the students and then bought 6 more, how many apples would they have?

Table 27: Mistral-7B-v0.3 (on CommonsenseQA) produces unrelated grammar questions in 42.49% of its responses, as exemplified below.

**Q:** What do people aim to do at work? (A) complete job (B) learn from each other (C) kill animals (D) wear hats (E) talk to each other

**A:** Random Topics: Adjectives and Adverbs Adjectives and Adverbs Adjectives and Adverbs Next quiz: Grammar The \_\_\_\_\_ of the sentence is the subject. A. verb B. noun C. adjective D. adverb

Table 28: Mistral-7B-v0.3 (on MultiArith) outputs irrelevant numeric sequences and terminates generation.

**Q:** Ned had to wash 9 short sleeve shirts and 21 long sleeve shirts before school. If he had only washed 29 of them by the time school started, how many did he not wash?

**A:** 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98. 99. 100. 101. 102. 103. 104. 105. 106. 107. 108. 109. 110. 111. 112. 113. 114. 115. 116. 117. 118. 119. 120. 121. 12

Table 29: Llama-3.1-8B (on CommonsenseQA) producing additional, unnecessary choices in multiple-choice scenarios.

**Q:** When drinking booze what can you do to stay busy? (A) reach tentative agreement (B) stay in bed (C) stop bicycle (D) examine thing (E) suicide

**A:** (F) stay busy (G) stay sober (H) stay in shape

Answer: F

Explanation: When drinking booze what can you do to stay busy? When drinking booze what can you do to stay busy?

Table 30: Gemma-3-7B (on MultiArith) arbitrarily generating answer choices even for non-multiple-choice questions.

**Q:** Luke was helping the cafeteria workers pick up lunch trays, but he could only carry 4 trays at a time. If he had to pick up 20 trays from one table and 16 trays from another, how many trips will he make?

**A:** A. 4

B. 5

C. 6

D. 7

E. 8

Show more

Step 1

1 of 2

Determine the number of trips Luke will make.

$$20 \div 4 = 5 \text{ trips}$$

Result

2 of 2

B. 5

Answer: F

Explanation: When drinking booze what can you do to stay busy? When drinking booze what can you do to stay busy?

Table 31: Gemma-3-7B (on Logical Deduction) generating an answer first before reasoning through it, but the subsequent solution lacks meaningful reasoning.

**Q:** The following paragraphs each describe a set of five objects arranged in a fixed order. The statements are logically consistent within each paragraph. On a shelf, there are five books: a brown book, a yellow book, an orange book, a green book, and a gray book. The green book is to the left of the gray book. The brown book is the third from the left. The gray book is the second from the right. The yellow book is to the left of the green book.

Options:

- (A) The brown book is the third from the left
- (B) The yellow book is the third from the left
- (C) The orange book is the third from the left
- (D) The green book is the third from the left
- (E) The gray book is the third from the left

**A:**

- (C) The orange book is the third from the left

Solution:

The green book is to the left of the gray book.  
The brown book is the third from the left.  
The gray book is the second from the right.  
The yellow book is to the left of the green book.

The green book is to the left of the gray book.  
The brown book is the third from the left.  
The gray book is the second from the right.  
The yellow book is to the left of the green book.

The green book is to the left of the gray book.  
The brown book is the third from the left.  
The gray book is the second from the right.  
The yellow book is to the left of the green book.

The green book is to the left of the gray book.  
The brown book is the third from the left.  
The gray book is the second from the right.  
The yellow book is to the left of the green book.



Table 32: Mistral-7B-v0.3 (on MultiArith) frequently generates "# Guidance" and "# Example", mimicking a few-shot style, which reflects the model's learned pattern for mathematical problem-solving.

**Q:** The school cafeteria had 23 apples. If they used 20 to make lunch for the students and then bought 6 more, how many apples would they have?

**A:** 26

### Guidance

The distributive property is a property of multiplication that allows us to distribute a factor over a sum or difference.

$$3(a + b) = 3a + 3b$$

$$5(a - b) = 5a - 5b$$

The distributive property is used to simplify expressions.

#### Example A

Simplify:  $5(a+b)$

Solution: Use the distributive property to distribute the 5 over the sum.

$$5(a + b) = 5a + 5b$$

$$= 5a + 5b$$

#### Example B

Simplify:  $3(a-b)$

Solution: Use the distributive property to distribute the 3 over the difference.

$$3(a - b) = 3a - 3b$$

$$= 3a - 3b$$

#### Example C

Simplify:  $2(a+b+c)$

Solution: Use the distributive property to distribute the 2 over the sum.

$$2(a + b + c) = 2a + 2b + 2c$$

$$= 2a + 2b + 2c$$

#### Concept Problem Revisited

The school cafeteria had 23 apples. If they used 20 to make lunch for the students and then bought 6 more, how many apples would they have?

The cafeteria had 23 apples. They used 20 to make lunch for the students. They bought 6 more.

$$23 - 20 = 3$$

$$20 + 3 = 23$$

The cafeteria had 23 apples. They used 20 to make lunch for the students. They bought 6 more.

$$23 - 20 = 3$$

$$20 + 3 = 23$$

The cafeteria had 23

### D.3 *Incomplete Reasoning: Resulting in an Incorrect or Partial Answer*

*Incomplete Reasoning* encompasses scenarios where the model initiates a chain of thought but stops prematurely or changes topic before reaching the correct conclusion. This phenomenon accounted for 5.45% of incorrect GSM8K responses in Llama 3.1-8B.

Table 33: Llama-3 8B (on MultiArith) failing to derive the final answer by prematurely stopping the Chain-of-Thought (CoT) process. The correct answer is 80, but the model halts before completing the final step of summing 32 and 48.

**Q:** Zoe was unboxing some of her old winter clothes. She found 8 boxes of clothing and inside each box there were 4 scarves and 6 mittens. How many pieces of winter clothing did Zoe have total?

**A:** 1. 8 scarves and 6 mittens 2. 8 scarves and 24 mittens 3. 32 scarves and 24 mittens 4. 32 scarves and 48 mittens

Zoe was unboxing some of her old winter clothes. She found 8 boxes of clothing and inside each box there were 4 scarves and 6 mittens. How many pieces of winter clothing did Zoe have total?

1. 8 scarves and 6 mittens 2. 8 scarves and 24 mittens 3. 32 scarves and 24 mittens 4. 32 scarves and 48 mittens

Table 34: Mistral-7B-v0.3 (on MultiArith) creating and solving its own few-shot examples instead of solving the given problem.

**Q:** Paige had 43 math problems and 12 science problems for homework. If she finished 44 of the problems at school, how many problems did she have to do for homework?

**A:** "1. 43 2. 44 3. 45 4. 46

### Watch This

Watch this video for help with the concepts in this section.

### Guidance

In the last section, we learned how to solve equations with fractions. In this section, we will learn how to solve equations with decimals.

#### Example A

Solve the equation for  $x$ .

$$0.2x + 0.3 = 0.5$$

Solution:

We can solve this equation by subtracting 0.3 from both sides.

$$0.2x + 0.3 - 0.3 = 0.5 - 0.3$$

$$0.2x = 0.2$$

Now we can divide both sides by 0.2.

$$\frac{0.2x}{0.2} = \frac{0.2}{0.2}$$

$$x = 1$$

#### Example B

Solve the equation for  $x$ .

$$0.05x + 0.02 = 0.07$$

Solution:

We can solve this equation by subtracting 0.02 from both sides.

$$0.05x + 0.02 - 0.02 = 0.07 - 0.02$$

$$0.05x = 0.05$$

Now we can divide both sides by 0.05.

$$\frac{0.05x}{0.05} = \frac{0.05}{0.05}$$

$$x = 1$$

#### Example C

Solve the equation for  $x$ .

$$0.005x + 0.002 = 0.007$$

Solution:

We can solve this equation by subtracting 0.002 from both sides.

$$0.005x + 0.002 - 0.002 = 0.007 - 0.002$$

$$0.005x = 0.005$$

Now we can divide both sides by 0.005

## E Error Analysis

This section provides an analysis of the errors observed in applying the AID method. The errors are categorized into four types. These fall into two broad groups: those where the injection had no effect on reasoning, and those where it altered the output but failed to overcome immature reasoning. The latter is further divided into three distinct patterns. The analysis is based on outputs from the Llama-3.1-8B model on GSM8K, with AID applied using the token “Well” under the  $k = 2$  setting. In each case, the model subsequently produced an incorrect answer.

### E.1 No Effect (4.40%)

When injection occurs after a substantial number of tokens have already been generated, the reasoning trajectory established prior to the injection often persists. In particular, if an incorrect answer has already been derived through a sequence of Chain-of-Thought (CoT) steps, the model tends to maintain this erroneous trajectory even after the injection. One potential solution can be to apply an adaptive  $k$  based on the length of the generation.

Table 35: Despite the injection of “Well,” the model maintained the previously derived incorrect answer of 77 and concluded the generation.

**Q:** A restaurant has 40 tables with 4 legs and 50 tables with 3 legs. Calculate the total number of legs the restaurant’s tables have.

**A (Gold Answer):** 310

**A (Zero-shot):** How many tables does the restaurant have? 40 tables with 4 legs = 160 legs 50 tables with 3 legs = 150 legs Total number of legs = 160 + 150 = 310 legs Total number of tables = 310/4 = 77.5 The restaurant has 77 tables. A restaurant has 40 tables with 4 legs and 50 tables with 3 legs. Calculate the total number of legs the restaurant’s tables have. How many tables does the restaurant have? 40 tables with 4 legs = 160 legs ... (repetition)

**A (Zero-shot + Ours):** How many tables does the restaurant have? 40 tables with 4 legs = 160 legs 50 tables with 3 legs = 150 legs Total number of legs = 160 + 150 = 310 legs Total number of tables = 310/4 = 77.5 The restaurant has 77 tables. A restaurant has 40 tables with 4 legs and 50 tables with 3 legs. Calculate the total number of legs the restaurant’s tables have. How many tables does the restaurant have? 40 tables with 4 legs = 160 legs 50 tables with 3 legs = 150 legs Total number of legs = 160 + 150 = 310 legs Total number of tables = 310/4 = 77.5 The restaurant has 77 tables. **Well**, I think that the restaurant has 77 tables.

### E.2 Immature → Immature (32.11%)

The injection prevented silence but failed to induce meaningful reasoning, resulting in a continuation of immature reasoning.

Table 36: Although the injection successfully broke the model’s silence, it led to a loop of meaningless repetition.

**Q:** A tub of ice cream costing \$13 is now sold at \$11. A packet of milk was sold at a discount of \$0.5. How much will you save if you buy 2 tubs of ice cream and 4 packets of milk?

**A (Gold Answer):** 6

**A (Zero-shot):** (Silence)

**A (Zero-shot + Ours):** **Well**, you will save \$2.5. This is because the discount is 13% of \$13 and 10% of \$5. The total discount is \$2.5. This is the same as 23% of \$11. So, the discount is 23% of \$11. This is the same as 23% of \$11. So, the discount is 23% of \$11 ... (repetition)

### E.3 Mature → Immature (0.29%)

The injection of "Well" can occasionally disrupt an otherwise coherent Chain-of-Thought (CoT) reasoning process. Even when greedy generation initially succeeds in producing a correct answer through mature reasoning, such injection may cause the model to deviate from the original trajectory and arrive at an incorrect conclusion.

Table 37: Prior to the injection of "Well," the model correctly answered 16 through coherent reasoning. However, the injection triggered a shift toward irrelevant and immature reasoning.

**Q:** Cynthia eats one serving of ice cream every night. She buys cartons of ice cream with 15 servings of ice cream per carton at a cost of \$4.00 per carton. After 60 days, how much will she spend on ice cream?

**A (Gold Answer):** 16

**A (Zero-shot):** How many cartons will she buy? Let  $x$  be the number of days. The number of cartons she buys is  $4x/15$ . The cost of each carton is \$4.00. The total cost is  $4x/15 \cdot 4$ . After 60 days, the total cost is  $4 \cdot 60 / 15 \cdot 4 = \$64.00$ . She will buy  $4 \cdot 60 / 15 = 16$  cartons.

**A (Zero-shot + Ours):** How many cartons will she buy? Let  $x$  be the number of days. The number of cartons she buys is  $4x/15$ . The cost of each carton is \$4.00. The total cost is  $4x/15 \cdot 4$ . After 60 days, the total cost is  $4 \cdot 60 / 15 \cdot 4 = \$64.00$ . She will buy  $4 \cdot 60 / 15 = 16$  cartons. **Well**, I'm not sure what you mean by "serving" of ice cream. If you mean a scoop, then you can use the same method as above. If you mean a cup, then you can use the same method as above. If you mean a pint, then you can use the same method as above. If you mean a quart, then you can use the same method as above. If you mean a gallon, then you can use the same method as above. If you mean a half-gallon, then you can use the same method as above. If you mean a pint, ... (repetition)

### E.4 Immature → Mature, but Failure (63.20%)

The injection successfully triggered logical reasoning, but the response remained incorrect due to earlier incorrect context or computational mistakes.

Table 38: The model initially generates answer choices arbitrarily, then produces a correct solution via appropriate Chain-of-Thought reasoning after the injection. However, it ultimately selects one of the earlier, incorrect choices, which does not include the correct answer derived from the reasoning.

**Q:** A water tank is filled with 120 liters of water. Celine used 90 liters of water from the tank to water her small garden. She was then able to collect rainwater that is twice as much as what was left. How many liters of water are in the tank now?

**A (Gold Answer):** 90

**A (Zero-shot):** A. 60 liters B. 120 liters C. 180 liters D. 240 liters A water tank is filled with 120 liters of water. Celine used 90 liters of water from the tank to water her small garden. She was then able to collect rainwater that is twice as much as what was left. How many liters of water are in the tank now? A. 60 liters B. 120 liters C. 180 liters D. 240 liters

**A (Zero-shot + Ours):** A. 60 liters B. 120 liters C. 180 liters D. 240 liters A water tank is filled with 120 liters of water. Celine used 90 liters of water from the tank to water her small garden. She was then able to collect rainwater that is twice as much as what was left. How many liters of water are in the tank now? A. 60 liters B. 120 liters C. 180 liters D. 240 liters **Well**, we know that Celine used 90 liters of water from the tank to water her small garden. So, that means that there are  $120 - 90 = 30$  liters of water left in the tank. She was then able to collect rainwater that is twice as much as what was left. So, we know that the amount of rainwater collected is  $2 \times 30 = 60$  liters. So, the total amount of water in the tank now is  $30 + 60 = 90$  liters. So, the answer is B.

## **F Artifact Licenses**

All artifacts employed in this study (i.e., models and datasets) are strictly governed by their respective license agreements. Specifically, models such as Llama-3.1-8B, Llama-3.2-1B, Mistral-7B-v0.3, Gemma-3-7B and QwQ-32B are subject to the terms and conditions set forth by their creators, which include usage restrictions, redistribution rights, and potential attribution requirements. For instance, some models may allow non-commercial research use but restrict commercial applications.

Likewise, the datasets utilized (e.g., GSM8K, MultiArith, StrategyQA, BigBench-Disambiguation QA, and BigBench-Logical Deduction) are released under open-access or research-oriented licenses. Researchers are advised to review any additional clauses that detail citation requirements, limitations on commercial use, or derivative works.

We confirm that all experiments in this work were conducted in strict compliance with these licenses. Any redistributions of the models or datasets, as well as any derivative works, must similarly adhere to the terms specified by the original authors and/or dataset providers. Detailed license information for each artifact can be found in the respective original publications and/or in the supplementary materials accompanying this paper.