THINKING ABOUT THINKING: METACOGNITIVE INFLUENCE TRACING FOR RELIABLE LLM REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Giving large language models (LLMs) *"time to think"* has emerged as a powerful strategy for enhancing reasoning. Prompting methods such as Chain-of-Thought (CoT) and reasoning-focused models like DeepSeek-R1 exemplify this paradigm. However, these approaches remain limited: they treat all reasoning steps as equally important, wasting computation and leaving the process vulnerable when fragile steps propagate errors. Inspired by findings in cognitive science on *critical periods and neural bottlenecks*—where certain experiences exert disproportionate influence—we introduce **Metacognitive Influence Tracing (`MIT`)**, a *diagnostic* method that identifies **critical junctures** in reasoning. `MIT` adopts influence principles inspired from cognitive science, and models the reasoning process as an influence graph. It then computes propagated influence via a **diffusion process** with the **heat kernel**, revealing latent *cognitive structural patterns* of machine reasoning in both models with explicit reasoning capability(reasoning models) and those without (non-reasoning models). Building on these insights, we propose **Adaptive Critical Sampling (`ACS`)**, an *intervention* framework that enhances Self-Consistency by selectively resampling at critical junctures rather than entire traces. This targeted resampling improves reasoning reliability while cutting redundant computation. Across six benchmark tasks and five models, `ACS` delivers an average accuracy gain of **7.48 points** while reducing computational cost by **59.75%**.

1 INTRODUCTION

Giving Large Language Models (LLMs) *"time to think"* (Fulford & Ng, 2023) has emerged as a powerful paradigm for unlocking their reasoning abilities. Prompting methods like Chain-of-Thought (CoT) (Wei et al., 2022a), which elicit step-by-step reasoning, and the development fo models with explicit reasoning capabilities, such as DeepSeek R1 (DeepSeek-AI et al., 2025), both leverage this principle. While these approaches extend the frontier of machine cognition, they also expose a common fragility: reasoning traces are easily disrupted by error propagation. Even a small mistake early on can disproportionately affect the final answer, leading to a confidently incorrect conclusion (Besta et al., 2024; Yao et al., 2024; Becker & Soatto, 2024).

A variety of strategies have been proposed to mitigate this brittleness, but they apply cognitive effort in broad or indiscriminate ways. **Self-Consistency** (Wang et al., 2023), is a brute-force approach that resamples multiple times. But it is computationally expensive because it repeatedly resamples entire reasoning trace. **Verification-based methods** (Cobbe et al., 2021; Lightman et al., 2023; Li et al., 2022) seek to validate or revise reasoning, but generally apply these checks indiscriminately without prioritization. Even sophisticated **tree-based approaches** (Yao et al., 2024; Hao et al., 2023), which use algorithms like Monte Carlo Tree Search to prioritize the most promising reasoning *paths*, lack a mechanism for identifying an intrinsically critical points *within* a single reasoning trace. Despite their advances, these paradigms still lack a metacognitive lens to determine critical junctures, where cognitive effort is most impactful, leading to an inefficient allocation of computational resources.

To address this gap, we turn to insights from cognitive science. Research on **critical periods** (Hensch, 2004) and **neural bottlenecks** (Dehorter & Del Pino, 2020) suggests that complex cognitive systems are not uniform. Instead, they contain essential moments—specific experiences or computational steps—that have a disproportionately large impact on the final outcome (Hensch, 2004). Recent research also find that tokens has various impact on the final answer where a wrong intermediate

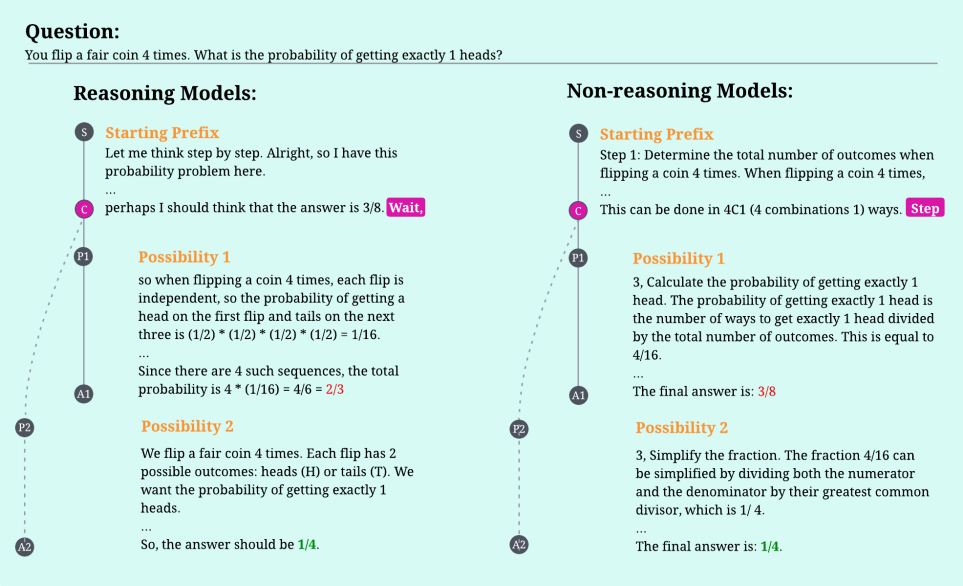


Figure 1: Existing methods often enhance performance by allowing LLMs additional "time to think." However, in both *Reasoning Models* like DeepSeek-R1 or *Non-reasoning Models* like LLaMA-3, some points within the reasoning trace are critical junctures that the final answer relies heavily on. Current approaches lack a principled mechanism to identify and prioritize such juncture points where cognitive effort has the greatest impact on the overall outcome. Our research surfaced such critical junctures while revealing cognitive markers LLM uses such as *"Wait"* or *"Step"*.

value still leads to the correct answer (Barez et al., 2025; Lin et al., 2024; Qian et al., 2025). We hypothesize that a similar principle governs LLM reasoning: their thought processes are not chains of equal links, but are defined by **critical junctures** upon which the final answer depends heavily on. An intervention at these precise moments should yield the greatest return on computational investment.

Driven by this insight, we introduce **Metacognitive Influence Tracing (`MIT`)**, a novel diagnostic method that aims at surfacing critical junctures in a model's reasoning, which are the spots where cognitive investment provide most benefits. It provides a metacognitive perspective— *thinking about the ways models think*—by transforming models' internal attention mechanism. Instead of using raw attention scores, which provide a noisy, one-step view of influence, `MIT` adopt three influence principles inspired by cognitive science, and models the entire reasoning trace as an influence graph where attention patterns form the initial connections. It then computes propagated influence by modeling a **diffusion process** with the **heat kernel** and detects consensus using **head-wise kurtosis**. We discovered that `MIT` uncovers a latent *cognitive structural pattern* underlying the reasoning process in both models with explicit thinking and reasoning capabilities (e.g. DeepSeek-R1), which we denote *Reasoning Models*, and those without (e.g. Llama-3), which we denote *Non-reasoning Models*. Building on this, we propose **Adaptive Critical Sampling (`ACS`)**, an intervention framework that integrates `MIT`'s insights to enhance Self-Consistency by focusing resampling efforts exclusively on identified critical junctures. Our work makes three core contributions:

- We propose a novel diagnostic method, **`MIT`**, which identifies critical junctures in reasoning traces across diverse model types. Leveraging influence principles to reinterpret the attention mechanism, `MIT` models the reasoning process with graph-based diffusion model. To the best of our knowledge, `MIT` is the first method to identify such critical junctures *within a single, complete reasoning trace*.
- We introduce a high-efficiency intervention framework, **`ACS`**, that enhances reasoning performance in both reasoning and non-reasoning models, leveraging the surgical insights from `MIT`.
- We conduct **extensive experiments** across six benchmarks covering arithmetic, commonsense, logical, and symbolic reasoning tasks, as well as five diverse models, demonstrating the broad applicability of our methods. By *"thinking about thinking,"* `ACS` achieves an average performance gain of **7.48 points** while reducing computational costs by an average of **59.75%**.

2 A Principled Framework for Metacognitive Analysis

Analyzing the step-by-step reasoning process is fundamentally a problem of understanding the relationships between the generated tokens. While a language model produces many internal signals such as entropy, the **attention mechanism** is the only component explicitly designed to model these token-to-token interactions. It is the underlying engine of relational information in a Transformer. However, raw attention scores provide a noisy and myopic view, suffering from known artifacts like attention sinks (Zhao et al., 2023) and over-attention to punctuation (Clark et al., 2019), where a few tokens attract disproportionate influence, obscuring the true logical flow. Also only direct influence is calculated whereas indirect impact is missing. To overcome these limitations, we introduce a framework built on two core innovations. First, we propose a set of cognitive science inspired **Influence Principles** to purify the raw attention signal. Second, we introduce a **Propagation Model** based on network science to capture the multi-step or multi-trace nature of reasoning.

2.1 Innovation 1: Influence Principles for Signal Purification

To isolate the true logical signal from the noise inherent in raw attention, we establish three principles when studying influence between tokens. Inspired by cognitive science, we treat semantic tokens as proxies for **concepts**—a term widely used in cognitive science to denote the basic units of thought. Such principles allow us to reinterpret the attention mechanism through this conceptual lens.

- **Influence is Contextual, Not Self-Referential.** A concept's meaning is defined by its interaction with other concepts. Inspired by the contextual modulation of neural responses in the brain (Reynolds & Desimone, 2003), we interpret the attention between different tokens as the primary signal of this conceptual interplay. To focus on this relational information, our method therefore excludes a token's self-attention, which is a non-contextual operation.
- **Influence is Semantic, Not Syntactic.** Logical reasoning operates on the meaning of concepts, not their grammatical arrangement. Supported by psycholinguistic evidence that humans recall meaning (semantics) over form (syntax) (Sachs, 1967), we aim to analyze the conceptual backbone of the reasoning trace. To achieve this, our method filters out non-semantic tokens (e.g., punctuation, stopwords) that serve primarily grammatical, rather than logical functions.
- **Influence Propagates.** The impact of a single concept is not confined to an immediate step but propagates through a logical chain. This is analogous to *spreading activation* in cognitive networks, where activating one concept sends a ripple of influence to others (Collins & Loftus, 1975). To capture this phenomenon, our framework must therefore move beyond single-step attention scores to model the full, multi-step dependencies of the influence flow.

2.2 Innovation 2: Modeling Influence Propagation as a Diffusion Process

While our influence principles purify the initial signal, they do not address the one-step, myopic nature of raw attention. To apply the principle of propagation, we model the reasoning trace as a directed graph and propose that the total, propagated influence can be measured by modeling a **diffusion process** on this graph. This approach naturally includes all possible influence paths while allowing us to control the extent of the diffusion.

This process is elegantly captured by the **matrix exponential**, which we augment with a damping factor γ to control the intensity of the influence spread. For a graph with an adjacency matrix A (our purified attention matrix), the total influence S is given by the Taylor series expansion of $\exp(\gamma A)$. The term $\frac{(\gamma A)^k}{k!}$ represents the influence over paths of length k, with the factorial denominator acting as a strong penalty for longer paths and γ scaling the overall propagation effect. The total influence is the sum over all possible path lengths:

$$S = \exp(\gamma A) = I + \gamma A + \frac{(\gamma A)^2}{2!} + \frac{(\gamma A)^3}{3!} + \dots \tag{1}$$

where S is the final influence matrix. The damping factor γ is a tunable hyperparameter that controls the "reach" of influence; a smaller γ restricts influence to more local paths, while a larger γ allows it to diffuse more widely. This formulation allows us to compute the total, propagated influence in a single, theoretically-grounded step.

3 METHODOLOGY

3.1 THE DIAGNOSTIC METHOD: MIT

We now introduce **Metacognitive Influence Tracing (MIT)**, a diagnostic method that embodies our principled framework. MIT is a multi-stage pipeline designed to systematically refine the raw, noisy attention patterns from a Transformer to surface a single, identified critical token. The full procedure of MIT can be found in Appendix Algorithm 2.

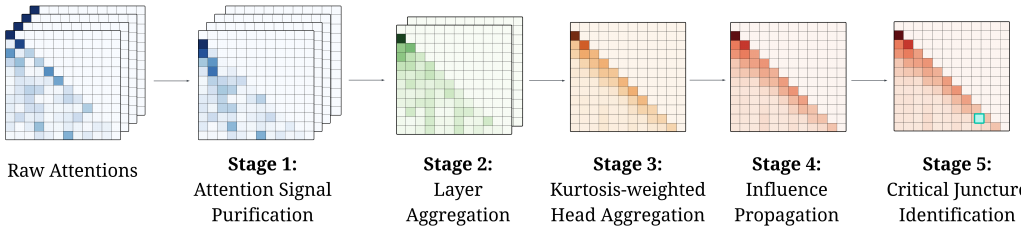


Figure 2: The MIT algorithm, surfacing a single critical token from raw multi-layer, multi-head attention matrices through cleaning, aggregations, propagation, and scoring.

3.1.1 STAGE 1: ATTENTION SIGNAL PURIFICATION

The first stage applies our influential principles to the raw attention signal. For a given reasoning trace with K tokens, we start with the full set of raw attention matrices, $\{A_{l,h} \in \mathbb{R}^{K\times K}\}$ for all layers l and heads h. We apply the following transformations to each individual matrix:

- **Contextual Filtering:** We enforce the principle of contextual influence by zeroing out the diagonal of each matrix, thereby removing self-attention scores.
- **Semantic Filtering:** We follow the semantic principle by zeroing out attentions to non-semantic tokens (e.g., stopwords, punctuation) and special tokens (e.g., <bos>).

The output of this stage is a set of purified attention matrices, $\{A'_{l,h} \in \mathbb{R}^{K\times K}\}$.

3.1.2 STAGE 2: LAYER AGGREGATION

Next, to handle the layer dimension, we compute an average attention matrix for each head across all L layers. This creates one consensus matrix for each of the H heads, resulting in a tensor $A'' \in \mathbb{R}^{H\times K\times K}$.

$$A''_h = \frac{1}{L}\sum_{l=1}^{L} A'_{l,h} \quad \text{for each head } h \in [1, H] \tag{2}$$

3.1.3 STAGE 3: KURTOSIS-WEIGHTED HEAD AGGREGATION

This stage distills the multi-head tensor A'' into a single base influence matrix $A \in \mathbb{R}^{K\times K}$. We call the token doing the attending the *target token*, and the token being attended to the *source token*. Source token would be strictly in front of the target token. We compute a unique set of head weights for each target token, hypothesizing that heads with more focused attention patterns are more involved in the reasoning for that specific step.

For each target token i, we calculate an importance weight, $w_{i,h}$, for each head h by computing the kurtosis of that head's attention distribution over all source tokens. A high kurtosis indicates a highly peaked, specialized attention pattern.

$$w_{i,h} = \text{Kurtosis}(A''_{h,i,:}) \tag{3}$$

These weights are then normalized across the heads (for each target token i) to form $w'_{i,h}$. The final entry A_{ij} in our base influence matrix is the weighted sum of the influence scores from token j to

token i across all heads:

$$A_{ij} = \sum_{h=1}^{H} w'_{i,h} \cdot A''_{h,ij} \tag{4}$$

The resulting matrix A represents the purified, consensus-based, one-step influence between tokens.

3.1.4 STAGE 4: INFLUENCE PROPAGATION

With the single, consensus-based influence matrix A from the previous stage, we now apply the principle of propagation. We apply the diffusion model by computing the matrix exponential, as defined in Equation 1. This computation transforms the one-step influence matrix A into the final propagated influence matrix S, where each entry S_{ij} represents the total, accumulated influence of token j on token i over all possible paths.

3.1.5 STAGE 5: CRITICAL JUNCTURE IDENTIFICATION

The final stage uses the propagated influence matrix S to find the single most influential token. Our goal is to identify which token j within the set of CoT steps (T_{CoT}) had the greatest total influence on the generation of the set of final answer tokens (T_{ans}). We calculate an influence score for each candidate token by summing its influence on all tokens in the final answer:

$$\text{Score}(j) = \sum_{i \in T_{ans}} S_{ij} \quad \text{for } j \in T_{CoT} \tag{5}$$

The token with the maximum score is designated as the critical juncture of the reasoning chain:

$$j_{critical} = \underset{j \in T_{CoT}}{\text{argmax}}\, \text{Score}(j) \tag{6}$$

This token, $j_{critical}$, is the final output of the `MIT` algorithm.

3.2 VALIDATION OF `MIT` - COGNITIVE MARKERS

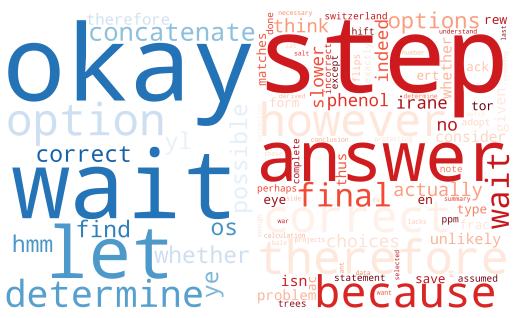
We validate that `MIT` functions as a meaningful analysis tool by showing that the critical junctures it identifies are not random but map onto a consistent cognitive structural pattern, which we call **cognitive markers**. We hypothesize LLM uses such markers as reasoning checkpoint and attend to them heavily when generating the final answer. Additional analysis are in Appendix Section A.7.

Table 1: Top critical junctures identified by `MIT`, aggregated by model type.

Model Class	**Dominant Cognitive Markers**	**Interpretation of Reasoning Style**
Reasoning Model	'okay' (54.4%), 'wait' (25.2%)	Employs dominant **cognitive markers**, indicating internal state management with explicit decision checkpoints.
Non-reasoning Model	'step' (23.2%), 'answer' (9.4%), 'therefore' (3.3%)	Relies on diverse **explicit procedural and linguistic markers** ('step', 'answer', logical connectives)

Applying `MIT` across both Non-reasoning Model and Reasoning Model, we find that such cognitive markers differ systematically. As summarized in Table 1 and visualized in Figure 3a (left), our analysis reveals that reasoning models have developed highly convergent cognitive strategies. They overwhelmingly rely on cognitive markers, with *"okay"* and *"wait"* collectively accounting for **79.6%** of all critical junctures identified across reasoning models. This concentration suggests that these models have learned consistent, abstract mechanisms to manage their internal cognitive state and implement deliberative pauses before proceeding with crucial reasoning steps.

In contrast, non-reasoning models ground their reasoning in the explicit structure of language and procedural organization. Instead of concentrated cognitive markers, they employ a wider variety of explicit procedural and linguistic tokens, such as *"step"* (23.2%), *"answer"* (9.4%), and logical connectives like *"therefore"* (3.3%), as their critical junctures. The more distributed concentration



(a) The distribution of top critical tokens. Reasoning models(left) show a highly concentrated strategy around the token *'wait'*, while non-reasoning models(right) exhibit more diverse patterns.

(b) An example of critical token using cognitive marker *"Wait"*, allows ACS to move away from ruminating on what the sequence is and arrive at the right answer. Details in Appendix Fig. 25.

Figure 3: Distribution and Effect of Cognitive Markers. More examples in Appendix A.12.

across diverse token types, as visualized in Figure 3a (right), suggests a more surface-level reasoning process that relies on overt textual structure and sequential organization patterns rather than internal state management. The ability of MIT to quantitatively distinguish between these two distinct cognitive styles—concentrated cognitive control versus distributed procedural reasoning—validates it as a powerful lens into the underlying mechanics of machine reasoning.

3.3 Application: Adaptive Critical Sampling

Having validated the ability of MIT to precisely identify critical junctures, we now introduce **Adaptive Critical Sampling (ACS)**, a framework designed to translate these metacognitive insights into significant efficiency and performance gains. Instead of the brute-force, full-trace resampling of Self-Consistency, ACS performs a surgical intervention, focusing computational effort only on the most critical juncture identified by MIT. This allows for a more intelligent allocation of resources, enhancing reasoning reliability without the prohibitive cost of existing methods. The full procedure is detailed in Algorithm 1.

A key advantage of this approach is its broad applicability; ACS provides a unified intervention strategy that is effective for both non-reasoning models and reasoning models . Examples of how ACS improves accuracy are in Appendix A.12.

Algorithm 1 Adaptive Critical Sampling (ACS)

Require: Query Q, Model M, Number of samples N
1: Generate initial trace $T_0 \leftarrow M(Q)$
2: Identify critical juncture token $j_{critical} \leftarrow \text{MIT}(T_0)$
3: Let T_{prefix} be the tokens in T_0 up to $j_{critical}$
4: **for** $k = 1$ to N **do**
5: Generate new suffix $T_{suffix,k} \leftarrow M(T_{prefix})$
6: Store final answer from $T_{prefix} \circ T_{suffix,k}$
7: **end for**
8: **return** Majority vote of the stored final answers

4 Experiments

4.1 Experimental Setup

Models. We evaluate on five LLMs spanning a wide range of capacities and reasoning behaviors: `Llama-3.1-8B-Instruct`, `DeepSeek-R1-Distill-Qwen-1.5B`, `SmolLM3-3B` in both *general* and *thinking* modes, and `Gemma-3-270M-IT`. This mix includes reasoning models with explicit reasoning and thinking capabilities and non-reasoning models without such capabilities,

enabling controlled comparisons across settings. We also intentionally focus on models from different families and sizes to showcase the broad applicability of our approaches.

Datasets. We group benchmarks by the primary reasoning skill they target: **Logical reasoning:** GPQA-DIAMOND (Rein et al., 2023); **Arithmetic reasoning:** MATH500 (Hendrycks et al., 2021) and AIME24 (Jia, 2024); **Commonsense reasoning:** STRATEGYQA (Geva et al., 2021); **Symbolic reasoning:** COINFLIP and LAST LETTER from Wei et al. (2022a). More detailed information about datasets and task taxonomy can be found in Appendix Section A.2.1.

Baselines and Hyperparameters. For baselines we compare with CoT and Self-Consistency with 9 samples, denoted CoT-SC. In the sampling process for ACS, we keep the prefix fixed and generate additional 8 candidates. This ensures that our approaches and Self-Consistency baseline have the same total number of samples. Temperature is set to be 0.7 throughout the experiments. For additional information in experiment set up please refer to Appendix Section A.2.2.

4.2 MAIN RESULTS

Table 2: Main results across six benchmarks and five models. We use accuracy percentage as the metric as multiple samples are generated. Our method ACS consistently outperforms CoT and CoT-SC across diverse reasoning tasks in both reasoning and non-reasoning models.

		Logical	Arithmetic		Commonsense	Symbolic		Avg.
Model	**Method**	GPQA-D	MATH500	AIME24	STRATEGYQA	COINFLIP	LASTLETTER	
Reasoning models								
DeepSeek -Distilled -Qwen-1.5B	CoT	29.62	86.53	33.33	69.05	95.31	79.25	65.85
	CoT-SC	40.74	88.46	33.33	**83.33**	95.31	79.25	70.40
	ACS (Ours)	**44.44**	**92.31**	**33.33**	80.95	**98.43**	**94.34**	**73.30**
SmolLM3-3B (Thinking Mode)	CoT	53.90	97.62	44.83	61.86	100.00	100.00	76.70
	CoT-SC	57.45	100.00	48.28	76.29	100.00	100.00	80.67
	ACS (Ours)	**59.57**	**100.00**	**51.72**	**90.72**	**100.00**	**100.00**	**83.67**
Non-reasoning models								
LLaMA-3.1-8B	CoT	43.18	83.56	38.95	72.53	97.00	84.44	69.94
	CoT-SC	38.64	86.30	42.86	76.92	97.00	91.11	72.14
	ACS (Ours)	**61.36**	**89.04**	**57.14**	**80.22**	**99.00**	**93.33**	**80.68**
SmolLM3-3B (General Mode)	CoT	35.48	83.33	25.00	61.54	95.65	83.63	64.11
	CoT-SC	40.32	85.71	33.33	69.23	97.10	87.27	68.16
	ACS (Ours)	**46.77**	**85.71**	**33.33**	**76.92**	**98.55**	**89.09**	**71.40**
Gemma-3 -270M-IT	CoT	25.00	42.42	9.09	10.29	73.33	4.76	27.82
	CoT-SC	16.67	45.45	**27.27**	10.29	73.33	2.38	29.23
	ACS (Ours)	**33.33**	**48.48**	18.18	**16.18**	**73.33**	**7.14**	**32.77**

Consistent Performance Gains. The primary finding from Table 2 is the consistent and significant performance improvement provided by ACS. Across all five model classes and all six reasoning benchmarks, ACS achieves the highest average accuracy. On average, ACS delivers a +7.5 point gain over CoT and a +4.2 point gain over CoT-SC. On the challenging GPQA-DIAMOND benchmark, ACS raises the accuracy of `LLaMA-3.1-8B` from 43.18% (CoT) and 38.64% (CoT-SC) to 61.36%, a jump of 18.18 absolute points over CoT and 22.72 points over CoT-SC.

Universal Applicability Across Model Types. A key goal of our work was to develop a universal framework effective for all types of models. The results validate this design: ACS provides improvements for both specialized "reasoning" models and general-purpose "non-reasoning" models. For instance, `SmolLM3-3B (Thinking Mode)` benefits substantially, reaching 90.72% on STRATEGYQA, up from 61.86% with CoT and 76.29% with CoT-SC. Meanwhile, ACS also boosts non-reasoning models, such as `SmolLM3-3B (General Mode)`, which improves on STRATEGYQA from 61.54% (CoT) and 69.23% (CoT-SC) to 76.92%. These results demonstrate that our

MIT-based diagnostic can identify meaningful critical junctures regardless of whether reasoning is explicit or emergent, making ACS a general-purpose enhancement for LLM reasoning.

Scalability and Model Size. The performance improvements scale consistently across models of different sizes. The results suggest that the larger the model, the larger the absolute improvement. LLaMA-3.1-8B benefits most, with ACS boosting its overall accuracy by +10.7 points compared to CoT and +8.5 points compared to CoT-SC. In contrast, the smallest model, Gemma-3-270M, still sees consistent but more modest gains of +5.0 points over CoT and +3.5 points over CoT-SC. These results show that ACS not only strengthens strong models but also yields meaningful improvements for small-scale models, highlighting the scalability and broad applicability of our approach.

4.3 Analysis

Table 3: We conducted additional analysis on our design with GPQA-Diamond data using DeepSeek-Qwen-1.5B (reasoning model) and LLaMA-3.1-8B (non-reasoning model).

		Reasoning Model		Non-Reasoning Model	
	Variant	**Accuracy (%)**	**Δ vs Full**	**Accuracy (%)**	**Δ vs Full**
Principals	w/o Signal Purification	14.28	-28.57	33.33	-27.78
Aggregation	*Layer Agg:* Average + Linear Decay	30.77	-12.08	55.56	-16.67
	Layer Agg: Attention Rollout	21.43	-21.42	44.44	-2.60
	Layer Agg: Average (Ours)	**42.85**	–	**61.11**	–
	Head Agg: Max Attention	35.72	-7.13	50.00	-11.11
	Head Agg: Mean Attention	38.46	-3.60	50.00	-11.11
	Head Agg: Kurtosis (Ours)	**42.85**	–	**61.11**	–
Diffusion	w First Order Kernel (no propagation)	21.43	-21.42	33.33	-27.78
	w Second Order Kernel	28.57	-14.28	38.89	-22.22
	w Resolvent Diffusion Kernel	35.71	-7.14	38.89	-22.22
	w Heat Kernel (Ours)	**42.85**	–	**61.11**	–

Effect of Influential Principles. By following the influential principles in Section 2.1, we introduce *Stage 1: Signal Purification* in the MIT method, which removes attention to non-semantic tokens and self-attention. Disabling this stage results in a dramatic drop in performance: **-28.57** accuracy points for the reasoning model and **-27.78** for the non-reasoning model. This highlights the importance of filtering noisy or non-semantic dependencies before applying structural analysis to the attention graph and the effectiveness of our influential principles. In fact, the introduction of influential principles provide the most performance gain compared to aggregation strategies and diffusion process. More work on the purification effectiveness can be found in in Appendix A.6.

Effect of Aggregation Strategies. We evaluate both layer-wise and head-wise aggregation strategies. For layer aggregation, our simple average pooling significantly outperforms alternatives: applying linear decay leads to a **-12.08** drop in the reasoning model, and using Attention Rollout results in a **-21.42** reduction. This suggests that averaging captures structural signals more robustly than decay or recursive schemes. On the head level, our kurtosis-based aggregation, which quantifies sharpness and consensus, consistently outperforms mean or max attention heuristics. Replacing kurtosis with max pooling leads to **-7.13** degradation (reasoning) and **-11.11** (non-reasoning), while mean pooling performs only slightly better. These results affirm the importance of capturing the distributional shape of attention, rather than relying on scalar summaries alone. More study on aggregation strategies can be found in Appendix A.6.

Effect of Diffusion Processes. We evaluate the impact of the diffusion module (Section 2.2) by varying the choice of propagation kernel. Restricting diffusion to a first-order kernel (i.e., no multi-hop propagation) results in a sharp performance drop, **-21.42** points on the reasoning model and **-27.78** on the non-reasoning model. This introduce the largest performance drop in all kernels, highlighting the importance of propagation. Extending to a second-order kernel partially recovers performance but remains inferior to the full model. Among deeper kernels, the resolvent (Neumann) kernel outperforms the shallow variants yet still falls short of the heat kernel. The heat kernel (matrix

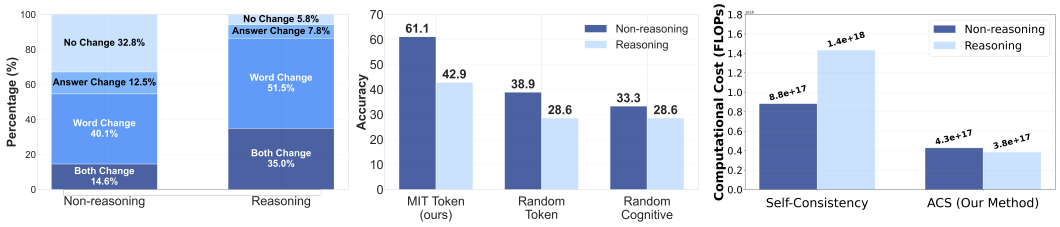
exponential) consistently achieves the best results across both model types. This pattern supports that crucial source–target relations are typically captured by short or middle paths, and that influence should decay with path length. Further analyses of propagation kernels are provided in Appendix A.4.

5 Discussion

Are critical junctures markers or causes? A natural question is whether a critical juncture should be understood as a causal point where changing the token directly alters the outcome. To examine this, we measured the correlation between differences in the sampled token at a critical juncture and differences in the resulting output. As shown in Figure 4a, we found no significant correlation. This indicates that the juncture's importance does not lie in the immediate lexical choice, nor does it represent a direct causal driver. Instead, we interpret it as an *marker of intervention sensitivity*—a point in the reasoning process where cognitive investment like resampling tends to be more impactful for downstream outcomes. We hypothesize that these junctures correspond to higher-variance regions in the model's latent space. Additional analysis on critical junctures is provided in Appendix A.7.

Are all the cognitive markers created equal? To confirm that MIT is identifying genuinely critical tokens, we compared our targeted intervention against two baselines in Figure 4b. Resampling from a random token yields poor results, dropping accuracy by **-14.28 points** (reasoning model) and **-22.22 points** (non-reasoning model). Even when we restrict the intervention to a randomly selected cognitive marker, performance is equally low. This proves that cognitive markers might come in the format of *"Wait", "Okay"*, but not all of them serve equally. The gains from ACS are critically dependent on the *precise* identification of the most influential juncture via MIT.

Are ACS expensive? To calculate the end to end cost for ACS including MIT, we use floating-point operations (FLOPs) as the metric for cost analysis since it fully covers the additional computation needed for MIT. The full detail of our cost calculation method is in Appendix A.8. As shown in Figure 4c, we reduce the average computational cost by approximately **-73.20%** for reasoning models and **-46.30%** for non-reasoning models, demonstrating substantial efficiency gains while improving performance. Additional analysis can be found in Appendix A.8.5.



(a) Token Changes Analysis (b) Token Selection Analysis (c) Efficiency Analysis

Figure 4: Discussion Analysis

6 Conclusion

In this paper, we introduce Metacognitive Influence Tracing (MIT), a diagnostic method that *"thinks about how LLMs think"* and identifies critical junctures in reasoning traces. MIT adopts insights from cognitive science and transform raw attention mechanism to accurately model the reasoning process as an influence graph. Powered by MIT, we also introduce Adaptive Critical Sampling (ACS), an intervention framework that exploits these junctures to resample selectively. Across six benchmarks and five models, ACS improves accuracy by an average of +7.48 points over CoT while cutting computation by 59.75% over Self-Consistency methods across both reasoning and non-reasoning models. To the best of our knowledge, MIT is the first method that is able to identify critical junctures within reasoning traces. It also surfaces distinct reasoning cognitive structural patterns via cognitive markers, offering a metacognitive lens into how LLMs *"think"*. Together, MIT and ACS show that targeting reasoning at its most influential moments yields systems that are more reliable and efficient.

Reproducibility Statement

We have taken extensive steps to ensure the reproducibility of our results. The methodology behind Metacognitive Influence Tracing (MIT) and Adaptive Critical Sampling (ACS) is described in detail in Sections 2–3 of the main text, with complete algorithmic procedures provided in Appendix Section A.3 for MIT and Algorithm 1 for ACS. To facilitate independent verification, we include a thorough description of datasets and task taxonomy in Appendix A.2.1, along with all experimental hyperparameters and setup details in Appendix A.2.2. Additional ablation studies (Appendix A.6–A.8) and sensitivity analyses (e.g., propagation parameters in Appendix A.5) document the robustness of our findings across design choices. Dataset sources such as GPQA-Diamond, MATH500, AIME24, StrategyQA, CoinFlip, and Last Letter are publicly available, and we describe pre-processing steps in Appendix A.2.1. For interpretability and replication of key results, we provide examples of MIT's diagnostic outputs and ACS interventions in Appendix A.12, as well as failure cases for transparency.

References

Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 4190–4197, 2020. URL https://aclanthology.org/2020.acl-main.385.

Fazl Barez, Tung-Yu Wu, Iván Arcuschin, Michael Lan, Vincent Wang, Noah Siegel, Nicolas Collignon, Clement Neo, Isabelle Lee, Alasdair Paren, Adel Bibi, Robert Trager, Damiano Fornasiere, John Yan, Yanai Elazar, and Yoshua Bengio. Chain-of-thought is not explainability. *Oxford Martin Human-Centred Explainable AI (AIGI) Working Paper*, July 2025. Working paper; published via Oxford Martin AIGI.

Evan Becker and Stefano Soatto. Cycles of thought: Measuring llm confidence through stable explanations. *arXiv preprint arXiv:2406.03441*, 2024. Proposes a framework using stability of generated explanations to quantify LLM confidence, improving uncertainty estimates over baselines.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michał Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 38, pp. 17682–17690. AAAI Press, 2024. doi: 10.1609/aaai.v38i16.29720. URL https://ojs.aaai.org/index.php/AAAI/article/view/29720.

Eric Bigelow, Ari Holtzman, Hidenori Tanaka, and Tomer Ullman. Forking paths in neural text generation. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=forking-paths. Published as a conference paper at ICLR 2025.

Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 782–791, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Chefer_Transformer_Interpretability_Beyond_Attention_Visualization_CVPR_2021_paper.html.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do not think that much for 2+3 = ? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does bert look at? an analysis of bert's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 276–286, 2019. doi: 10.18653/v1/W19-4828.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Łukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John

Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. Introduces GSM8K dataset and training verifiers for solution validation.

Allan M. Collins and Elizabeth F. Loftus. A spreading-activation theory of semantic processing. *Psychological Review*, 82(6):407–428, 1975. doi: 10.1037/0033-295X.82.6.407.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, Nicholas Thumiger, Aditya Desai, Ion Stoica, Ana Klimovic, Graham Neubig, and Joseph E. Gonzalez. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*, 2025. doi: 10.48550/arXiv.2502.08235. URL https://arxiv.org/abs/2502.08235.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, January 2025. Introduces DeepSeek-R1-Zero and DeepSeek-R1 trained via RL with minimal or no SFT, achieving performance comparable to OpenAI-o1-1217.

Nathalie Dehorter and Isabel Del Pino. Shifting developmental trajectories during critical periods of brain formation. *Frontiers in Cellular Neuroscience*, 14:83, 2020. doi: 10.3389/fncel.2020.00283. Discusses how developmental trajectories are constrained by neural "bottlenecks," where specific processes exert outsized influence on long-term outcomes.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3563–3578, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.212. URL https://aclanthology.org/2024.findings-acl.212/.

Yihe Dong, Simon Kornblith, Mohammad Norouzi, and David J. Fleet. Attention sinks in vision transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 10143–10154, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f7f0e2a5b91c1c7a46a70f6b8a1b95e9-Abstract.html.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2210.00720. Preprint, arXiv:2210.00720, 2022.

Isa Fulford and Andrew Ng. Chatgpt prompt engineering for developers, 2023. URL https://www.deeplearning.ai/short-courses/chatgpt-prompt-engineering-for-developers/. DeepLearning.AI OpenAI short course.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021. URL https://arxiv.org/abs/2101.02235.

Shibo Hao, Yifan Gu, Haoyu Ma, J. J. Hong, Ziyang Wang, Da Zheng, and Zhiting Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021. URL https://arxiv.org/abs/2103.03874.

Takao K. Hensch. Critical period regulation. *Annual Review of Neuroscience*, 27:549–579, 2004. doi: 10.1146/annurev.neuro.27.070203.144327. Seminal review explaining how developmental "critical periods" represent windows of heightened neural plasticity, where specific experiences shape long-term circuit function.

John Hewitt and Christopher D. Manning. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 4129–4138, 2019. URL https://aclanthology.org/N19-1419.

Sarthak Jain and Byron C. Wallace. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 3543–3556, 2019. URL https://aclanthology.org/N19-1357.

Maxwell Jia. Aime_2024: American invitational mathematics examination 2024 (i & ii). Hugging Face Datasets, 2024. URL https://huggingface.co/datasets/Maxwell-Jia/AIME_2024. 30 problems filtered from AIME 2024; community dataset card.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. Attention is not only a weight: Analyzing transformers with attention sinks. In *Proceedings of the 2020 ACL Workshop BlackboxNLP*, pp. 6–15, 2020. URL https://aclanthology.org/2020.blackboxnlp-1.1.

Takeshi Kojima, Shixiang "Shane" Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems (NeurIPS 2022)*, 2022. URL https://arxiv.org/abs/2205.11916. Preprint, arXiv:2205.11916.

Yixuan Li, Ziyang Fu, Zhexin Zhang, Zhiyang Teng, and William Yang Wang. Making large language models better reasoners with step-aware verifier. *arXiv preprint arXiv:2206.02336*, 2022.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *arXiv preprint arXiv:2305.20050*, 2023. Explores outcome vs. process supervision for intermediate reasoning correctness.

Zicheng Lin, Tian Liang, Jiahao Xu, Xing Wang, Ruilin Luo, Chufan Shi, Siheng Li, Yujiu Yang, and Zhaopeng Tu. Critical tokens matter: Token-level contrastive estimation enhances llm's reasoning capability. *arXiv preprint arXiv:2411.19943v2*, 2024. URL https://arxiv.org/abs/2411.19943v2.

Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling. *arXiv preprint arXiv:2502.06703*, 2025. URL https://arxiv.org/abs/2502.06703. Explores test-time scaling strategies and shows that smaller LLMs (e.g., 1 B) can outperform much larger ones (e.g., 405 B) on MATH-500 and AIME24 with optimal compute allocation.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 14014–14024, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/e0eacd98397163492779cba04e15d053-Abstract.html.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Tamera Lanham, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Andy Jones, Anna Goldie, Sam Bowman, Zac Hatfield-Dodds, Tom Conerly, Sheer El-Showk, Daniel Hernandez, Jackson Kernion, Kai Niederhausen, Deep Ganguli, Jared Kaplan, Sam McCandlish, Tom Brown, and Chris Olah. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022. URL https://arxiv.org/abs/2209.11895.

Jianfeng Pan, Senyou Deng, and Shaomang Huang. Coat: Chain-of-associated-thoughts framework for enhancing large language models reasoning. *arXiv preprint arXiv:2502.02390*, 2025. doi: 10.48550/arXiv.2502.02390. URL https://arxiv.org/abs/2502.02390.

Chen Qian, Dongrui Liu, Haochen Wen, Zhen Bai, Yong Liu, and Jing Shao. Demystifying reasoning dynamics with mutual information: Thinking tokens are information peaks in llm reasoning. *arXiv preprint arXiv:2506.02867v1*, 2025. URL https://arxiv.org/abs/2506.02867v1.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023. URL https://arxiv.org/abs/2311.12022.

John H. Reynolds and Robert Desimone. Interacting roles of attention and visual salience in the representation of objects in the brain. *Journal of Vision*, 3(11):1–1, 2003. doi: 10.1167/3.11.1.

Jacqueline S. Sachs. Recognition memory for syntactic and semantic aspects of connected discourse. *Perception & Psychophysics*, 2(9):437–442, 1967. doi: 10.3758/BF03208784.

Sofia Serrano and Noah A. Smith. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 2931–2951, 2019. URL https://aclanthology.org/P19-1282.

Zhiqing Sun, Hongkun Yu, Tongzhou Wang, and Yiming Yang. A closer look at attention sinks in transformers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1234–1245, 2022. URL https://aclanthology.org/2022.emnlp-main.88.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 4593–4601, 2019. URL https://aclanthology.org/P19-1452.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 5797–5808, 2019. URL https://aclanthology.org/P19-1580.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain-of-thought reasoning in language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023. Introduces the "self-consistency" decoding method, which samples multiple reasoning paths and marginalizes over them to improve accuracy.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022a. URL https://arxiv.org/abs/2201.11903. Introduces the Chain-of-Thought prompting method to improve reasoning in large LMs.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022b. URL https://arxiv.org/abs/2201.11903.

Yuyang Wu, Yifei Wang, Ziyu Ye, Tianqi Du, Stefanie Jegelka, and Yisen Wang. When more is less: Understanding chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*, 2025.

Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie Wang, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas Griffiths, Yuandong Tian, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2024.

Wayne Zhao, Haoyu Zhang, Kai Zhang, Wayne Xin Zhou, and Danqi Chen. Attention sink: Do transformer attention heads have long-tail redundancy? *arXiv preprint arXiv:2309.17476*, 2023.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2205.10625. Preprint, arXiv:2205.10625.

A APPENDIX

APPENDIX CONTENTS

A.1 Related Work

Enhancing Chain-of-Thought (CoT) Since *Chain-of-Thought* (CoT) prompting was popularized for step-wise reasoning, a large body of work has explored how to elicit, structure, and aggregate intermediate thoughts. This work usually applies to both non-reasoning models and reasoning models. Zero-shot and few-shot CoT show that lightweight hints can bootstrap decomposition without heavy finetuning (Kojima et al., 2022; Wei et al., 2022b). *Self-Consistency* (SC) improves reliability by sampling diverse reasoning paths and voting on answers (Wang et al., 2023). Beyond linear traces, researchers proposed richer structures: *Tree-of-Thoughts* (ToT) performs breadth/depth exploration with value-guided selection (Yao et al., 2024), and *Graph-of-Thoughts* connects partial solutions and subproblems in a general graph to enable reuse and recombination (Besta et al., 2024). Task-decomposition templates such as *Least-to-Most* and complexity-aware prompting steer models toward simpler sub-goals (Zhou et al., 2022; Fu et al., 2023).

Enhancing LLM Reasoning Capability Beyond formatting thoughts, several threads aim to improve inference time reasoning itself, often referred to as Test Time Scaling. This line of work usually applies to model with reasoning capabilities. Test-time compute scaling strategies increase sample count, depth, or verification steps, often coupled with selective aggregation and verification to filter inconsistent traces (Cobbe et al., 2021; Dhuliawala et al., 2024). COTA encourages multiple complementary perspectives before convergence (Pan et al., 2025). Other studies probe how far we can push "small" models with better training, curricula, and targeted heuristics, sometimes surpassing naively prompted much larger models (Liu et al., 2025).

At the same time, multiple papers warn that "more thinking" can backfire: both controlled studies and empirical audits document overthinking and propose inference-time policies to cap or adapt CoT length (Chen et al., 2024; Cuadron et al., 2025). Wu et al. (2025) uncover an inverted-U relation between CoT length and accuracy, with optimal lengths growing with task difficulty but shrinking with model capability; RL training further biases models toward shorter, more efficient chains (Wu et al., 2025). Complementary work finds that excessively long CoTs can harm performance on simple instances, cautioning against unbounded test-time scaling (Chen et al., 2024; Cuadron et al., 2025; Xia et al., 2025).

Attention Attribution Recent work has examined whether attention weights can serve as faithful explanations, but evidence shows that raw attention distributions are often unreliable and can be manipulated without changing predictions (Serrano & Smith, 2019; Jain & Wallace, 2019). Studies of BERT and related models reveal recurring head patterns linked to syntax and coreference, while many heads appear redundant or dispensable (Clark et al., 2019; Voita et al., 2019; Michel et al., 2019). To better capture information flow, techniques such as attention rollout and relevance propagation have been proposed to trace or aggregate attention through layers, yielding more faithful token-level attributions (Abnar & Zuidema, 2020; Chefer et al., 2021). In parallel, probing methods such as classifier probes, edge probing or structural probes, demonstrate that specific layers encode linguistic features in a pipeline-like progression (Hewitt & Manning, 2019; Tenney et al., 2019). More recently, researchers have discovered "sink tokens" that absorb disproportionate attention mass, leading to diluted signals across depth and biased rollout analyses, with mitigation strategies explored in both language and vision Transformers (Kobayashi et al., 2020; Dong et al., 2021; Sun et al., 2022).

A.2 Experiment Setup

A.2.1 Datasets and task taxonomy.

To comprehensively evaluate reasoning across diverse domains, we group benchmarks by the primary reasoning skill they target. This taxonomy spans logical, arithmetic, commonsense, and symbolic reasoning, ensuring that our analysis captures a wide spectrum of cognitive capabilities.

- **Logical reasoning: GPQA-Diamond** (Rein et al., 2023). GPQA-Diamond is a recently introduced benchmark consisting of graduate-level, Google-proof questions designed to probe advanced logical reasoning. The dataset emphasizes multi-hop reasoning and deductive inference, requiring models to integrate evidence across multiple statements before arriving at a conclusion. Unlike typical factual QA benchmarks, GPQA-Diamond minimizes reliance on web-searchable facts, thereby isolating genuine reasoning ability from memorization.
- **Arithmetic reasoning: MATH500 and AIME24** (Hendrycks et al., 2021; Lightman et al., 2023; Jia, 2024). The arithmetic category evaluates the capacity of models to manipulate numbers, apply mathematical formulas, and perform multi-step computations. MATH500 is a curated subset of 500 problems sampled from the broader MATH dataset (Hendrycks et al., 2021), covering algebra, geometry, probability, and other competition-style mathematics. AIME24 is a collection of problems from the 2024 American Invitational Mathematics Examination, one of the most challenging pre-collegiate contests. The questions are designed to require multi-step derivations and precise symbolic manipulation, often beyond simple arithmetic.
- **Commonsense reasoning: StrategyQA** (Geva et al., 2021). StrategyQA is a benchmark focusing on implicit commonsense reasoning. Unlike direct QA datasets, the questions often cannot be answered with a single fact but instead require models to generate a strategy for combining multiple weak signals (e.g., "Could a crocodile run a marathon?"). This dataset tests a model's ability to operationalize background knowledge and apply everyday reasoning skills in novel contexts.
- **Symbolic reasoning: CoinFlip and Last Letter** (Wei et al., 2022a). These synthetic tasks were introduced to stress-test chain-of-thought prompting. In CoinFlip, the model must track the state of a coin after a sequence of flips, which requires deterministic state updates over potentially long sequences. In Last Letter, the model must return the last letter of each word in a given string and concatenate them. Despite their apparent simplicity, these tasks are challenging for LLMs, as they require precise symbolic manipulation, memory of intermediate steps, and robustness to input length.

Together, these datasets form a comprehensive suite that spans from formal logic and competition-level mathematics to real-world commonsense and strict symbolic reasoning. This breadth allows us to evaluate whether improvements from our proposed method generalize across fundamentally different reasoning paradigms. For ablation study, we select the first 50 questions for GPQA-Diamond as testing dataset.

A.2.2 Hyper Parameters and Additional Details

For the max token length, we follow the practice of Liu et al. (2025) to set the max token to be 8192 for all the models except for LLaMA-3.1-8B, where we set the max token to 4096. For the sampling process, we set the max token uniformly to 4096.

For smaller modes, these set up has cause some of the questions not to be answered as the thinking process do not finish. For such cases, we exclude them from the calculation of the final accuracy as we consider this as model limitation, not the limitation of our methods.

We set the temperature to be 0.7 throughout all the models in all the generation process for both our method and baselines.

We consider punctuations, special tokens and stop words from nltk as non-semantic tokens.

A.3 THE MIT METHODOLOGY ALGORITHM

For the full procedures for the MIT method, please refer to the algorithm below.

Algorithm 2 Metacognitive Influence Tracing (MIT)

Require: Raw attention matrices $\{A_{l,h}\}_{l=1..L,\, h=1..H}$ with $A_{l,h} \in \mathbb{R}^{K\times K}$, token sequence T, answer token indices T_{ans}, CoT token indices T_{CoT}, diffusion scale $\gamma > 0$

procedure MIT($\{A_{l,h}\}, T, T_{\text{ans}}, T_{\text{CoT}}$)
// Stage 1: Attention Signal Purification
  Initialize purified set $\{A'_{l,h}\} \leftarrow \varnothing$
  for $l = 1$ to L **do**
    for $h = 1$ to H **do**
      $A_{\text{temp}} \leftarrow A_{l,h}$
      $A_{\text{temp}} \leftarrow \text{FILTERSEMANTICTOKENS}(A_{\text{temp}}, T)$
      $\text{diag}(A_{\text{temp}}) \leftarrow \mathbf{0}$ ▷ remove self-attention
      Add A_{temp} to $\{A'_{l,h}\}$
    end for
  end for
// Stage 2: Layer Aggregation
  Initialize $A'' \in \mathbb{R}^{H\times K\times K}$ with zeros
  for $h = 1$ to H **do**
    $A''_h \leftarrow \frac{1}{L}\sum_{l=1}^{L} A'_{l,h}$
  end for
// Stage 3: Kurtosis-Weighted Head Aggregation
  Initialize $A \in \mathbb{R}^{K\times K}$ with zeros
  for $i = 1$ to K **do** ▷ target token index
    $w \in \mathbb{R}^{H}$ ▷ per-head weights for row i
    for $h = 1$ to H **do**
      $w_h \leftarrow \text{Kurtosis}(A''_h[i,:])$
    end for
    $w \leftarrow w / \sum_{h=1}^{H} w_h$ ▷ normalize
    for $j = 1$ to K **do** ▷ source token index
      $A[i,j] \leftarrow \sum_{h=1}^{H} w_h \cdot A''_h[i,j]$
    end for
  end for
// Stage 4: Influence Propagation
  $S \leftarrow \exp(\gamma A)$ ▷ matrix exponential diffusion over token graph
// Stage 5: Critical Juncture Identification
  Initialize $\text{Score} \in \mathbb{R}^{|T_{\text{CoT}}|}$ with zeros
  for each token index $j \in T_{\text{CoT}}$ **do**
    $\text{Score}(j) \leftarrow \sum_{i\in T_{\text{ans}}} S[i,j]$
  end for
  $j_{\text{critical}} \leftarrow \arg\max_{j\in T_{\text{CoT}}} \text{Score}(j)$
  return j_{critical}
end procedure

A.4 Propagation Kernels

A.4.1 Theoretical Analysis

To apply the principle of propagation, we model the reasoning trace as a directed graph and propose that the total, propagated influence can be measured by a diffusion process on this graph. Formally, this leads us to define a family of *diffusion kernels*, or equivalently *propagation operators*, parameterized by different matrix functions of the attention matrix $A \in \mathbb{R}^{N\times N}$. Each kernel describes how information diffuses across multiple hops, ranging from purely local propagation to global mixing, with different mathematical properties and constraints.

First Order Kernel The most basic form is the first-order kernel, which corresponds to one-step diffusion:

$$K_{\text{1-hop}} = A. \tag{7}$$

This operator captures only direct interactions between tokens, reflecting immediate influence between neighbors. While computationally efficient and stable, it cannot capture multi-hop dependencies.

Second Order Kernel To extend beyond immediate neighbors, we may incorporate two-step diffusion via the second-order kernel:

$$K_{\text{2-hop}} = A + A^2. \tag{8}$$

Here the term A^2 encodes influence transmitted through a single intermediary, while the additive form balances first- and second-order contributions. This kernel thus expands the local receptive field, but still truncates higher-order paths.

Resolvent Diffusion Kernel To model paths of arbitrary length, we consider the *resolvent diffusion kernel*, also known as the Green's function kernel, given by the Neumann series:

$$K_{\text{resolvent}} = \sum_{k=1}^{\infty} A^k = A(I - A)^{-1}, \tag{9}$$

which converges when the spectral radius $\rho(A) < 1$. This constraint ensures stability but may be too restrictive in practice. A damping parameter $\alpha \in (0, 1)$ can be introduced:

$$K_{\text{resolvent},\alpha} = \sum_{k=1}^{\infty} (\alpha A)^k = \alpha A(I - \alpha A)^{-1}, \tag{10}$$

which guarantees convergence for all $\alpha < 1/\rho(A)$. The resolvent kernel assigns equal weight to paths of all lengths (modulo damping), allowing very long walks to contribute substantially. While expressive, this sensitivity to the spectrum of A and to the choice of α must be carefully controlled.

Heat Kernel An alternative diffusion kernel is the *heat kernel*, defined via the matrix exponential with a damping factor $\gamma > 0$:

$$K_{\text{heat}}(\gamma) = \exp(\gamma A) = I + \gamma A + \frac{(\gamma A)^2}{2!} + \frac{(\gamma A)^3}{3!} + \cdots. \tag{11}$$

Unlike the resolvent kernel, the heat kernel includes paths of all lengths but discounts them by $\gamma^k/k!$, strongly suppressing long walks. This factorial decay emphasizes short- and medium-range interactions. Moreover, the exponential series converges unconditionally for any square matrix A, making it numerically stable and well-defined regardless of spectral radius.

The spectral interpretation highlights these differences. If $A = Q\Lambda Q^{-1}$ with eigenvalues $\{\lambda_i\}$, then

$$K_{\text{heat}}(\gamma) = Q\,\text{diag}(e^{\gamma\lambda_1}, \ldots, e^{\gamma\lambda_N})Q^{-1}. \tag{12}$$

Thus the damping parameter γ acts as a nonlinear spectral filter: small values yield a local approximation close to $I + \gamma A$, while larger values amplify the contributions of dominant eigenmodes, enhancing global propagation. In contrast, the resolvent kernel corresponds to a rational spectral filter:

$$K_{\text{resolvent},\alpha} = Q\,\text{diag}\left(\frac{\alpha\lambda_1}{1-\alpha\lambda_1}, \ldots, \frac{\alpha\lambda_N}{1-\alpha\lambda_N}\right)Q^{-1}, \tag{13}$$

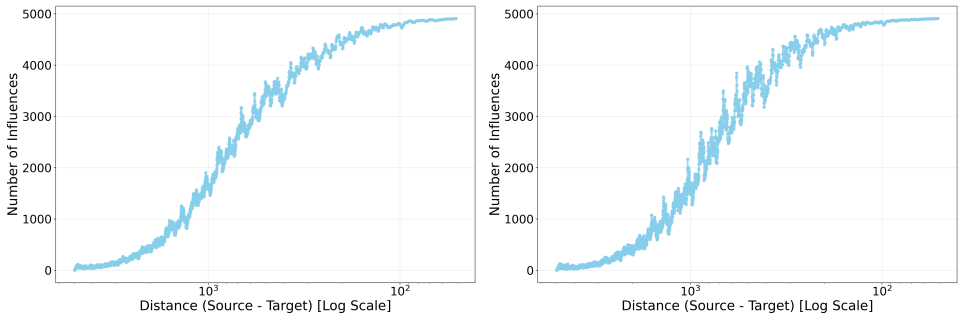
which assigns equal geometric weight to paths of different lengths, modulated only by damping.

In summary, diffusion kernels provide a principled way to formalize propagation in attention. The first- and second-order kernels are local and stable but limited in scope. The resolvent kernel allows arbitrarily long-range interactions but requires spectral constraints and careful damping. The heat kernel is stable for all matrices, tunable through γ, and naturally emphasizes moderate path lengths via factorial discounting. This diffusion perspective unifies multiple propagation kernels and clarifies their trade-offs in balancing locality, stability, and expressivity.

A.4.2 Empirical Analysis

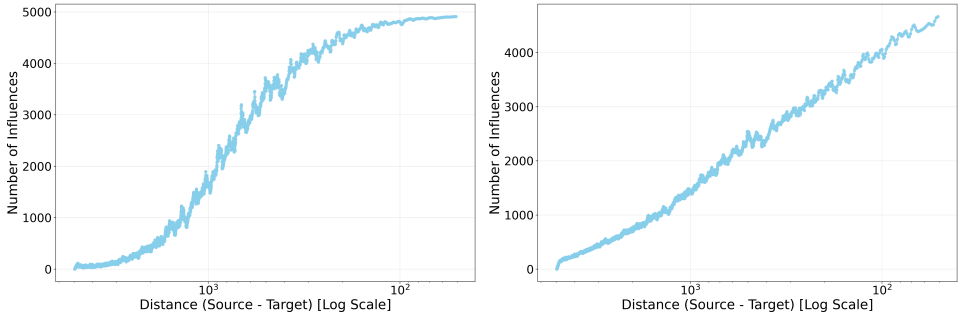
Reasoning Model We also conducted empirical comparison to compensate our theoretical analysis. We start by studying the impact of different kernels on the reasoning model. Using DeepSeek-Distilled- Qwen-1.5B model on the problem gpqa-diamond–0020, we want to study how different kernels impact the most influential token for each token within a generated trace.

We first identify the most influential token for each token (target token) in the propagated attention matrix, then we group them by the distance between the most influential token and the target token. This help us evaluate the diffusion process spread across the graph.



(a) Influence Count by Distance - Heat Kernel

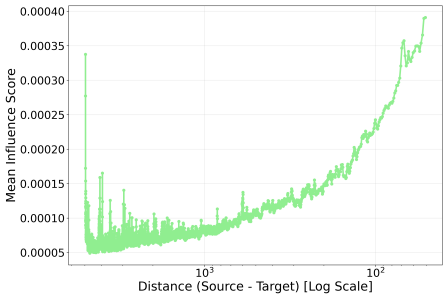
(b) Influence Count by Distance - First Order Kernel



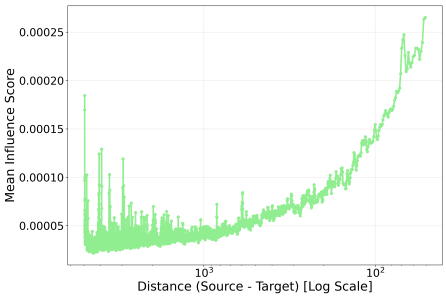
(c) Influence Count by Distance - Second Order Kernel

(d) Influence Count by Distance - Resolvent Kernel

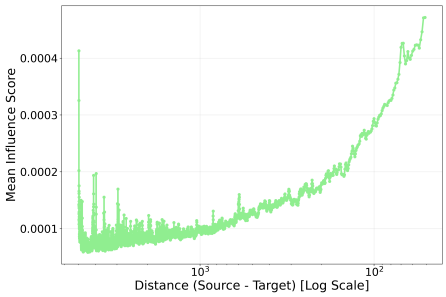
Figure 5: Influential Count Study for Propagation Kernels. This plot shows the distribution of distances between the most influential token and the corresponding target token. The curve exhibits a relatively smooth trend, with the majority of critical influences concentrated within a distance of approximately 100 tokens, after which the influence count gradually saturates with occasional jumps.



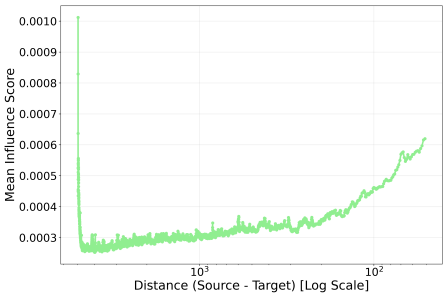
(a) Mean Score by Distance - Heat Kernel



(b) Mean Score by Distance - First Order Kernel

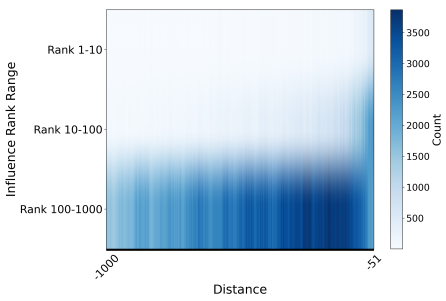


(c) Mean Score by Distance - Second Order Kernel

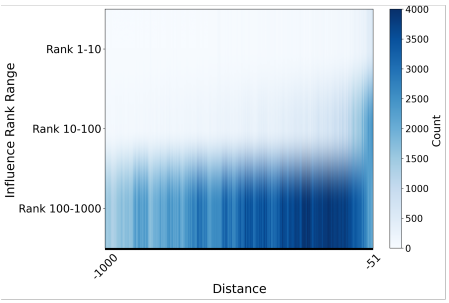


(d) Mean Score by Distance - Resolvent Kernel

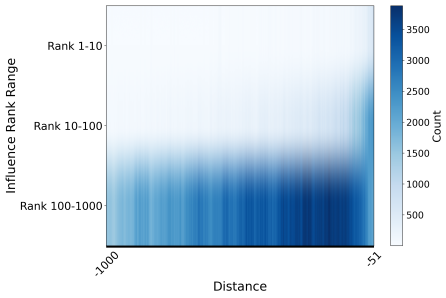
Figure 6: Mean Score Study for Propagation Kernels. This plot shows the distribution of influential score based on distances. We observe a spike in score when the distance is really far. We hypothesize that this resembles the attention sink pattern.



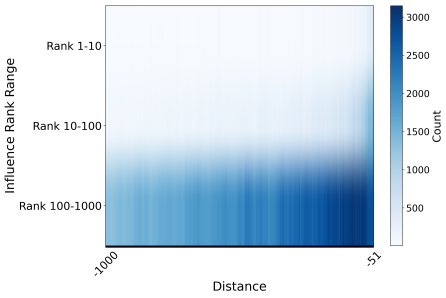
(a) Rank Distribution by Distance – Heat Kernel



(b) Rank Distribution by Distance – First Order Kernel

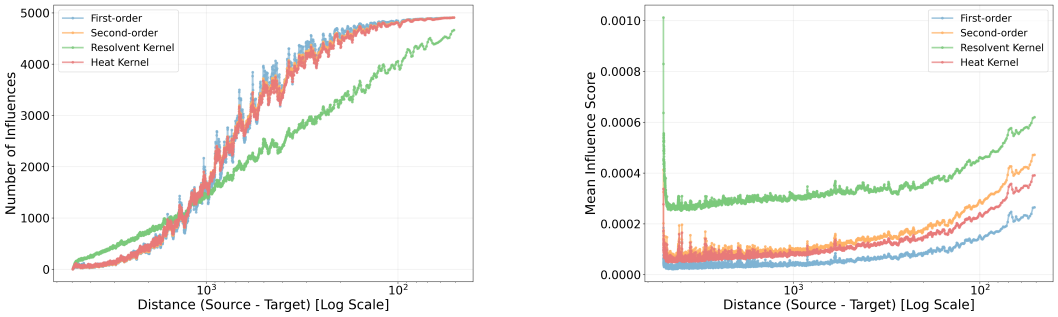


(c) Rank Distribution by Distance – Second Order Kernel



(d) Rank Distribution by Distance – Resolvent Kernel

Figure 7: Rank Distribution Study for Propagation Kernels. For Resolvent Kernel where paths of different distances are considered equally, we observe a blurring effect where details are lost and hypothesis that it is the cause of the relatively low performance compared to the Heat Kernel.



(a) Kernels Influential Count Comparison Analysis (b) Kernels Influential Score Comparison Analysis

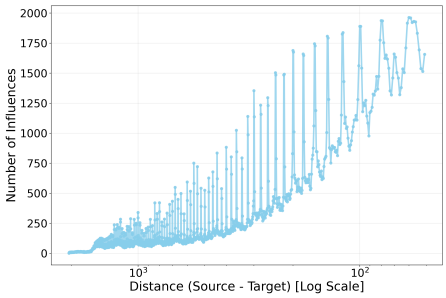
Figure 8: Influential Count and Score Comparison. We observe similar trends in both model analysis. First order kernel, second order kernel and heat kernel are relatively closer to each other whereas resolvent kernel demonstrates some distance.

Non-Reasoning Model We follow the same strategy and test it on non-reasoning model - LLaMA-3.1-8B on the same question. Overall we observe similar trends between different kernels and we also observe some interesting behavior for LLaMA that we did not see in tests in reasoning models.

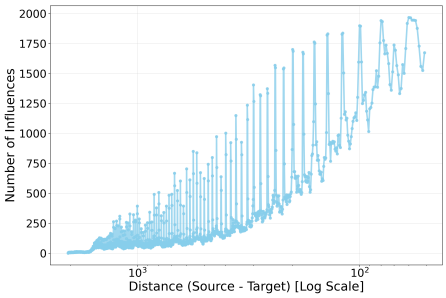
First observation is that compared to reasoning models, non-reasoning model does not show as much distinction between Resolvent kernel as the other kernels.

Also for non-reasoning models, we observe a regular spike in both influence count by distance, and the mean score by distance. We hypothesize that this is caused by the model dividing tokens into buckets with cognitive markers, and use such markers, for easier state managements.

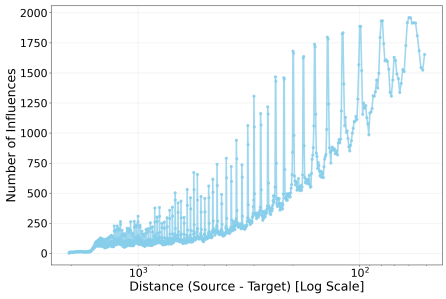
We also observe a U shape in mean score by distance, where the mean score first goes down as distance increases, however it gradually goes back up when distance continuously increases.



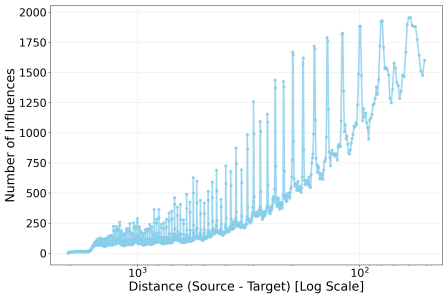
(a) Influence Count by Distance - Heat Kernel



(b) Influence Count by Distance - First Order Kernel

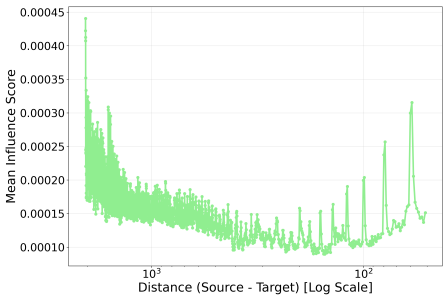


(c) Influence Count by Distance - Second Order Kernel

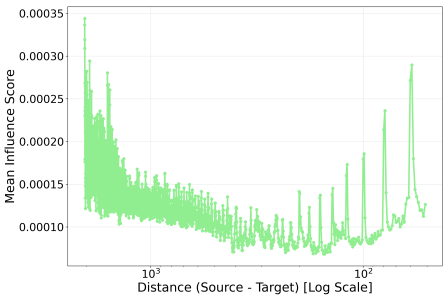


(d) Influence Count by Distance - Resolvent Kernel

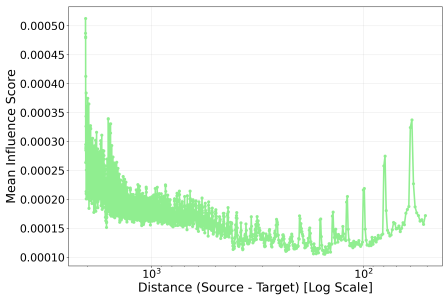
Figure 9: Influential Count Study for Propagation Kernels. All kernels follow a similar trend. We also observe for LLaMA there are occasional spikes repeatedly occurring through out the distance range.



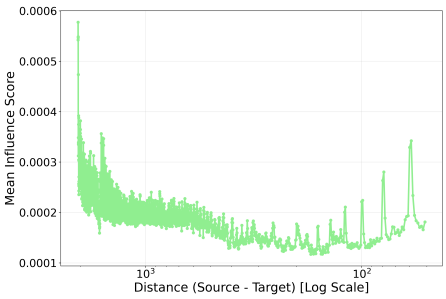
(a) Mean Score by Distance - Heat Kernel



(b) Mean Score by Distance - First Order Kernel

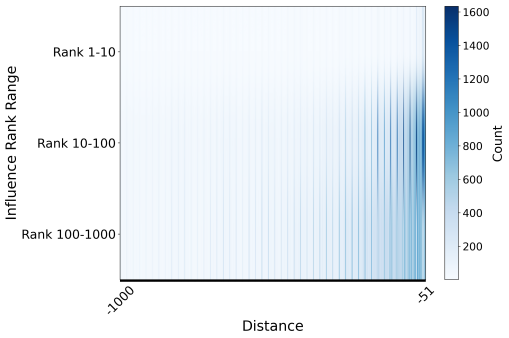


(c) Mean Score by Distance - Second Order Kernel

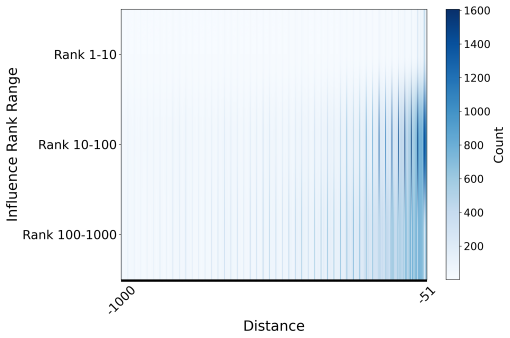


(d) Mean Score by Distance - Resolvent Kernel

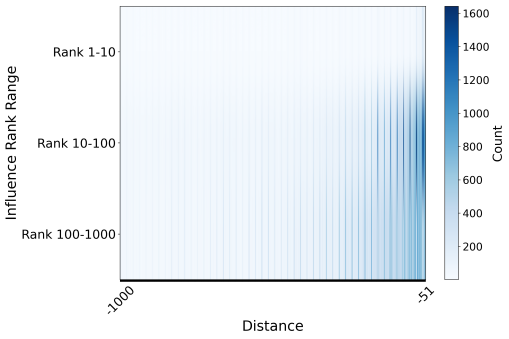
Figure 10: Mean Score Study for Propagation Methods. We notice an U shape in all kernels, where the mean score first goes down as distance increases, but goes up when the distance is really big. A spike in the far distance is also observed in reasoning models in Figure 6, however for non-reasoning models the pivot point happens way earlier and the trend going up is more smooth.



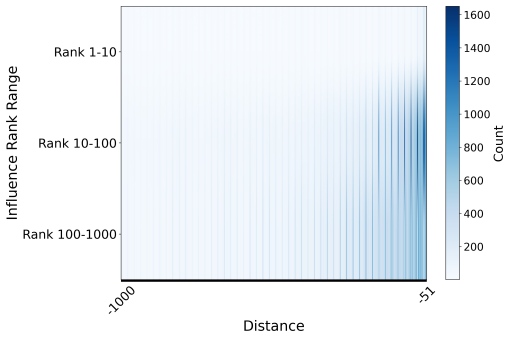
(a) Rank Distribution by Distance - Heat Kernel



(b) Rank Distribution by Distance - First Order Kernel

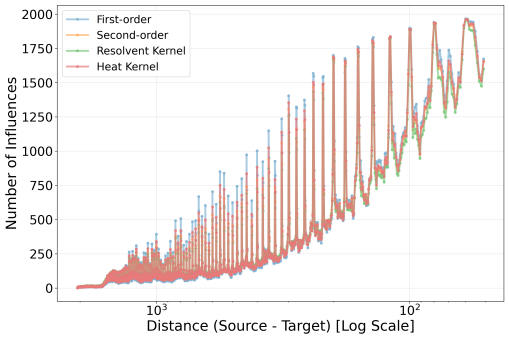


(c) Rank Distribution by Distance - Second Order Kernel

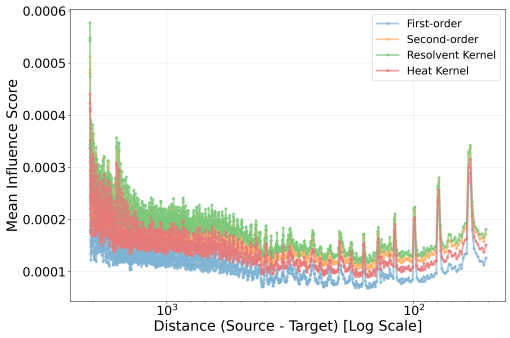


(d) Rank Distribution Distance - Resolvent Kernel

Figure 11: Rank Distribution Study for Propagation Methods. All kernels exhibit similar patterns. The blurring effect in non-reasoning models is not as strong as that of reasoning models in Figure 7.



(a) Reasoning Model Strategy Analysis



(b) Non-Reasoning Model Strategy Analysis

Figure 12: Influential Count and Score Comparison. Generally speaking, compared to reasoning models, different kernels follow a much more similar patterns for non-reasoning models.

A.5 PROPAGATION PARAMETER

Selecting the Propagation Parameter γ To better understand the influence of the exponential decay parameter γ, we visualized the rank distribution across a range of values: $\gamma \in \{1, 2, 3, 5, 8, 13\}$. As shown in Figure 13, smaller values of γ (e.g., 1 and 2) emphasize higher influence concentration in the mid- to high-ranked tokens (i.e., rank 100–1000) in the distance between 50 to 1000, while preserving noticeable gradients across distance. In contrast, higher values of γ (e.g., 5, 8 or 13) over-emphasize long-distance influence, resulting in overly flat and uniform rank maps between the distance 50 to 1000, indicative of diminished propagation power. Based on these insights, we selected $\gamma = 2$ as the default for exponential propagation, as it provides a good trade-off between sparsity and depth-aware diffusion.

This choice is further supported by the ablation results in Figure 14. For non-reasoning models, performance consistently declines as γ increases beyond 2, while for reasoning models we observe an unexpected spike at $\gamma = 13$. These differences suggest that reasoning and non-reasoning models occupy distinct optimization regimes, and that the optimal γ may depend on the model's reasoning style rather than a single universal setting.

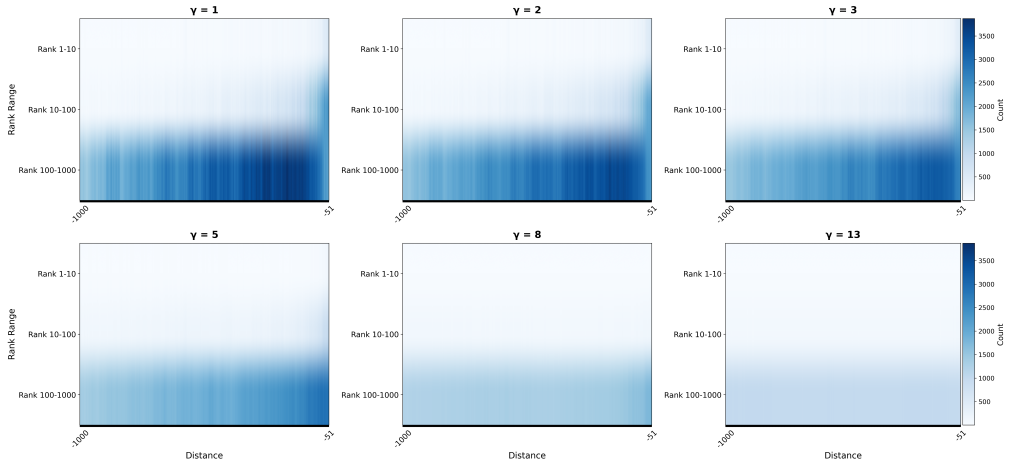


Figure 13: Rank distribution across different exponential decay values ($\gamma \in \{1, 2, 3, 5, 8, 13\}$).

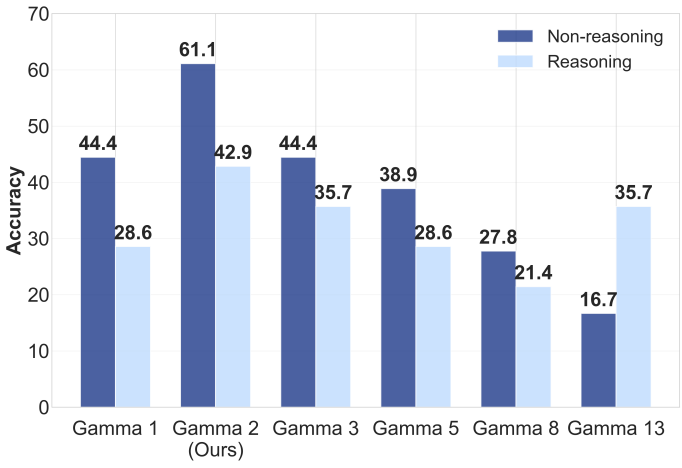


Figure 14: ACS performance under different gamma values: ($\gamma \in \{1, 2, 3, 5, 8, 13\}$)

A.6 LAYER WISE AGGREGATION ANALYSIS

We conducted a layer-wise aggregation analysis to study the impact of different methods on information propagation across layers, for both reasoning and non-reasoning models. Our findings are consistent across model types, and we summarize the key insights as follows:

- **Cleaning** improves interpretability by isolating semantically meaningful dependencies, allowing logical influence patterns to emerge more clearly.
- **Average** and **Average Decay** aggregation strategies highlight fine-grained distinctions across layers, enabling the emergence of more nuanced, structured patterns.

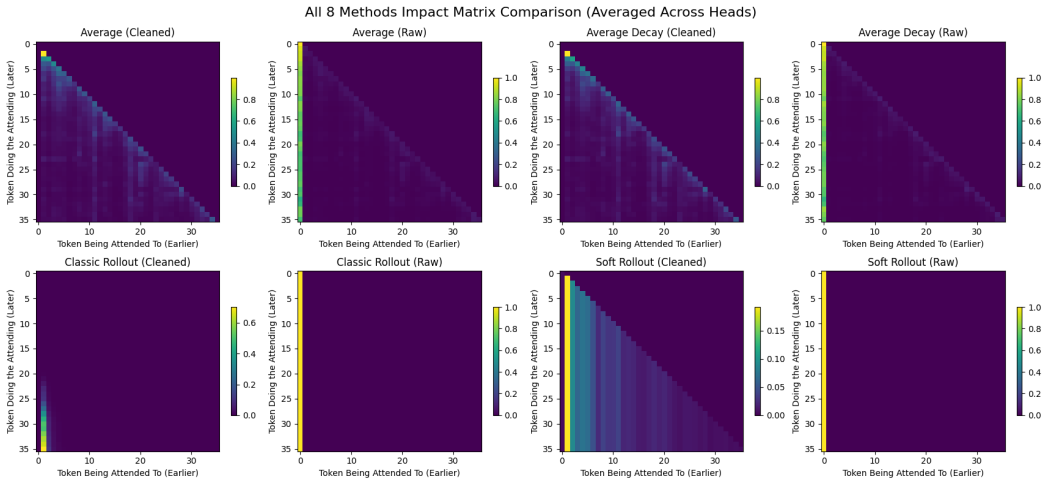


Figure 15: Layer-wise Aggregation Analysis (LLaMA-3.1-8B)

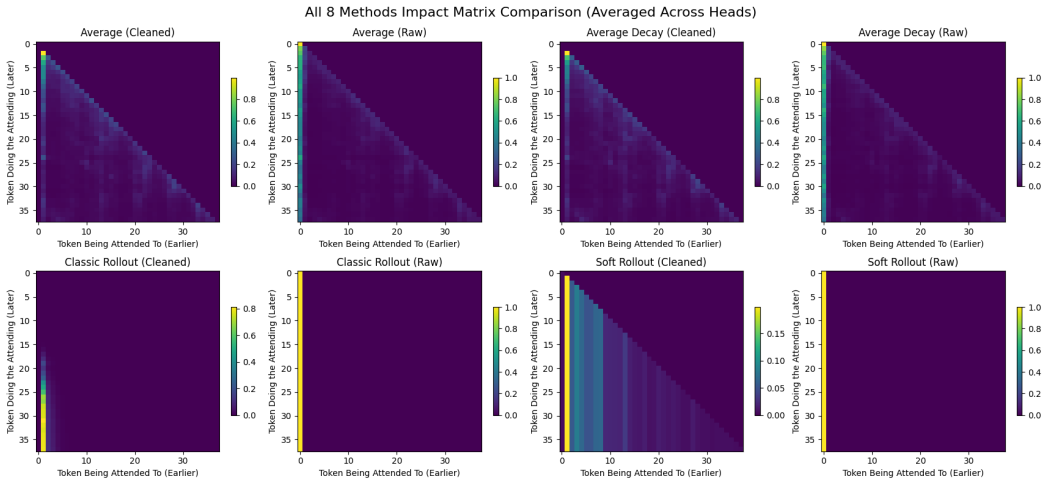


Figure 16: Layer-wise Aggregation Analysis (DeepSeek-Distilled-Qwen-1.5B)

A.7 Critical Token Study

A.7.1 Cognitive Marker Distribution

We study the vocabulary of reasoning and non-reasoning models that are surfaced by `MIT` , and present their distributions grouped by model in Figure 17. Across both reasoning and non-reasoning models, we observe consistent reliance on a small set of cognitive markers. Words such as *"okay"* and *"step"* appear as dominant tokens across multiple models, highlighting their role as universal markers of process control and explicit reasoning progression.

Interestingly, the *SmoLM3-3B (Thinking)* model exhibits a strikingly uniform pattern: all critical junctures are marked by the token *"okay"*, which is also found in the reasoning-oriented *DeepSeek-Distilled-Qwen-1.5B*. This suggests that these models adopt a highly cognitive style of reasoning, repeatedly signaling checkpoints of internal validation. However the particular word used differ from model and can be due to the training process of each model such as SmoLM3-3B (Thinking) prefer *"okay"* whereas DeepSeek-Distilled-Qwen-1.5B prefers *"wait"*.

By contrast, non-reasoning models such as *Gemma-3-270M-IT* and *LLaMA-3.1-8B* reveal more heterogeneous distributions (Figure 17). Tokens such as *"answer"*, *"therefore"* surface as frequent critical junctures, reflecting a more linguistically diverse and procedural reasoning style. The particular word *"step"* is given in the prompt, so it might not be consistent when prompt changes. But other word that connects reasoning together such as *"therefore"*, *"because"*, *"given"* do not present within the prompts.

These findings suggest that reasoning models converge toward concentrated cognitive markers to signal decision checkpoints, while non-reasoning models spread cognitive load across diverse linguistic markers. This contrast highlights a fundamental difference in how model classes manage and externalize their reasoning processes.

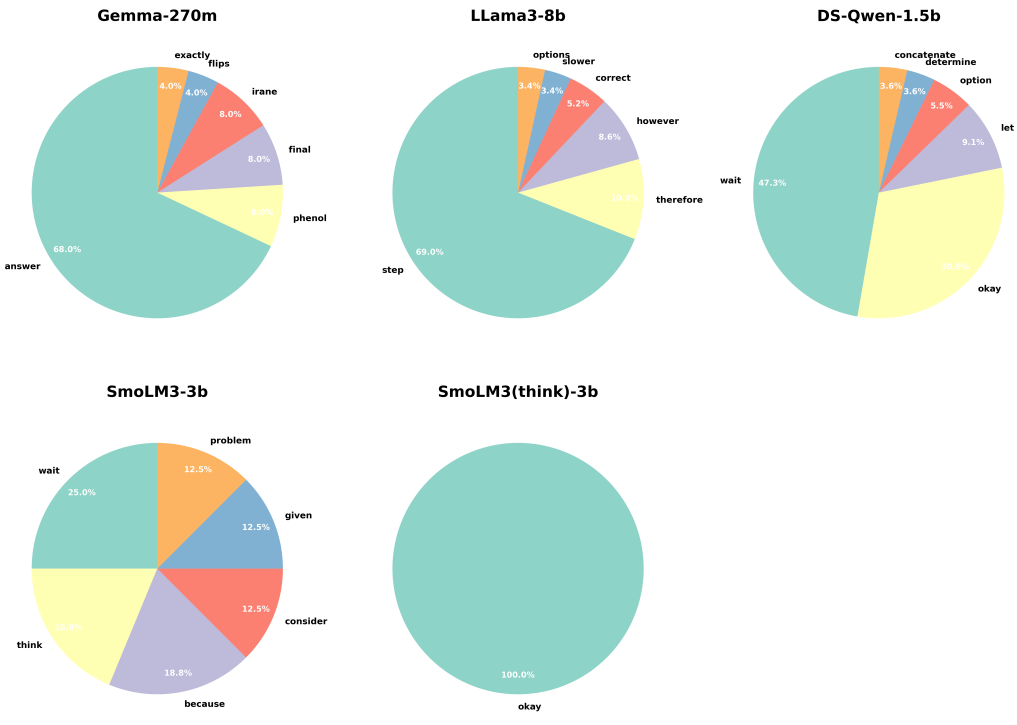


Figure 17: The distribution of cognitive marker surfaced as critical junctures by models

A.7.2 Critical Token Resample Word Study

Figures 18 and 19 compare the tokens surfaced by `MIT` at critical junctures with the first tokens generated during resampling. Overall, the vocabulary does not change substantially: dominant cognitive markers such as *"okay"* and *"wait"* continue to appear frequently after resampling. Likewise, procedural markers such as *"step"* and *"final"* remain stable across both stages. However, especially for reasoning model, we notice the first token generated during resampling is much more diverse compared to the critical juncture token surfaced by `MIT`. This supports our hypothesis that critical junctures are gateways towards high-variance regions, which resample allows more candidates to be created, leading to a diverse vocabulary.

.

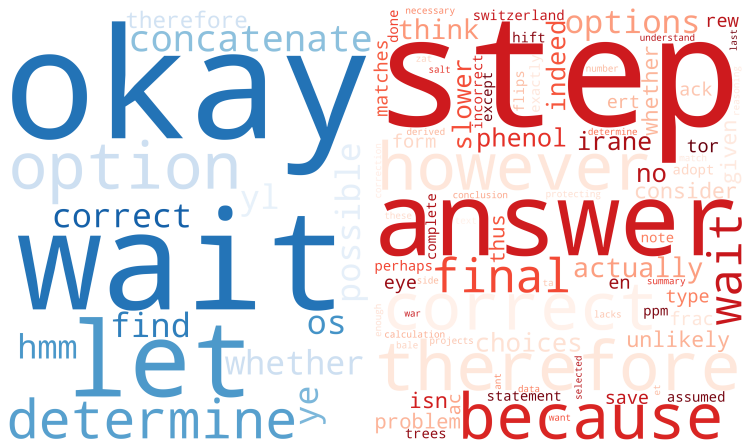


Figure 18: The word cloud for the critical token. Reasoning model on the left in blue, non-reasoning model on the right in red.

Figure 19: The word cloud for the sampled first token. Reasoning model on the left in blue, non-reasoning model on the right in red.

A.7.3 MOST INFLUENTIAL TOKEN DISTANCE STUDY

We study the distance between the critical token surfaced by `MIT` and the start of the answer token to gauge the diffusion scope `MIT` provides. As shown in Figure 20, our analysis reveals distinct cognitive patterns across different language models in terms of how far their most influential tokens are positioned from the final answer. Generally reasoning model has exhibit longer range. We hypothesize this capability allows it to understand previous trace and lead to better performance.

For non-reasoning models, Gemma-3-270M-IT demonstrates the most localized reasoning strategy, suggesting it relies heavily on immediate contextual cues near the answer location. LLaMA-3.1-8B and SmolLM3-3B exhibit similar distributed reasoning patterns with median distances of 74.5 and 77 tokens respectively, though their high standard deviations (1,534 and 2,462 tokens) indicate significant variability in their reasoning strategies.

For reasoning models, DeepSeek-Distilled-Qwen-1.5B has median distance of 492 tokens, showing more structured long-range reasoning than the non-reasoning models but remaining more focused than the bigger reasoning models. SmolLM3-3B(Think), the specialized reasoning variant, shows the most systematic pattern with a median distance of 780 tokens—demonstrating that explicit reasoning training encourages models to establish critical decision points much earlier in the thought process. The relatively low variance in SmolLM3-3B(Think) (502 tokens) compared to its base model suggests that reasoning-trained models develop more consistent cognitive strategies, while non-reasoning models exhibit more unpredictable patterns depending on the specific question type.

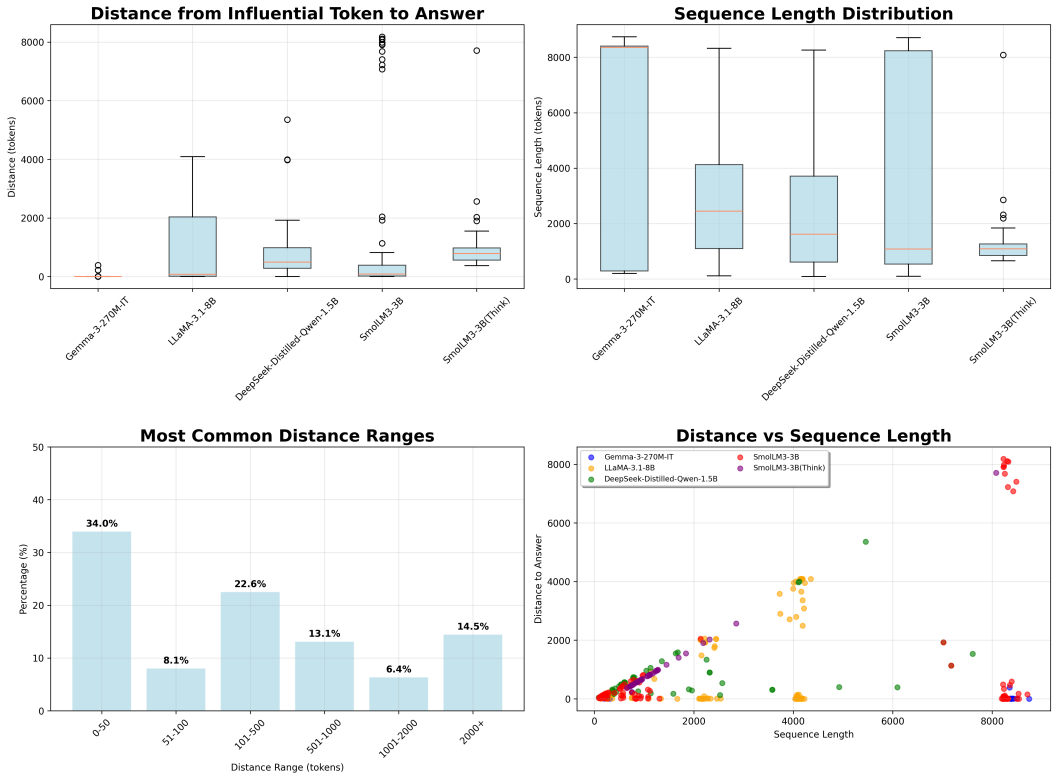


Figure 20: The distance between the surfaced critical token by `MIT` and the start of the answer token. Our analysis reveals distinct cognitive patterns across different language models in terms of how far their most influential tokens are positioned from the final answer. Generally reasoning model has exhibit longer range. We hypothesize this capability allows it to understand previous trace and lead to better performance.

A.8 COST CALCULATION

To provide a fair and rigorous comparison between Self-Consistency (SC) and our proposed Adaptive Critical Sampling (ACS), we must account for the full computational cost of each method. We propose using **Floating-Point Operations (FLOPs)** as a unified currency to measure both LLM inference and the numerical computations within MIT.

The total cost for any method can be expressed as:

$$\text{TotalCost}_{\text{method}} = \text{Cost}_{\text{diagnostic}} + \text{Cost}_{\text{generation}} \tag{14}$$

For Self-Consistency, $\text{Cost}_{\text{diagnostic}}$ is zero. For ACS, it is the cost of running the MIT algorithm.

A.8.1 COST OF LLM GENERATION

The computational cost of autoregressive generation from a transformer model is dominated by the forward passes. A common approximation for the FLOPs required for a single forward pass is $2 \times P \times T$, where P is the number of model parameters and T is the sequence length. Since generation is sequential, the total cost for a sequence of length T_{total} is the sum of the costs for generating each token:

$$\text{Cost}_{\text{gen}}(T_{\text{total}}) \approx \sum_{i=1}^{T_{\text{total}}} 2 \times P \times i \approx P \times T_{\text{total}}^2 \tag{15}$$

A.8.2 COST OF THE MIT DIAGNOSTIC

The cost of the MIT diagnostic is non-trivial and can be broken down by its most computationally intensive stages, operating on a sequence of length K:

- **Aggregation (Stages 1-3):** These stages involve matrix manipulations across all layers (L) and heads (H) for the $K \times K$ attention matrices. The cost is roughly proportional to the square of the sequence length:
$$\text{Cost}_{\text{agg}} \propto L \times H \times K^2 \tag{16}$$
- **Propagation (Stage 4):** This is the primary computational bottleneck. It requires calculating the matrix exponential of the $K \times K$ influence matrix. The complexity of this numerical operation is approximately cubic in the matrix dimension:
$$\text{Cost}_{\text{prop}} \propto K^3 \tag{17}$$

The total cost of MIT is therefore dominated by the propagation step, scaling cubically with the length of the reasoning trace.

$$\text{Cost}_{\text{MIT}}(K) \approx c_1(L \times H \times K^2) + c_2(K^3) \tag{18}$$

where c_1 and c_2 are machine-dependent constants and we set c_1 to be 8 and c_2 to be 40 base on benchmarking on aws ml.g5.48xlarge instance.

A.8.3 PROPOSED COST COMPARISON FRAMEWORK

Using this framework, we can define the total cost for generating N samples for each method. Let T_{full} be the length of a full reasoning trace and T_{suffix} be the length of the trace generated after a critical juncture.

The total cost for **Self-Consistency** is:

$$\text{TotalCost}_{\text{SC}} = N \times \text{Cost}_{\text{gen}}(T_{\text{full}}) \tag{19}$$

The total cost for **Adaptive Critical Sampling** is:

$$\text{TotalCost}_{\text{ACS}} = \underbrace{\text{Cost}_{\text{gen}}(T_{\text{full}})}_{\text{Initial Trace}} + \underbrace{\text{Cost}_{\text{MIT}}(T_{\text{full}})}_{\text{Diagnostic Overhead}} + \underbrace{(N-1) \times \text{Cost}_{\text{gen}}(T_{\text{suffix}})}_{\text{Resampling Suffixes}} \tag{20}$$

This framework provides a principled and transparent methodology for evaluating the true efficiency trade-offs, revealing under which conditions (e.g., long vs. short reasoning traces) the diagnostic overhead of MIT is justified by the savings in generation cost.

A.8.4 THEORETICAL COST ANALYSIS

In practice, the apparent cubic overhead of MIT should be understood in relation to the dominant cost of autoregressive generation. While $\text{Cost}_{\text{MIT}}(K)$ scales as K^3, the generation cost scales as $P \times T^2$, where P is the number of model parameters. For modern LLMs with P in the billions and context lengths in the thousands, $P \gg K$, meaning the per-token matrix multiplications in generation dominate the computational bill. As a result, the one-time K^3 diagnostic is negligible compared to repeatedly multiplying against billions of parameters during token generation. Put differently, the crossover point where $\text{Cost}_{\text{MIT}}(K)$ becomes comparable to Cost_{gen} only arises for very small models (tens to hundreds of millions of parameters) or extremely long traces. For the parameter scales where self-consistency is typically applied (multi-billion parameter LLMs), MIT is therefore asymptotically and empirically cheaper than regenerating full traces, which explains the consistent efficiency gains of ACS over SC.

For example, consider $K = 2048$ tokens and a model with $P = 7 \times 10^9$ parameters. The diagnostic propagation step requires on the order of $K^3 \approx 8.6 \times 10^9$ operations. By contrast, generating a full trace costs approximately $P \times K^2 \approx (7 \times 10^9) \times (4.2 \times 10^6) \approx 3 \times 10^{16}$ operations. Thus, the MIT overhead is roughly six orders of magnitude smaller than the generation cost. This illustrates that, for billion-parameter models and typical reasoning trace lengths, the cubic diagnostic is essentially negligible relative to autoregressive decoding.

A.8.5 EMPIRICAL COST ANALYSIS

Interestingly, the reasoning model demonstrates a lower overall computational cost compared to the average non-reasoning model. This counterintuitive result arises from how critical junctures are identified and the computational trade-offs inherent in our Adaptive Critical Sampling (ACS) approach. For example, reasoning models such as *DeepSeek-Distilled-Qwen-1.5B* tend to identify critical junctures later in their reasoning traces (at approximately 3,304 tokens). In this case, the diagnostic overhead introduced by MIT (1.51×10^{12} FLOPs) is effectively amortized across substantial resampling savings. As a result, the majority of *DeepSeek-Distilled-Qwen-1.5B*'s total cost (3.52×10^{17} FLOPs) comes from efficient suffix resampling rather than redundant full-sequence generation.

By contrast, non-reasoning models display greater variability in computational efficiency. *LLaMA-3.1-8B* achieves notable savings (61.0% reduction) by identifying critical points at an optimal 2,225 tokens, whereas *Gemma-3-270M-IT* detects critical junctures much earlier (563 tokens). This early detection leads to heavier suffix regeneration costs. Specifically, for *Gemma-3-270M-IT*, our method reduces computational cost from 5.69×10^{17} **FLOPs** under Self-Consistency to 3.89×10^{17} **FLOPs**—a reduction of **31.6%**. However, the early critical juncture means the MIT diagnostic overhead is less effectively amortized, highlighting how our method's computational efficiency depends critically on the timing of critical moment identification.

A.9 MIT MEMORY CONSUMPTION CALCULATION

A.9.1 THE MEMORY CONSUMPTION OF MIT

The primary limitation on the scalability of the Metacognitive Influence Tracing (MIT) algorithm is its memory consumption. The core of the method relies on storing and processing full $K \times K$ attention and influence matrices, where K is the length of the reasoning trace. This results in a memory complexity that scales **quadratically**, or $O(K^2)$, with the sequence length, which can become prohibitively large for very long sequences.

A.9.2 MITIGATION 1: LAYER-BY-LAYER CALCULATION

In our application, we mitigate this issue by avoiding the need to load the entire raw attention tensor (of size $L \times H \times K \times K$) into memory at once. Instead, the implementation performs a **layer-by-layer calculation**. It processes the attention matrices for each layer sequentially, updating a running aggregate for each attention head before discarding the raw layer data. This approach reduces the peak memory requirement from $O(L \times H \times K^2)$ down to $O(H \times K^2)$. While this is a significant practical improvement, it does not change the fundamental quadratic scaling with respect to the sequence length K.

A.9.3 MITIGATION 2: REDUCING COMPLEXITY TO $O(K)$

To achieve a true linear memory complexity of $O(K)$, a promising future direction is the integration of **sparse attention techniques**, such as a **sliding window** model. In this approach, instead of each token attending to all previous tokens, it would only attend to a fixed number of its immediate neighbors (a "window" of size w). This would reduce the memory required to store attention scores from $O(K^2)$ to $O(K \times w)$, which is linear if the window size w is a constant.

Beyond memory savings, adopting a sliding window also has significant implications for computational efficiency. In the current MIT framework, the diffusion step requires computing the matrix exponential of a dense $K \times K$ influence matrix, leading to cubic cost $O(K^3)$ as listed in Equation 18. By restricting attention to a banded sparse structure with bandwidth w, the propagation can be reformulated using sparse linear algebra routines. This reduces the effective cost to approximately $O(K \cdot w^2)$, since each token interacts only within its local neighborhood. When $w \ll K$, this can yield orders-of-magnitude reductions in floating-point operations, especially for long reasoning traces. Thus, sliding window not only lowers memory complexity but also offers a pathway to substantially reduce the computational overhead of MIT.

Informing Sliding Window Size with Research Findings The critical challenge with a sliding window approach is selecting an appropriate window size, as one that is too small could fail to capture important long-range dependencies in the reasoning process. Our findings in Figure 20, which is the analysis of the distance between critical tokens and the final answer, reveals that while a significant portion (over 20%) depend on much longer-range connections, sometimes exceeding 2000 tokens, the majority of influential interactions (over 60%) occur within a 500-token range.

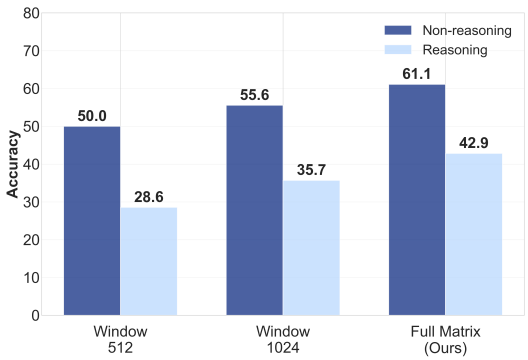


Figure 21: Sliding Window Comparison

Experiments Based on findings from Figure 20, we have conducted two experiments with the window size set as 512, 1024 respectively. The results are shown in Figure 21. In both settings we do observe a drop in accuracy, as the reduced window size limit the range of influential token searching. However we do notice that our propagation method with heat kernel enable tokens beyond the window size to be surfaced. This is supported by Figure 22 where we observe most influential token beyond the window size. For 28.6% of cases for window size 512 for reasoning models, the distance between most influential token and the answer are over 512. We hypothesize that tuning the propagation scale such as the gamma value can help with the propagation of influential tokens and leave it as future work.

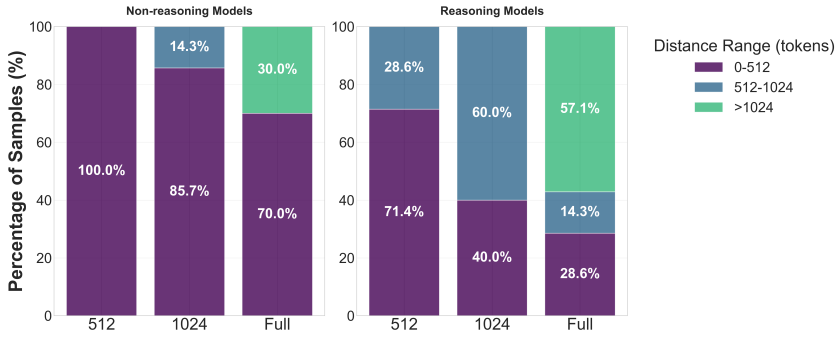
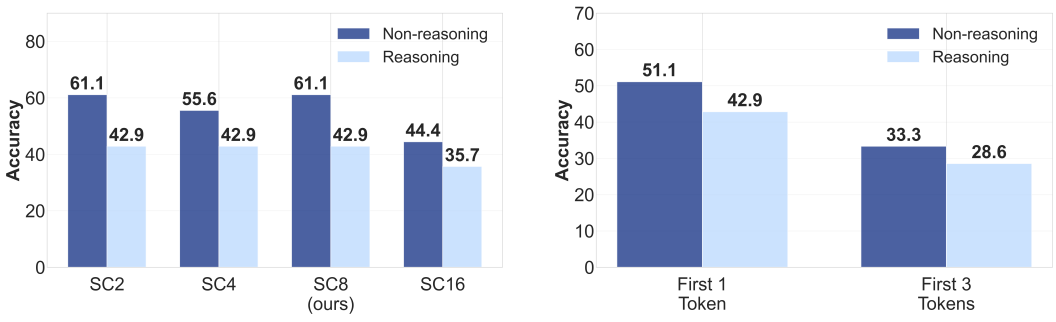


Figure 22: Distance between most influential token and answer token in different window size settings

A.10 Additional Experiments

Can we sample less? We conducted additional experiments to study how the number of samples affect the overall accuracy of ACS. We tried sample 2, 4, 8, 16 times and observe that reducing the number of samples does not necessarily decrease the performance, neither does increasing samples improve the performance as shown in Figure 23a. Based on the experiment, sampling 2 times will achieve similar effects.

Should we consider multiple critical junctures? Instead of just using the most critical juncture, we also evaluate the strategy of sampling the first 3 critical junctures, 3, 3, 2 time respectively. We observe a performance degradation when we consider more critical junctures as shown in Figure 23. This further validates our observation that not all critical junctures are created equal, and that cognitive investment are most impactful at the most critical juncture.



(a) Sampling Count Comparison (b) Sampling Strategy Comparison

Figure 23: Comparison of sampling approaches.

A.11 PROMPTS

Below are the prompts used in testing. We have follow the prompts used in (Olsson et al., 2022). And for the non-reasoning models like Gemma, we adopt the LLaMA prompt in that paper. For models with reasoning capabilities, we adopt the prompt of Qwen in that paper.

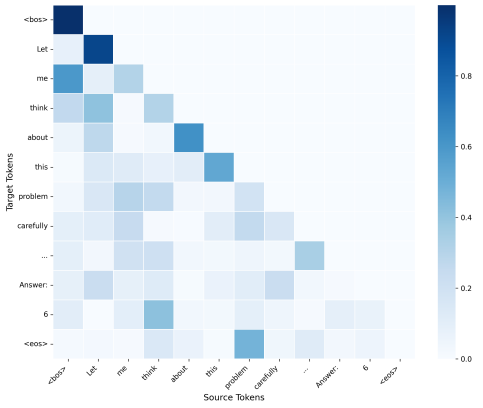
Table 4: Model Prompts

Model	**Prompt**
LLaMA-3.1-8B	Solve the following problem efficiently and clearly. - For simple problems (2 steps or fewer): Provide a concise solution with minimal explanation. - For complex problems (3 steps or more): Use this step-by-step format: ## Step 1: [Concise description] [Brief explanation and calculations] ## Step 2: [Concise description] [Brief explanation and calculations] ... Regardless of the approach, always conclude with: Therefore, the final answer is: `\boxed{answer}`. I hope it is correct. Where [answer] is just the final number or expression that solves the problem.
DeepSeek-Distilled–Qwen-1.5B	Please reason step by step, and put your final answer within `\boxed{}`.
SmolLM3-3B	Please reason step by step, and put your final answer within `\boxed{}`.
SmolLM3-3B(Think)	Please reason step by step, and put your final answer within `\boxed{}`.
Gemma-3-270M-IT	Solve the following problem efficiently and clearly. - For simple problems (2 steps or fewer): Provide a concise solution with minimal explanation. - For complex problems (3 steps or more): Use this step-by-step format: ## Step 1: [Concise description] [Brief explanation and calculations] ## Step 2: [Concise description] [Brief explanation and calculations] ... Regardless of the approach, always conclude with: Therefore, the final answer is: `\boxed{answer}`. I hope it is correct. Where [answer] is just the final number or expression that solves the problem.

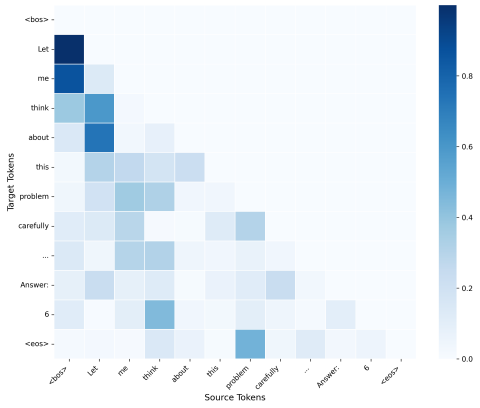
A.12 EXAMPLES

Below are examples of different components in MIT and ACS.

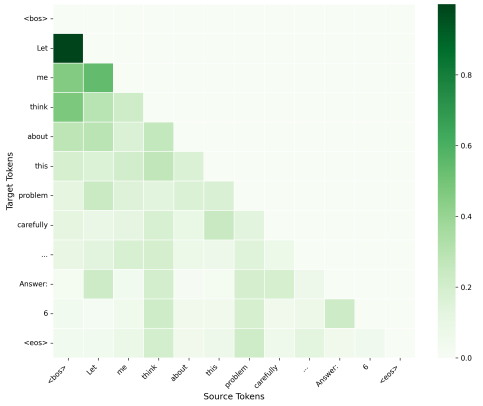
A.12.1 EXAMPLE FOR MIT



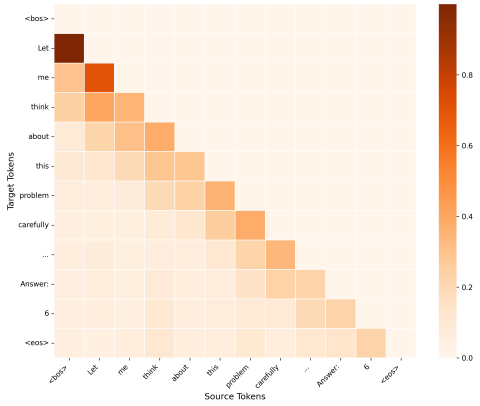
(a) Stage 0: Raw Attention (Layer 6, Head 3)



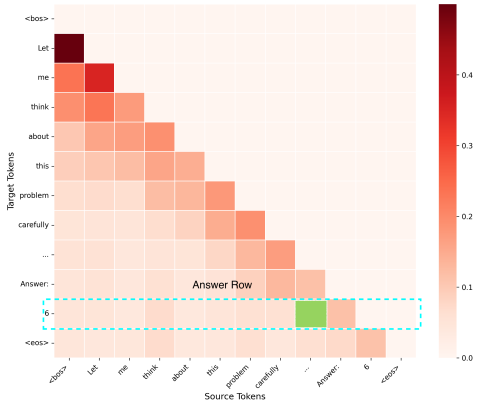
(b) Stage 1: Signal Purification (Layer 6, Head 3)



(c) Stage 2: Layer Aggregation (Head 3)



(d) Stage 3: Kurtosis-Weighted Head Aggregation



(e) Stage 4: Influence Propagation

Figure 24: Illustration example of the MIT algorithm. Raw attention goes through purification, layer aggregation, head aggregation and influence propagation to form the final score matrix. Only the rows related to the final answers were taken into consideration to select the most critical token.

A.12.2 EXAMPLE FOR ACS WITH REASONING MODEL

Question:

Given the sequence of words: apple shark vulture yak xerus. Concatenate the last letter of each word to form a new string. What is the resulting string?
Answer Choices:
(A) tkeks
(B) eteks
(C) ekeks
(D) ekejs
Please reason step by step, and put your final answer within \\boxed{}.

Original Answer Prefix

Let me try to think step by step. Okay, so I have this sequence of words: apple, shark, vulture, yak, xerous.

Wait

Original Answer Suffix

Wait, no, hold on, the sequence is apple, shark, vulture, yak, xerus. Wait, no, hold on, the sequence is apple, shark, vulture, yak, xerous. Wait, no, the user wrote: apple shark vulture yak xerus. So the words are: apple, shark, vulture, yak, xerous. Wait, hold on, xerous is a word? Or is it xerous? Wait, no, xerous is a word. Or is it xerous? Wait, no, xerous is a word. Wait, the last letters: apple ends with p, shark ends with k, vulture ends with t, yak ends with k, xerous ends with s. So the resulting string would be pktks.\n\nWait, but the options are (A) tkeks, (B) eteks, (C) ekeks, (D) ekejs. Hmm, none of these options match pktks. So maybe I misread the words.
Wait, let me check: the...Apple: last letter is t.\n\nShark: last letter is k.\n\nVulture: last letter is e.
Yak: last letter is k. Xerous: last letter is s.
So concatenated: tkeks. The final answer is: **\\boxed{A}**.

ACS Generated Suffix 1

I need to take the last letter of each word and then concatenate them all together to form a new string. Then, I have to figure out which of the given options is the resulting string. The options are (A) tkeks, (B) eteks, (C) ekeks, (D) ekejs. First, I need to make sure I understand the problem correctly. I have a list of five words: apple, shark, vulture, yak, xerous. For each of these words, I need to pick the last letter. Once I have those last letters, I concatenate them in the order they appear in the list to form a new string. Then, I have to see which of the given options matches this new string. Alright, so let's break it down step by step. Starting with the first word: apple. The last letter is 'e', right? Because 'apple' is spelled a-p-p-l-e. So the last letter is 'e'. Next, the second word is 'shark'. Let's see: s-h-a-r-k. So the last letter is 'k'....Concatenating these last letters in order gives us the string \"ekeks\".Thus, the resulting string is **\\(\\boxed{C}\\)**

ACS Generated Suffix 2

The task is to take the last letter of each word and concatenate them to form a new string. Then, among the given options, pick the correct one. The options are (A) tkeks, (B) eteks, (C) ekeks, (D) ekejs.Alright, let's break it down step by step. First, I need to identify the last letter of each word in the given sequence. The words are: apple, shark, vulture, yak, xerous.Starting with the first word, \"apple.\" The last letter is 'e.' So, the first character of the resulting string is 'e.'Next, the second word is \"shark.\" The last letter is 'k.' So, the second character is 'k.'\n\nMoving on to the third word, \"vulture.\" The last letter is 'e.' So, the third character is 'e.'Fourth word is \"yak.\" The last letter is 'k.' So, the fourth character is 'k.'Lastly, the fifth word is \"xerous.\" The last letter is 's.' So, the fifth character is 's.'Putting all these together, the resulting string is 'ekeks.'Looking at the options, (A) tkeks, (B) eteks, (C) ekeks, (D) ekejs. Hmm, 'ekeks' doesn't exactly match any of these. Wait, did I do that right?Let me double-check each last letter. Apple ends with 'e,' shark ends with 'k,' vulture ends with 'e,' yak ends with 'k,' ...Concatenating these last letters in order, we get: e, k, e, k, s, which forms the string \"ekeks\".Thus, the resulting string is **\\(\\boxed{C}\\)**

Figure 25: Example of how ACS allows correction for the generated answer. The critical juncture is the word "*wait*" in this case, allowing the following trace to ignore the hesitation on the sequence.

A.12.3 EXAMPLE FOR ACS WITH NON-REASONING MODEL

Question:

Name reactions in chemistry refer to a specific set of well-known chemical reactions that are typically named after their discoverers or the scientists who made significant contributions to their development. These reactions have had a profound impact on the field of chemistry and are often used as fundamental building blocks in various chemical syntheses. Identify the reactants for the following name reactions.
A + H2SO4 ---> 2,8-dimethylspiro[4.5]decan-6-one
B + BuLi + H+ ---> 4-methyl-1-phenylpent-3-en-1-ol

Answer Choices:
(A) A = 2,7-dimethyloctahydronaphthalene-4a,8a-diol, B = 4-methyl-1-phenylpent-3-en-1-one
(B) A = 2,8-dimethylspiro[4.5]decan-6-ol, B = 4-methyl-1-phenylpent-3-en-1-one
(C) A = 2,8-dimethylspiro[4.5]decan-6-ol, B = (((3-methylbut-2-en-1-yl)oxy)methyl)benzene
(D) A = 2,7-dimethyloctahydronaphthalene-4a,8a-diol, B = (((3-methylbut-2-en-1-yl)oxy)methyl)benzene

---

Original Answer Prefix

Step 1: To solve this problem, we first need to identify the name reactions and their corresponding reactants.
Step 2: The first reaction is A + H2SO4 ---> 2,8-dimethylspiro[4.5]decan-6-one. This reaction is likely the 2,7-dimethyloctahydronaphthalene-4a,8a-diol hydrolysis reaction.
Step

Original Answer Suffix

Step 3: The second reaction is B + BuLi + H+ ---> 4-methyl-1-phenylpent-3-en-1-ol. This reaction is likely the Friedel-Crafts alkylation of 4-methyl-1-phenylpent-3-en-1-one.
Step 4: Therefore, we can identify the reactants as A = 2,7-dimethyloctahydronaphthalene-4a,8a-diol and B = 4-methyl-1-phenylpent-3-en-1-one.
The final answer is: **A**

ACS Generated Suffix 1

Step 3: The second reaction is B + BuLi + H+ ---> 4-methyl-1-phenylpent-3-en-1-ol. This reaction is likely the 4-methyl-1-phenylpent-3-en-1-one alkylation reaction, but it seems like it's a lithiation reaction of the benzene ring,.. The given reactant for this reaction is likely (((3-methylbut-2-en-1-yl)oxy)methyl)benzene.
Step 4: Based on the above analysis, we can conclude that the correct reactants for the given name reactions are A = 2,7-dimethyloctahydronaphthalene-4a,8a-diol and B = (((3-methylbut-2-en-1-yl)oxy)methyl)benzene.
Step 5: Therefore, the correct answer is option D.
The final answer is: **D**

ACS Generated Suffix 2

Step 3: The second reaction is B + BuLi + H+ ---> 4-methyl-1-phenylpent-3-en-1-ol. This reaction is likely the Grignard reaction. In a Grignard reaction, a Grignard reagent (RMgX) reacts with a carbonyl compound to form an alcohol. The Grignard reagent is formed by the reaction of an alkyl halide with magnesium.
Step 4: The Grignard reagent formed in this reaction is likely 4-methyl-1-phenylpent-3-en-1-ylmagnesium bromide, which reacts with H+ to form the alcohol 4-methyl-1-phenylp...which would then react with 4-methyl-1-phenylpent-3-en-1-one to form the alcohol 4-methyl-1-phenylpent-3-en-1-ol.
Step 5: Therefore, the reactants for the given name reactions are A = 2,7-dimethyloctahydronaphthalene-4a,8a-diol and B = (((3-methylbut-2-en-1-yl)oxy)methyl)benzene.
The correct answer is (D).
Step 6: Therefore, the final answer is (D).
Step 7: Therefore, the final answer is **D**

Figure 26: Example of how ACS allows correction for the generated answer. The critical juncture is the word "*step*" in this case. A different step 3 leads to the correct answer.

A.12.4 FAILURE EXAMPLE FOR ACS

Question:
Given the sequence of words: turtle carrot hippo. Concatenate the last letter of each word to form a new string. What is the resulting string?

Answer Choices:
(A) eho
(B) etc
(C) eto
(D) etv

Original Answer Prefix

Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n ...Final Answer: (A)\nFinal Answer: (A)\nFinal

Answer

Original Answer Suffix

Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n ...Final Answer: (A)\n Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n ...Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n
Final Answer: **A**

ACS Generated Suffix 1

Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n ...Final Answer: (A)\n Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n ...Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n
Final Answer: **A**

ACS Generated Suffix 2

Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n ...Final Answer: (A)\n Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n ...Final Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\nFinal Answer: (A)\n
Final Answer: **A**

Figure 27: Example of how ACS fails. A lot of the time the failure comes from identifying a critical juncture too late where the model are already confirmed about an answer.

A.13 Potential Impact and Future Works

Our work demonstrates that LLM reasoning is not a uniform chain of steps but a process punctuated by critical junctures. We define such critical junctures as where cognitive or computational resource would make the most impact. We introduced **Metacognitive Influence Tracing (`MIT`)**, a principled diagnostic method to identify these junctures, and showed how this understanding can be translated into a high-efficiency interventive reasoning framework, **Adaptive Critical Sampling (`ACS`)**. The ability to pinpoint these critical moments has implications that extend beyond improving Chain-of-Thought prompting, touching on challenges in AI safety, evaluation, and future model development.

Implications for AI Safety and Alignment. A key challenge in AI safety is understanding and preventing undesirable model behavior, such as hallucinations or harmful content. Our results, similar to findings on "forking tokens," (Bigelow et al., 2025) suggest that a model may be just one critical step away from producing a radically different—and potentially unsafe—output. `MIT` can serve as a diagnostic tool for failure analysis: when a model produces an undesirable result, `MIT` traces the reasoning back to the specific flawed inference that initiated the failure. This moves beyond detecting that an output is wrong, providing mechanistic insight into *where* it could be corrected, and enabling targeted safety interventions.

Implications for Model Development and Evaluation. The insights from `MIT` could inform more efficient model design. By aggregating critical junctures across thousands of problems, we can potentially map the model components (e.g., attention heads or layers) most crucial for reasoning. This data could guide structured pruning or distillation, yielding smaller models that preserve disproportionate reasoning capability. Furthermore, our findings suggest that static evaluations measuring only the final answer can be misleading. A model might appear confident in its conclusion, yet `MIT` reveals that it passed through highly uncertain junctures where outcomes could have easily diverged, exposing brittleness hidden by the final result.

Limitations and Future Work. Our approach also has limitations. `MIT` currently relies on explicit, text-based reasoning traces, raising questions about its applicability to tasks where reasoning is implicit. While `ACS` is more efficient than brute-force baselines, the `MIT` analysis itself can be computationally intensive; future work could develop more efficient approximations of influence propagation such as mitigation method mentioned in Section A.9.3.

A promising direction is to integrate `MIT` with other reasoning frameworks. Critical junctures identified by `MIT` could guide the branching of search algorithms such as Tree of Thoughts, concentrating expansion at points of highest leverage. This would combine the breadth of exploration with the precision of targeted intervention, paving the way for hybrid reasoning systems. Also, applying `ACS` repeatedly can also enable continuous improvement.

Another future work is to zoom out the scope, instead of using token as the basic unit, we can zoom out to sentence or paragraph level as the basic unit of thought. This can provide another granularity on the reasoning process of LLMs.

A.14 Use of LLMs

We have used LLMs to polish writing for this paper.