Diagnosing Memorization in Chain-of-Thought Reasoning One Token at a Time

Huihan Li[1*] **You Chen**[2*] **Siyuan Wang**[1] **Yixin He**[1] **Ninareh Mehrabi**[3†]
Rahul Gupta[4] **Xiang Ren**[1]
[1]University of Southern California [2]University of California, San Diego
[3]Meta [4]Amazon AGI
{huihanl,xiangren}@usc.edu, yoc033@ucsd.edu

Abstract

Large Language Models (LLMs) perform well on reasoning benchmarks but often fail when inputs alter slightly, raising concerns about the extent to which their success relies on memorization. This issue is especially acute in Chain-of-Thought (CoT) reasoning, where spurious memorized patterns can trigger intermediate errors that cascade into incorrect final answers. We introduce **STIM**, a novel framework for **S**ource-aware **T**oken-level **I**dentification of **M**emorization, which attributes each token in a reasoning chain to one of multiple memorization sources – local, mid-range, or long-range – based on their statistical co-occurrence with the token in the pretraining corpus. Our token-level analysis across tasks and distributional settings reveals that models rely more on memorization in complex or long-tail cases, and that local memorization is often the dominant driver of errors, leading to up to 67% of wrong tokens. We also show that memorization scores from **STIM** can be effective in predicting the wrong tokens in the wrong reasoning step. **STIM** offers a powerful tool for diagnosing and improving model reasoning and can generalize to other structured step-wise generation tasks.[1]

1 Introduction

Large Language Models (LLMs) perform well on reasoning tasks but often fail under slight input changes, raising concerns about overreliance on memorization (Hong et al., 2025; Lou et al., 2024; Jin et al., 2024; Salido et al., 2025). Long Chain-of-thought (CoT) (Wei et al., 2022) chains are especially vulnerable, as spurious memorization can introduce early errors that derail final answers. As inference-time scaling encourages longer CoTs, detecting token-level memorization is critical for assessing reasoning reliability, particularly under distributional shifts from frequent to rare inputs (Xie et al., 2024; Prabhakar et al., 2024).

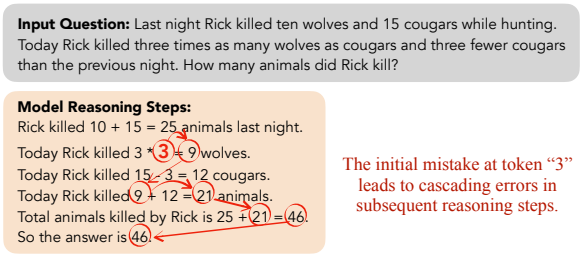


Figure 1: Cascading errors in Chain-of-thought (CoT) reasoning can stem from a single mis-predicted token, often influenced by incorrect memorization of pretraining data. This motivates our study on the impact of token-level memorization.

We argue that memorization in long Chain-of-thought generations must be identified at the **token level** rather than the sequence level. A single faulty step can cause cascading errors, often stemming from a few erroneous tokens (Figure 1). Identifying these tokens and whether they result from memorization is essential. Moreover, we argue that accurately measuring token-level memorization requires accounting for **multiple sources of influence**, including both the input prompt and prior output tokens, which jointly shape each token's generation (Table 1).

Prior approaches are insufficient for analyzing token-level memorization and how memorization patterns from different sources shift under distributional change: existing metrics do not target memorization at the level of individual tokens, instead reporting a single score for the entire sequence or final answer. Moreover, they either focus solely on memorization in the output sequence (McCoy et al., 2023; Merrill et al., 2024; Lu et al., 2024) or assess the influence of memorization from the input (Carlini et al., 2022; Biderman et al., 2023; Li et al., 2025; Wang et al., 2024), without accounting for multiple sources of influence on token-level memorization.

*Equal Contribution
†Work done in Amazon AGI
[1]https://github.com/INK-USC/STIM

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 3159–3181
November 4-9, 2025

Examples of Simultaneous Memorization Sources of Different Strength
Question: Here is a list: [pitanga, pitanga, yumberry, yumberry, pitanga, yumberry, pitanga, yumberry, pitanga, pitanga]. How many times does 'pitanga' appear on it? **Model Output**: 1. The first element is 'pitanga'. ... 10. The tenth element is 'pitanga'. Now, counting the occurrences of 'pitanga': We have 'pitanga' appearing 5 times at positions 1, 2, 5, 7, and 10. So the answer is 5.
Analysis: Token 5 is incorrect and has the highest influence from local context, where the n-gram "appearing 5" has a high pretraining frequency. **STIM** scores are: local = 0.81, mid = 0.24, long = 0.03, suggesting that the model was influenced by locally frequent patterns rather than actual counting.
Question: What is (100101100 - 100101100 * 0.1) * 111110100 + 100101100 * 1111 / 1100100 * 111110100 * 0.1 + (100101100 - 100101100 * 0.1 - 100101100 * 1111 / 1100100) * 111110100 equal to? (base-2) **Model Output**: To solve this, we first handle the multiplication and division in base-2, converting to base-10 as necessary: 1. $(100101100 \times 0.1_{\text{base-2}} = 10010110_{\text{base-2}})$ (which is 180 in base-10) ... So the answer is 26807.536.
Analysis: Token 180 is incorrect and has the highest influence from input tokens, having higher pretraining co-occurrence frequency with "111110100", "*" and "100101100" in the input. **STIM** scores are local=-0.19, mid=0.09, long=0.156, indicating long-range memorization being the primary influence.

Table 1: Each error token is affected simultaneously by different memorization sources, but some sources may be stronger than others. Shown are examples with local context (local) and input prompt (long) having maximum influence, with tokens associated with the source highlighted. How we obtain **STIM** scores for each source is explained in Section 4.

To address these gaps, we propose **STIM** (**S**ource-aware **T**oken-level **I**dentification of **M**emorization), a framework that captures token-level memorization by tracing influences from both the input and prior outputs on erroneous reasoning steps. For each token, **STIM** computes the strength of three memorization sources: (1) *local*, from frequent continuations of immediately preceding tokens; (2) *long-range*, from frequent co-occurrence with prompt tokens; and (3) *mid-range*, when the model generates the target token when conditioned only on a prefix of the generation, we identify tokens that frequently co-occur with the target token in pretraining. **STIM** offers a fine-grained view of multi-source memorization and its strength at each token.

We begin our analysis by using **STIM** to uncover broad memorization trends across tasks, input distributions, and correctness. As reasoning complexity increases, models exhibit greater reliance on memorization. Distribution shifts toward rare or atypical inputs also lead to stronger memorization signals. Interestingly, while memorization often supports correct answers in base settings, it more frequently contributes to errors in long-tail scenarios, suggesting defective recall when faced with unfamiliar contexts.

To demonstrate the utility of our framework, we apply it to the task of identifying erroneous tokens in erroneous reasoning steps. By tracing the dominant source of memorization for each incorrect token, we find that local memorization (continuations driven by immediately preceding tokens) is the most common cause of error (up to 67%). However, under distribution shift, complex tasks show a marked decline in local memorization-driven mistakes, implying reduced reliance on familiar patterns. Finally, we assess the effectiveness of **STIM** in pinpointing erroneous tokens via Precision@k and Recall@k, showing that high memorization scores are strong indicators of reasoning failures.

Our novel **STIM** framework is the first to enable fine-grained, token-level memorization identification in long reasoning chains, considering multiple influence sources from both input and generated context. Through systematic evaluation across diverse Chain-of-thought tasks, we demonstrate its crucial role in pinpointing memorized content that

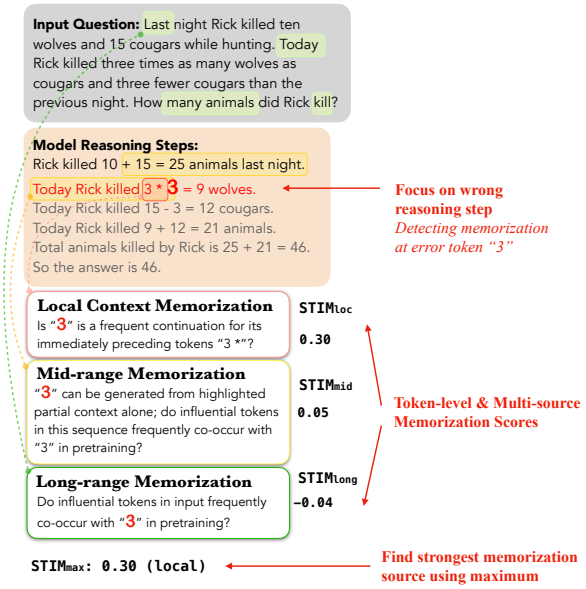


Figure 2: **STIM** quantifies local, mid-range, and long-range memorization at each token in a faulty reasoning step and identifies the dominant source. **STIM** is helpful for detecting error tokens.

leads to errors and reveal how different memorization sources dominate across tasks and under distributional shift. While focused on Chain-of-thought, **STIM** is extendable to other long-form formats such as dialogue or summarization. This framework offers a powerful tool for diagnosing LLM reasoning failures and advancing research into genuine reasoning capabilities.

2 Related Works

Pretraining Data Memorization Metrics. A growing body of work investigates how and when large language models (LLMs) memorize their pretraining data. Early approaches focused on measuring extractability – the ease with which specific training data can be reproduced from the model—highlighting risks to privacy and model overfitting (Carlini et al., 2022; Biderman et al., 2023). Other efforts have analyzed memorization via novelty metrics, which assess the similarity of model outputs to seen data (McCoy et al., 2023; Merrill et al., 2024; Lu et al., 2024). Another class of work quantifies memorization through n-gram overlap or token-level attribution, often drawing connections between model predictions and local or distant pretraining contexts (Li et al., 2025; Wang et al., 2024).

Distinguishing Memorization from Reasoning. Beyond surface-level memorization, recent work has aimed to disentangle genuine reasoning from pattern recall. Xie et al. (2024) fine-tune models on controlled datasets to probe the boundary between memorization and generalization in reasoning tasks. Complementary to this, Hong et al. (2025) approach the problem from a mechanistic interpretability perspective, identifying internal circuits associated with memorized versus reasoned responses. On the modeling side, Lou et al. (2024) propose methods for quantifying chaotic and foundational memorization by analyzing in-context learning dynamics at the logit level, while Jin et al. (2024) focus on how training conditions shape these behaviors. In multiple-choice settings, Salido et al. (2025) introduce a technique to isolate reasoning by deliberately eliminating answer options that could be matched through memorized heuristics. Finally, Prabhakar et al. (2024) study the interplay between probability calibration, memorization, and noise in chain-of-thought prompting, revealing that correct answers can arise from superficial token-level cues rather than coherent reasoning.

3 Preliminary Analysis: CoT Reasoning-Memorization Correlation

Recent work shows that language models struggle to generalize to rare task formats or uncommon input entities despite strong performance on frequent patterns, suggesting that memorization significantly influences model behavior (Wu et al., 2024; Dziri et al., 2023; Li et al., 2024a). To investigate how memorization impacts CoT Reasoning, we construct a controlled set of reasoning tasks and introduce targeted long-tail transformations that either decrease the frequency of input entities or alter task formats to less common variants. This section introduces our experimental setup and motivates our framework by demonstrating how reasoning performance changes under distributional shift.

3.1 Setup

Tasks. We evaluate memorization on four reasoning tasks of varying complexity: *Applied Math, Formula Calculation, Counting, and Capitalization.* All tasks require step-by-step reasoning under a Chain-of-thought prompting setup as well as a direct answer setup(full prompt in Appendix G). We collect/construct each task as follows (more details in Appendix A:

- *Applied Math*: GSM8K (Cobbe et al., 2021).
- *Formula Calculation*: Final equations extracted from GSM8K answer derivations.

- *Counting*: Lists of fruit entities with controlled frequency ranges (10^3 to 10^7) and varying lengths (10–50).
- *Capitalization*: Book titles sampled from the Book Cover Dataset (Iwana et al., 2016).

These tasks involve different reasoning complexity levels. *Applied Math* and *Formula Calculation* involve multi-step deduction, where each intermediate step depends on correct synthesis of all prior steps. In contrast, *Counting* and *Capitalization* are more locally structured, where each step is relatively independent, and the correct answer often depends on a single, final decision. This distinction allows us to examine how memorization interacts with different reasoning dynamics, especially when tracing errors in multi-step generation.

Long-tail Transformations. To evaluate memorization under distributional shift, we apply long-tail transformations to each task (Table 5). These modifications reduce the frequency of input entities or reformulate tasks in less common formats:

- *Applied Math & Formula Calculation*: Frequency reduction via digit expansion or converting integers to floats; task variation by expressing problems in base-2 (Li et al., 2024b).
- *Counting*: Lower-frequency countable entities and increased list lengths.
- *Capitalization*: Atypical formulation requiring capitalization of the last word's first letter.

Each transformation is applied independently, and the final long-tail set per task pools all variants. The total number of base and long-tail examples we collect is in Table 6.

Model and Pretraining Data. Our experiments and analyses utilize OLMo 2(OLMo et al., 2024) (`OLMo-2-1124-13B-Instruct`) under default HuggingFace settings and greedy decoding. Its pretraining corpus, Dolma 1.7 (Soldaini et al., 2024), is indexed via Infinigram (Liu et al.), allowing token frequency lookup. To our knowledge, only OLMo and Pythia offer fully open, indexed pretraining data; however, Pythia underperforms on GSM8K (<5%), limiting its use in CoT studies. To show generalizability of our methodology, we replicate all main analyses on `olmo2-1124-7B-Instruct` in Appendix D.

3.2 Performance under Distribution Shift

OLMo 2's performance across both direct answer and CoT formats for all tasks is listed in Table 2.

(a) **CoT Reasoning**

Task	Base	Long-tail
Applied Math	82.0	25.6
Formula Calculation	89.8	39.3
Counting	43.1	19.2
Capitalization	26.7	53.1

(b) **Direct Answer**

Task	Base	Long-tail
Formula Calculation	33.3	11.3
Counting	40.5	21.4
Capitalization	87.5	47.8

Table 2: Model accuracy (%) on reasoning tasks under base and long-tail input distributions. Direct Answer for Applied Math is omitted due to poor accuracy.

Base Distribution Outperforms Long-tail For *Applied Math*, *Formula Calculation*, and *Counting*, performance in the base distribution setting consistently outperforms that in the long-tail distribution. This trend reinforces the notion that current models rely heavily on memorization, benefiting from high-frequency entities and familiar patterns in the input. The observed performance degradation under long-tail input distributions reflects the model's difficulty generalizing to rare entities or less common problem variations.

CoT Amplifies the Base vs. Long-tail Gap The performance gap between base and long-tail distributions is significantly larger under the CoT format compared to direct answers. This suggests that the extended generation sequences in CoT reasoning exacerbate the model's reliance on memorized patterns, increasing error probability when familiar cues are absent. The phenomenon necessitates deeper analysis of how CoT prompts may amplify memorization-related errors.

Capitalization Highlights Token-level Fragility An exception arises in the *Capitalization* task, where CoT notably degrades performance in the base setting. While the model can directly retrieve correct answers for well-known book titles in the direct answer format, CoT reasoning introduces intermediate steps that expose the model to more opportunities for spurious generations. This example illustrates that while certain forms of memorization (e.g., direct recall) can support accurate performance, others (e.g., misplaced pattern matching during reasoning) can introduce critical errors. The discrepancy underscores the need for token-level analysis to understand when memorization

helps versus when it harms.

In summary, the results demonstrate that memorization plays a central role in model behavior under distribution shift, particularly under CoT, where longer reasoning chains amplify its effects. This motivates our fine-grained analysis of token-level memorization dynamics in following sections.

4 Measuring Token-Level Memorization

In multi-step reasoning, token predictions are influenced by the local context, the input prompt, and previously generated output. Each contributes to memorization differently: local context drives frequent continuations, while prompts and past outputs reflect longer-range associations from pretraining. Disentangling these sources enables more precise diagnosis of how memorization affects reasoning, especially under distributional shifts. We introduce **S**ource-aware **T**oken-level **I**dentification of **M**emorization (**STIM**), a method for identifying token-level memorization from local, mid-range, and long-range sources.

4.1 Memorization from Distinct Contextual Sources

We now describe how to quantify memorization for a target token x, conditioned on the full context $p = [\text{input}; \text{output}_{<x}]$, where $\text{output}_{<x}$ denotes all preceding tokens in the generated answer before x.

Local Context Memorization (*Local*) This score quantifies how much a token's generation is driven by frequent continuations in its immediate local context. For a target token x, we identify w, the longest contiguous prefix such that the n-gram $[w; x]$ appears at least once in the pretraining corpus. At decoding time, we extract the top-20 candidate tokens $x_i{}_{i=1}^{20}$ with probabilities $P(x_i \mid p)$ and retrieve their corresponding n-gram frequencies $f([w; x_i])$. The local memorization score is computed as the Spearman correlation:

$$\text{STIM}_{loc}(x) = \rho\left(\{P(x_i \mid p)\}_{i=1}^{20},\ \{f([w; x_i])\}_{i=1}^{20}\right) \quad (1)$$

Input-driven Memorization (*Long-Range*) This score measures the influence of salient input tokens that co-occurred with the target token in pretraining. We identify the top-5 most influential input tokens to the target token using token saliency (Tuan et al., 2021)[2], denoted $S_l = s_1, \ldots, s_5$. At decoding time, we obtain the top-20 candidates x_i with probabilities $P(x_i \mid p)$ and compute their co-occurrence frequencies with S_l, denoted $f(S_l, x_i)$. The long-range memorization score is defined as:

$$\text{STIM}_{long}(x) = \rho\left(P(x_i \mid p)_{i=1}^{20}, f(S_l, x_i)_{i=1}^{20}\right) \quad (2)$$

Partial Output Memorization (*Mid-Range*) This score captures how much a token's generation is influenced by spurious associations within the partially generated answer. For a target token x, we find the shortest contiguous prefix of the model's generated answer (excluding input tokens) that leads to the model generating x when conditioned on that span. We then identify the top-5 most salient tokens in this context $S_m = \{s_1, \ldots, s_5\}$ using token saliency. We collect the top-20 candidate tokens $\{x_i\}_{i=1}^{20}$ with probabilities $P(x_i \mid p)$. For each candidate x_i, we compute the average co-occurrence frequency with the salient tokens S, denoted $f(S_m, x_i)$. The mid-range memorization score is defined as:

$$\text{STIM}_{mid}(x) = \rho\left(P(x_i \mid p)_{i=1}^{20}, f(S_m, x_i)_{i=1}^{20}\right) \quad (3)$$

Dominant Source Attribution To capture the full extent of memorization effects at the token level, we also introduce STIM_{max}, taking the maximum of local, long and mid-range memorization scores. We identify the **dominant source** as the memorization source with the highest score, indicating which contextual factor most strongly influenced the token's generation.

5 Token-Level Memorization Analysis

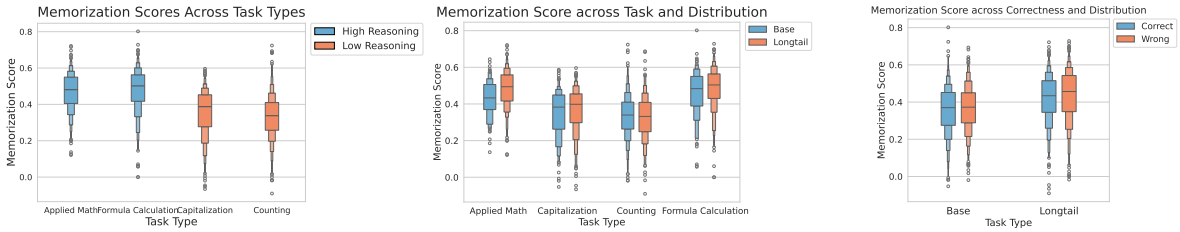
In this section, we use **STIM** to perform fine-grained token-level analysis. We begin by describing our experimental setup and how we select essential reasoning steps to analyze. We then apply **STIM** to compare token-level memorization patterns across tasks, distributions (base vs. long-tail), and correctness (correct vs. incorrect reasoning).

5.1 Experimental Setup and Step Selection

Our analyses aim to contrast the model's token-level memorization behavior between base and long-tail input distributions, as well as between correct and incorrect CoT reasoning chains.

In analyzing reasoning failures, we focus on the **first erroneous step** in each flawed chain rather than the entire output sequence. This design choice is motivated by the observation that subsequent

[2]Ablation see Appendix E.



(a) By Task (b) By Distribution (c) By Correctness

Figure 3: Comparison of STIM$_{max}$ distribution between task reasoning complexity, distribution and correctness.

reasoning steps often suffer from cascading effects caused by earlier mistakes, so such errors may not be directly attributable to memorization. By isolating the first incorrect step, we target the *initial point of failure* where memorization-related influences are most likely to manifest. This approach also aligns with recent practice in process supervision (Luo et al., 2024; Lightman et al., 2023), where Process Reward Models (PRMs) are commonly trained to identify the earliest incorrect step.

For each task and for both the base and long-tail settings, we uniformly sample 200 examples where the model's final answer is incorrect (the **wrong** set) and 200 examples where the final answer is correct (the **correct** set). To identify the reasoning steps that lead to incorrect final answers, we apply VersaPRM (Zeng et al., 2025), a multi-domain Process Reward Model trained across 14 diverse domains. A step is classified as erroneous if its PRM score (ranging from 0 to 1) below an empirically determined threshold of 0.9. We exclude non-substantive statements like "Let's verify" or "Let's think step by step". We focus on the first one below the threshold as the first erroneous step for analysis. In practice, PRMs are not always reliable. They may miss reasoning errors and assign overly high scores even when the final answer is wrong. To ensure error in the selected step, we exclude such cases from our **wrong** set and retain only examples containing at least one step with a PRM score below 0.9. For each example in the *correct* set, we select the reasoning step with the lowest PRM score to provide a point of contrast. All subsequent analyses are conducted on these selected reasoning steps. We verify the reliability of VersaPRM in Appendix E.

5.2 Analyzing Memorization Patterns by Task, Distribution, and Correctness

Now we illustrate the presence of memorization across tasks, distributions and correctness. While each token may be predominantly influenced by different sources, STIM$_{max}$, as the maximum of all three sources of memorization, represents the highest level of contextual influence at each token. Therefore, we present our observations on the distribution of STIM$_{max}$.

More complex reasoning tasks have higher memorization score. Across all four settings, Applied Math and Formula Calculation consistently show higher dominance scores compared to Capitalization and Counting (Figure 4a). This pattern aligns with the nature of the tasks: Applied Math and Formula Calculation involve more complex reasoning, requiring the model to iteratively synthesize partial solutions at each step. In contrast, Counting and Capitalization demand relatively simple, local decisions during intermediate steps, with synthesis needed only at the final stage.

Memorization scores are higher in long-tail settings. In 3 of 4 tasks, long-tail examples (brown) show consistently higher memorization scores than base examples (blue) (Figure 4b). This may seem counterintuitive: base examples contain more frequent entities, so one might expect greater reliance on memorized content. Instead, the elevated scores suggest that when faced with unfamiliar inputs, the model often inappropriately falls back on spurious patterns memorized during pretraining, leading to more errors. Additional evidence is provided in Section 6.

The exception is *Counting*, where scores are similar across settings. This likely stems from the task's rigid reasoning format (e.g., "The *X*th element is *Y*"). Despite longer lists in the long-tail setting, the model typically makes its first mistake around the 8th step in both cases. As a result, error patterns remain structurally similar, leaving little room for memorization differences to emerge.

Distribution Shift Reverses the Role of Memorization. We observe a shift in the relationship between memorization and correctness across dis-

tributions (Figure 4c). In the base settings, correct reasoning steps exhibit higher memorization scores than incorrect ones, suggesting that memorized content aligns well with the input and supports accurate reasoning. This is especially evident in more complex tasks like Applied Math and Formula Calculation, where correct solutions may involve retrieving familiar mathematical equation patterns seen during pretraining. In contrast, in the long-tail setting , incorrect reasoning steps have higher memorization scores, indicating that the model often falls back on memorized but ill-fitting patterns when faced with less familiar inputs. This suggests that memorization, while helpful in familiar contexts, can hinder generalization under distribution shift – especially when the model applies high-frequency completions to novel or rare scenarios. The reversal in score patterns highlights how the same underlying memorization behavior can contribute to success or failure depending on the alignment between the input and the model's pretraining data.

Task	Dist.	Local	Mid	Long
Applied Math	Base	**0.673**	0.127	0.200
	Long-tail	**0.474**	0.299	0.227
Formula Calculation	Base	**0.517**	0.186	0.297
	Long-tail	**0.443**	0.253	0.304
Counting	Base	**0.520**	0.208	0.272
	Long-tail	**0.663**	0.207	0.130
Capitalization	Base	**0.647**	0.141	0.212
	Long-tail	**0.649**	0.185	0.166

Table 3: Dominant memorization source (%) by task and distribution type. Each row shows the percentage of erroneous tokens where each source (Local, Mid, Long) was the most influential.

6 Detecting the Wrong Token using STIM

With our framework for computing token-level memorization scores across multiple sources, we now explore a key application: identifying tokens that cause reasoning failures. We assess whether high memorization scores can pinpoint erroneous tokens in flawed reasoning steps, so as to demonstrate the framework's value as a diagnostic tool for understanding how memorized content drives errors in long-form generation.

For each example that the model answers wrongly, we use `gpt4-o` to identify the wrong tokens in the erroneous reasoning step identified by VersaPRM described in Section 4. The exact prompt used for `gpt4-o` is in Appendix B.

Dominant Source of Erroneous Memorization. We begin by identifying the **most influential type of memorization** (local, mid-range, or long-range) for each wrong token across tasks and distributional settings. This analysis reveals which memorization source most often drives the model's reasoning in cases where it generates incorrect outputs. The percentages reported in Table 3 reflect the share of all identified error tokens where a particular source has the highest memorization score.

Across all tasks and distributions, **local memorization** consistently emerges as the most frequent driver of erroneous token generation, accounting for as high as 67% of error tokens. This suggests that models often fall for spurious short-range patterns even in tasks requiring structured reasoning.

While this heavy reliance on local cues is generally undesirable, since local context carries minimal task-relevant information, its distributional dynamics reveal an interesting trend. Under distribution shift from base to long-tail, **high-reasoning tasks** such as *Applied Math* and *Formula Calculation* show a **notable decrease** in the proportion of errors attributable to local memorization. In contrast, **low-reasoning tasks** like *Counting* and *Capitalization* show little to no such reduction.

This suggests that when facing unfamiliar, long-tail inputs, the model is forced to abandon brittle local heuristics in high-reasoning tasks and may instead attempt to engage more global or structured reasoning strategies. However, for simpler tasks where local patterns suffice even in the long-tail, the model continues to rely on them.

Correspondence between high memorization and token error. To further examine the alignment between high memorization and reasoning error, we also report **Precision@k** and **Recall@k** for $k \in [1, 3]$, because we find that `gpt-4o` at most identifies 3 wrong tokens in a reasoning step.

Precision@k measures whether the top-k tokens with highest memorization scores are wrong tokens. It is defined as:

$\mathcal{W}$: Set of wrong tokens

$\mathcal{S}$: Token set with top-k memorization score

$$Precision@k = \frac{\sum_{\text{token} \in \mathcal{W}} \mathbb{I}(\text{token} \in \mathcal{S})}{min(|\mathcal{W}|, k)} \quad (4)$$

The denominator handles cases where the number of wrong tokens is less than k, when we take

(a) Precision@k

Task	Level	P@1	P@2	P@3
Applied Math	$\text{STIM}_{\max}$	45.5	45.2	57.2
	Random	15.8	23.2	29.8
Formula Calc.	$\text{STIM}_{\max}$	41.0	49.2	60.5
	Random	20.9	34.1	38.5
Counting	$\text{STIM}_{\max}$	21.0	28.8	41.7
	Random	10.1	16.3	23.2
Capitalization	$\text{STIM}_{\max}$	17.2	32.0	42.7
	Random	10.0	17.9	26.6
All Tasks	$\text{STIM}_{\max}$	**31.2**	**38.8**	**50.5**
	Random	14.2	22.9	29.5

(b) Recall@k

Task	Level	R@1	R@2	R@3
Applied Math	$\text{STIM}_{\max}$	40.5	55.0	67.0
	Random	12.8	24.8	35.2
Formula Calc.	$\text{STIM}_{\max}$	37.5	54.0	68.0
	Random	15.5	29.2	40.7
Counting	$\text{STIM}_{\max}$	19.5	31.0	45.5
	Random	13.0	25.5	37.7
Capitalization	$\text{STIM}_{\max}$	17.5	34.5	46.0
	Random	11.4	27.4	40.1
All Tasks	$\text{STIM}_{\max}$	**28.8**	**43.6**	**56.6**
	Random	13.2	26.7	38.4

Table 4: Precision and Recall @1–3 of $\text{STIM}_{\max}$ for identifying the wrong token in erroneous reasoning steps.

the minimum of the two. The final score is the average Precision@k across all evaluation examples in each task and each distribution.

Recall@k measures whether the wrong tokens appear within the top-k tokens with highest memorization scores. It is defined as:

$$Recall@k = \begin{cases} 1, & \text{if any of the wrong tokens exists in} \\ & \text{top-k highest memorization tokens} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

The final score is the average Recall@k across all examples in each task and distribution.

In Table 4 we report Precision and Recall @1-3 (1 meaning the highest memorization score token and 3 meaning the top 3 high memorization score token) of each task using $\textbf{STIM}_{max}$, as well as aggregated over all tasks. For Precision and Recall of individual components ($\textbf{STIM}_{local}$, $\textbf{STIM}_{long}$, $\textbf{STIM}_{mid}$) see Appendix F.

Token-level memorization scores meaningfully correlate with erroneous tokens For all 4 tasks, Precision and Recall@1–3 scores are well above random chance (calculation described in Appendix C), especially in more complex reasoning tasks like *Applied Math* and *Formula Calculation*. This indicates that tokens with higher memorization scores are indeed more likely to be error-inducing.

Complex tasks show higher Precision and Recall@k *Applied Math* and *Formula Calculation* consistently have higher precision and recall across all levels compared to *Counting* and *Capitalization*. This suggests that errors in complex reasoning tasks are more likely to be associated with memorization.

Computational Complexity of Wrong Token Identification with STIM The overall complexity of identifying wrong tokens using **STIM** is $O(mn)$, where m denotes the number of salient tokens selected by the token saliency method, and n is the number of alternative candidate tokens considered for each target token.

Precision and recall improve with higher *k*. Both precision and recall increase consistently from $k = 1$ to $k = 3$ across tasks and levels, indicating that while the top memorized token is not always the erroneous one, the true error is often among the top three. This suggests that our method can serve as an effective first-pass filter, narrowing down the set of candidate wrong tokens that require further verification.

7 Conclusion

Our paper presented **STIM**, a novel diagnostic framework for fine-grained, token-level identification of memorization in Chain-of-thought reasoning, capturing multiple sources of memorization by analyzing both the input prompt and generated context. Our evaluation across diverse CoT tasks shows that memorization intensifies with task complexity and long-tail distribution shift, often leading to errors – especially driven by local memorization. Beyond aggregate trends, **STIM** reliably identifies error-inducing tokens, with high memorization scores correlating with incorrect tokens across tasks. This predictive signal underscores the practical value of token-level memorization as a lens on reasoning failures. **STIM** offers a foundation for deeper insights into model behavior and toward developing more robust, genuinely reasoning-capable LLMs.

Limitations

One limitation of our framework lies in the choice of token saliency method. We adopt LERG, a perturbation-based approach, to identify influential tokens in mid- and long-range memorization due to its favorable compute efficiency. However, LERG may miss finer-grained influence patterns compared to more computationally intensive alternatives, such as gradient-based methods or causal tracing approaches, which could yield more precise attributions.

Another constraint is the use of a pre-trained PRM (Process Reward Model) as the step verifier for detecting reasoning errors. While the verifier achieves strong performance in practice, it is not perfect and may occasionally mislabel erroneous or correct steps. Incorporating stronger verification models or ensemble-based approaches could further improve the robustness of our identification pipeline.

Finally, our analysis is limited by the availability of open-source language models with fully indexed pretraining corpora. Currently, only a few models, such as OLMo and Pythia meet these requirements, but Pythia is too weak to complete many tasks in Chain-of-thought reasoning. Widely used models either lack dataset transparency or are not fully open, restricting broader applicability of our framework across model scales in our work.

Risk

Tracing of proprietary, sensitive information or PII. **STIM** might be used on mal-intentioned tasks, where researchers may prompt models produce sensitive information and then apply **STIM** to analyze which areas in the prompt contributes most for retrieving such information.

Environmental tax. Another potential risk is increasing environmental burdens because we extensively ping infinigram API when searching through pretraining corpora, leading to extra usage of electricity and power.

Use and Distribution

All data we collected through LLMs in our work are released publicly for usage and have been duly scrutinized by the authors. Our work does not collect information that can be used to identify individual people or contents that may be offensive.

Our framework **STIM** may only be used for analysis and examinations of language models follown the ethics guideline of the community. Using **STIM** on mal-intentioned tasks is a potential threat, but the authors strongly condemn doing so.

Acknowledgments

This research is supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via the HIATUS Program contract #2022-22072200006, the Defense Advanced Research Projects Agency with award HR00112220046, and NSF IIS 2048211. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. Huihan Li was supported by Amazon ML Fellowship from the USC + Amazon Center on Secure and Trusted Machine Learning.

References

Stella Biderman, Usvsn Prashanth, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, and Edward Raff. 2023. Emergent and predictable memorization in large language models. *Advances in Neural Information Processing Systems*, 36:28072–28090.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*.

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

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, and 1 others. 2023. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36:70293–70332.

Yihuai Hong, Dian Zhou, Meng Cao, Lei Yu, and Zhijing Jin. 2025. The reasoning-memorization interplay in language models is mediated by a single direction. *arXiv preprint arXiv:2503.23084*.

Brian Kenji Iwana, Syed Tahseen Raza Rizvi, Sheraz Ahmed, Andreas Dengel, and Seiichi Uchida. 2016.

Judging a book by its cover. *arXiv preprint arXiv:1610.09204*.

Mingyu Jin, Weidi Luo, Sitao Cheng, Xinyi Wang, Wenyue Hua, Ruixiang Tang, William Yang Wang, and Yongfeng Zhang. 2024. Disentangling memory and reasoning ability in large language models. *arXiv preprint arXiv:2411.13504*.

Huihan Li, Arnav Goel, Keyu He, and Xiang Ren. 2025. Attributing culture-conditioned generations to pretraining corpora. In *The Thirteenth International Conference on Learning Representations*.

Huihan Li, Yuting Ning, Zeyi Liao, Siyuan Wang, Xiang Li, Ximing Lu, Wenting Zhao, Faeze Brahman, Yejin Choi, and Xiang Ren. 2024a. In search of the long-tail: Systematic generation of long-tail inferential knowledge via logical rule guided search. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2348–2370.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024b. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2961–2984.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Jiacheng Liu, Sewon Min, Luke Zettlemoyer, Yejin Choi, and Hannaneh Hajishirzi. Infini-gram: Scaling unbounded n-gram language models to a trillion tokens. In *First Conference on Language Modeling*.

Siyu Lou, Yuntian Chen, Xiaodan Liang, Liang Lin, and Quanshi Zhang. 2024. Quantifying in-context reasoning effects and memorization effects in llms. *arXiv preprint arXiv:2405.11880*.

Ximing Lu, Melanie Sclar, Skyler Hallinan, Niloofar Mireshghallah, Jiacheng Liu, Seungju Han, Allyson Ettinger, Liwei Jiang, Khyathi Chandu, Nouha Dziri, and 1 others. 2024. Ai as humanity's salieri: Quantifying linguistic creativity of language models via systematic attribution of machine text against web text. *arXiv preprint arXiv:2410.04265*.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, and 1 others. 2024. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*.

R Thomas McCoy, Paul Smolensky, Tal Linzen, Jianfeng Gao, and Asli Celikyilmaz. 2023. How much do language models copy from their training data? evaluating linguistic novelty in text generation using raven. *Transactions of the Association for Computational Linguistics*, 11:652–670.

William Merrill, Noah A Smith, and Yanai Elazar. 2024. Evaluating n-gram novelty of language models using rusty-dawg. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14459–14473.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, and 1 others. 2024. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*.

Akshara Prabhakar, Thomas L Griffiths, and R McCoy. 2024. Deciphering the factors influencing the efficacy of chain-of-thought: Probability, memorization, and noisy reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3710–3724.

Kun Qian, Shunji Wan, Claudia Tang, Youzhi Wang, Xuanming Zhang, Maximillian Chen, and Zhou Yu. 2024. Varbench: Robust language model benchmarking through dynamic variable perturbation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16131–16161.

Eva Sánchez Salido, Julio Gonzalo, and Guillermo Marco. 2025. None of the others: a general technique to distinguish reasoning from memorization in multiple-choice llm evaluation benchmarks. *arXiv preprint arXiv:2502.12896*.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, and 1 others. 2024. Dolma: an open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15725–15788.

Yi-Lin Tuan, Connor Pryor, Wenhu Chen, Lise Getoor, and William Yang Wang. 2021. Local explanation of dialogue response generation. *Advances in Neural Information Processing Systems*, 34:404–416.

Xinyi Wang, Antonis Antoniades, Yanai Elazar, Alfonso Amayuelas, Alon Albalak, Kexun Zhang, and William Yang Wang. 2024. Generalization vs memorization: Tracing language models' capabilities back to pretraining data. *arXiv preprint arXiv:2407.14985*.

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

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. 2024. Reasoning or reciting? exploring the capabilities and limitations of language

models through counterfactual tasks. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1819–1862.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. 2024. On memorization of large language models in logical reasoning. *arXiv preprint arXiv:2410.23123*.

Kayo Yin and Graham Neubig. 2022. Interpreting language models with contrastive explanations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 184–198.

Thomas Zeng, Shuibai Zhang, Shutong Wu, Christian Classen, Daewon Chae, Ethan Ewer, Minjae Lee, Heeju Kim, Wonjun Kang, Jackson Kunde, and 1 others. 2025. Versaprm: Multi-domain process reward model via synthetic reasoning data. *arXiv preprint arXiv:2502.06737*.

A Head and Long-tail Data of each reasoning task

Applied Math The head data is the original test set of GSM8K (Cobbe et al., 2021). The Long-tail transformation process includes identifying the numerical entities in the original problems, shifting the numerical entities, and recalculating the final result. Following Varbench (Qian et al., 2024), we get the extracted numerical variables that are identified by LLMs and verified by experts. Then we apply digit-expansion, int-to-float-conversion, and base-changing transformation to those entities. For digit-expansion, we randomly sample two prime numbers p_1, p_2 between 10 to 30, and convert the number entity var into $p_1 \times var + p_2$; for int-to-float conversion, we divide the original variable by 100; for base-changing, we convert the numbers into base-2, limiting 8 bits of precision for floating numbers. Finally , we use the solution function in Varbench to get the final result with shifted value for each numeric variable.

Formula Calculation The head data is the formulas extracted from GSM8K solutions. We combine all the partial formulas in the step-by-step solutions into the final compositional formula. Similar to Applied Math long-tail transformation process, we change each numerical variable of the left expression into numbers with larger magnitude, floating numbers and base-2 number, keeping them be equal to the correspondent value in the applied math problem.

Counting We use `gpt4-o` to generate multiple fruits and then select the fruits with pretraining frequency having order of magnitude $10^3, 10^4, 10^5, 10^6, 10^7$. Then we randomly combine them to form the counting list, with the length ranging from $10, 20, 30, 40, 50$. Each list only contains two types of fruits.

Capitalization We first filter the book title datasets (Iwana et al., 2016) by restricting the length equal to $3, 5, 7, 9, 11$ and then randomly sample 300 examples for each length group. Finally, we lower the character in the titles to get the original string and only capitalize the first letter of the last word of the original string to acquire the long-tail entity.

Number of examples for each reasoning task and settings are shown in Table 6.

Task	Base	Long-tail
Applied Math	**Question Prompt:** James gets 10 new CDs. Each CD cost $15. He gets them for 40% off. He decides he doesn't like 5 of them and sells them for 40. How much money was he out? Decimal integers with small magnitude and Base-10 calculation	**Question Prompt:** James gets 1010 new CDs. Each CD cost $1111. He gets them for 101000% off. He decides he doesn't like 101 of them and sells them for 101000. How much money was he out? Decimal numbers with large magnitude; floating number; Base-2 calculation
Formula Calculation	**Question Prompt:** What is 220 + 23 - 80 equal to? Decimal integers with small magnitude and Base-10 calculation	**Question Prompt:** What is 2437 + 270 - 897 equal to? Decimal numbers with large magnitude; floating number; Base-2 calculation
Counting	**Question Prompt:** Here is a list: [apple, pear,$\cdots$, pear]. How many times does 'apple' appear on it? Fruit's pretraining frequency $\geq 10^5$ and list length ≤ 20	**Question Prompt:** Here is a list: [keule, ugli,$\cdots$, keule]. How many times does 'ugli' appear on it? Fruit's pretraining frequency $< 10^5$ or list length > 20
Capitalization	**Question Prompt:** Here is a string: "history and obstinacy". Change the format of the string so that it can be a title. Existing book titles and capitalize into title format	**Question Prompt:** Here is a string: "reasons to live". Change the format of the string so that only the first letter of the last word is capitalized. Existing book titles and capitalize first letter of last word

Table 5: Illustrative prompts contrasting standard (Base) and more complex (Long-tail) versions of reasoning tasks, with grey annotations explaining the difference in difficulty.

Task	Distribution	# Examples
Applied Math	Base (GSM8K)	1319
	Long-tail (Digit Expansion)	1319
	Long-tail (Integer to Float)	1319
	Long-tail (Base 2)	1319
Formula Calculation	Base (GSM8K)	1314
	Long-tail (Digit Expansion)	1314
	Long-tail (Integer to Float)	1314
	Long-tail (Base 2)	1314
Counting	Base (Length ≤ 20 & frequent entities)	6000
	Long-tail (Length > 20 & infrequent entities)	17500
Capitalization	Base (Cap Title)	1500
	Long-tail (Cap Last Word)	1500

Table 6: Number of Examples Per Task and Distributional Setting

B Identifying Wrong Tokens with `gpt4-o`

We use `gpt4-o` to identify the wrong tokens in model's erroneous reasoning step for all tasks. The detailed prompts are shown in Table 7.

C Calculating Random Chance Baseline for Precision@k and Recall@k

For precision@k, we calculate all the possible k tokens' combinations from the wrong reasoning steps, which forms C_n^k sets $\mathcal{S}_i$, where n is number of candidate tokens. Then we calculate the proportion of wrong tokens in the $\mathcal{S}_i$ for each combination. Finally, we implement the average across all combinations:

$$P@k_{random} = \frac{1}{C_n^k} \sum_{i=1}^{C_n^k} \frac{\sum_{\texttt{word} \in \mathcal{W}} \mathbb{I}(\texttt{ token } \in \mathcal{S}_i)}{\min(|\mathcal{W}|, k)} \tag{6}$$

For recall@k, we calculate the proportion of randomly selected k tokens in the wrong token set. Denote n as the number of candidate tokens, m as the number of wrong tokens, then $R@k_{random}$ is defined as:

$$R@k_{random} = 1 - \frac{C_{n-m}^k}{C_n^k} \tag{7}$$

Prompt Template for gpt4-o wrong token identification

System message:
You are a helpful reasoning agent that can identify wrong tokens in another model's reasoning steps, by first generating your reasoning and then giving a final answer. You follow the examples given to you.
User message
Below you will see a reasoning question and a model's partial step-wise reasoning answer for it. The last step of the model's answer is wrong. Identify all the incorrect token(s) in that step that independently cause the step to be wrong, in "token, it's preceding token" format. Ignore the tokens whose error is caused by previous input tokens. You must give your answer from the given candidate tokens and choose at least one token. You will see two examples:
Example 1
Question: Last night Rick killed 1010 wolves and 1111 cougars while hunting. Today Rick killed 11 times as many wolves as cougars and 11 fewer cougars than the previous night. How many animals did Rick kill? (Calculate in Base-2)
Step-wise Reasoning: Rick killed 11 * 1111 = 12221 wolves today.
Candidate tokens: "Rick", preceded by ""; "killed", preceded by "Rick"; "11", preceded by "killed"; "*", preceded by "11"; "1111", preceded by "*"; "=", preceded by "1111"; "12221", preceded by "="; "wolves", preceded by "12221"; "today", preceded by "wolves"
Reasoning: The token 1111 is semantically incorrect because at this step the model first needs to calculate the number of cougars which is 1111 - 11 = 1100, which is the correct number here. The token 12221 is also semantically incorrect. [...] Thus, 12221 is the incorrect token that directly causes the step to be wrong.
Answer: "1111", preceded by "*"; "12221", preceded by "="

[...one more example]

Now it's your turn:
Question: Here is a list: [orange, banana, orange, banana, banana, banana, orange, orange, banana, orange]. How many times does 'orange' appear on it?
Step-wise Reasoning: To determine how many times the symbol 'orange' appears in the list [orange, banana, orange, banana, banana, banana, orange, orange, banana, orange], we can simply count each occurrence of 'orange'.
The first element is 'orange'.
The third element is 'orange'.
The seventh element is 'orange'.
The ninth element is 'orange'.
Candidate tokens: "ninth", preceded by "The"; "element", preceded by "is"; "'orange'", preceded by 'is'
Reasoning:

Table 7: Prompt of identifying wrong tokens for gpt4-o

D Reproducing Main Results on OLMo2 7B Instruct (`olmo-2-1124-7b-instruct`)

Although our model choice is constrained by available open-source models with fully indexed pretraining corpora, we reproduce our analysis on OLMo2 7B Instruct, a model trained on the same data as OLMo2 13B Instruct but only smaller in size. We select one task with high reasoning complexity (Applied Math) and one task with low reasoning complexity (Capitalization). All generation and evaluation settings remain the same.

In summary, our additional experiments in OLMo2 7B Instruct show that all our findings and performance of STIM are **generalizable to new models**, at least of the same architecture and different size.

D.1 Analyzing Memorization Patterns by Task, Distribution and Correctness (Section 5.2, Figure 3)

Figure 4 reproduces the original three findings of Section 5.2.

Original Finding 1: More complex reasoning tasks have higher memorization score The mean of memorization score for Applied Math (high complexity) is 0.463 while the mean of memorization score for Capitalization (low complexity) is 0.38.

Original Finding 2: Memorization scores are higher in long-tail settings The mean of memorization scores for base and long-tail tasks for Applied Math are 0.43 and 0.482; the mean of memorization scores for base and long-tail tasks for Capitalization are 0.346 and 0.415. For both tasks, the memorization scores are higher in long-tail settings.

Original Finding 3: Distribution Shift Reverses the Role of Memorization. The mean of memorization scores for correct and wrong examples in base tasks are 0.392 and 0.356, with correct examples higher; The mean of memorization scores for correct and wrong examples in long-tail tasks are 0.443 and 0.468, with wrong examples higher.

D.2 Dominant Source of Erroneous Memorization (Section 6, Table 3)

Table 8 shows the dominant memorization source (%) by task and distribution type. The two findings in the original papers both hold: 1) local memorization consistently emerges as the most frequent driver of erroneous token generation, although the actual percentage decreases (from 67% to 57%); this shows that smaller models may be less impacted by spurious short-range patterns than larger models 2) high reasoning task (Applied Math) show a notable decrease in the proportion of errors attributable to local memorization when shifting from base to long-tail distribution; low reasoning task (Capitalization) shows no reduction but rather some increase.

Task	Dist.	Local	Mid	Long
Applied Math	Base	**0.571**	0.242	0.187
	Longtail	**0.423**	0.317	0.260
Capitalization	Base	**0.476**	0.169	0.355
	Longtail	**0.515**	0.240	0.245

Table 8: Dominant memorization source (%) by task and distribution type for OLMo2 7B Instruct. Each row shows the percentage of erroneous tokens where each source (Local, Mid, Long) was the most influential.

D.3 Correspondence between high memorization and token error (Section 6, Table 4)

Table 9 shows the Precision@k and Recall@k performance for k=1,2,3.

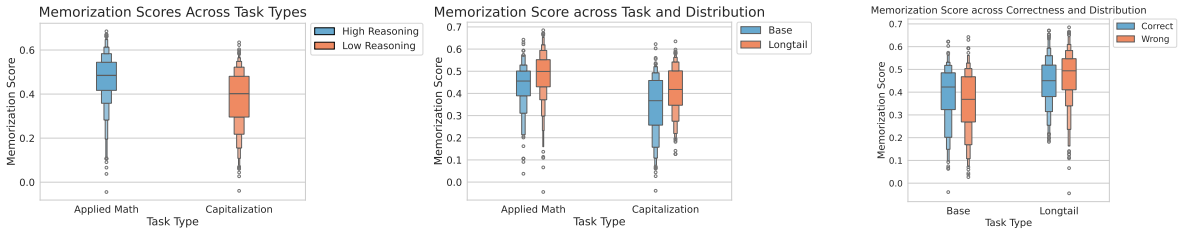
Original Finding 1: Token-level memorization scores meaningfully correlate with erroneous tokens For both tasks, Precision and Recall @1-3 are well above random chance. Compared to the original result on OLMo2 13B Instruct, the Precision@k scores and most of Recall@k scores increase.

Original Finding 2: Complex tasks show higher Precision and Recall@k This is true from the results above across k=1,2,3.

Original Finding 3: Precision and recall improve with higher k This is also true for both Applied Math and Capitalization tasks.

E Ablation Studies on Process Reward Models and Token Saliency Models

To further verify the reliability of VersaPRM in identifying genuinely incorrect reasoning steps, we calculated the proportion of examples where GPT-4o did not detect any erroneous tokens within the



(a) By Task (b) By Distribution (c) By Correctness

Figure 4: Comparison of STIM_{max} distribution between task reasoning complexity, distribution and correctness for OLMo2 7B Instruct.

Task	Level	1	2	3
Applied Math	STIM_{max}	0.460	0.513	0.574
	Random	0.173	0.279	0.317
Capitalization	STIM_{max}	0.225	0.318	0.465
	Random	0.089	0.154	0.229

(a) Precision@k

Task	Level	1	2	3
Applied Math	STIM_{max}	0.385	0.580	0.645
	Random	0.140	0.274	0.381
Capitalization	STIM_{max}	0.215	0.340	0.505
	Random	0.105	0.208	0.309

(b) Recall@k

Table 9: Precision and Recall @1–3 of STIM_{max} for identifying the wrong token in erroneous reasoning steps for OLMo2 7B Instruct.

steps selected by VersaPRM (as described in Section 6). These proportions were 2.0% for Applied Math and Formula Calculation, 0.5% for Counting, and 2.5% for Capitalization. Such low rates indicate that VersaPRM's selection of incorrect steps is reasonably robust.

To assess the sensitivity of our wrong-token prediction results to the choice of token saliency method other than LERG, we conducted an ablation study using an alternative approach based on contrastive explanations (Yin and Neubig, 2022) on 20 sampled incorrect examples per task. Contrastive explanations are valuable for highlighting why a token was chosen over a specific alternative. However, CE introduces instability due to challenging foil selection and incurs higher computational costs for language generation. Overall, LERG offers more consistent and efficient attribution in this setting. For this particular ablation, we select foil tokens using the alternative 5 tokens with highest token probability, at the same decoding step of the target token.

In Table 10 we report the Precision@k and Recall@k with each token saliency method.

For P@1, P@2, R@1, and R@2, STIM_{max} (LERG) performs similarly to or better than STIM_{max} (CE). For P@3 and R@3, the results are mixed, but the differences remain modest. These observations suggest that while the choice of token saliency method can influence STIM's predictive performance, the impact is limited. Finally, we would like to emphasize that the STIM framework is independent of the particular choice of PRM or token saliency method. Its overall predictive capacity could be further improved as more accurate PRMs for multi-domain reasoning and more advanced token-saliency techniques become available.

F Individual Component Analysis: STIM_{loc}, STIM_{mid} and STIM_{long}

Table 11 includes the predictive performance of each individual STIM score (local, mid, long) on wrong token identification and report their P@1 and R@1.

The results indicate that none of STIM_{local}, STIM_{mid}, or STIM_{long} consistently achieve the highest P@1 or R@1 across all three sources. For Applied Math and Capitalization, STIM_{local} shows the best performance, while for Formula Only and Counting, STIM_{long} performs best. This variation in performance across individual scores suggests that STIM_{max} is more robust overall. When aggregated across all tasks, STIM_{max} achieves the highest P@1 and R@1.

G Prompt for Direct Answer and Chain-of-thought format of each reasoning task

We use few-shot prompting methods for each task. The prompt formats in COT and Direct settings are shown in Table 12 to Table 25

Task	Method	P@1	P@2	P@3
Applied Math	$\text{STIM}_{\max}$ (CE)	0.400	0.450	0.583
	$\text{STIM}_{\max}$ (LERG)	0.400	0.450	0.575
	Random	0.225	0.318	0.317
Formula Only	$\text{STIM}_{\max}$ (CE)	0.350	0.475	0.533
	$\text{STIM}_{\max}$ (LERG)	0.400	0.500	0.533
	Random	0.256	0.442	0.445
Counting	$\text{STIM}_{\max}$ (CE)	0.100	0.325	0.325
	$\text{STIM}_{\max}$ (LERG)	0.100	0.325	0.475
	Random	0.082	0.156	0.235
Capitalization	$\text{STIM}_{\max}$ (CE)	0.200	0.300	0.500
	$\text{STIM}_{\max}$ (LERG)	0.250	0.350	0.450
	Random	0.096	0.179	0.268

(a) P@k

Task	Method	R@1	R@2	R@3
Applied Math	$\text{STIM}_{\max}$ (CE)	0.350	0.550	0.700
	$\text{STIM}_{\max}$ (LERG)	0.350	0.550	0.700
	Random	0.159	0.301	0.417
Formula Only	$\text{STIM}_{\max}$ (CE)	0.350	0.500	0.550
	$\text{STIM}_{\max}$ (LERG)	0.400	0.550	0.600
	Random	0.211	0.369	0.500
Counting	$\text{STIM}_{\max}$ (CE)	0.100	0.350	0.350
	$\text{STIM}_{\max}$ (LERG)	0.100	0.350	0.500
	Random	0.082	0.163	0.244
Capitalization	$\text{STIM}_{\max}$ (CE)	0.200	0.300	0.500
	$\text{STIM}_{\max}$ (LERG)	0.250	0.350	0.450
	Random	0.121	0.241	0.357

(b) R@k

Table 10: Precision (P@k) and Recall (R@k) at rank k across tasks and methods, ablating on CE and LERG.

Task	Method	P@1
Applied Math	Local	0.440
	Mid	0.355
	Long	0.410
	$\text{STIM}_{\max}$	0.455
	Random	0.158
Formula Only	Local	0.375
	Mid	0.400
	Long	0.435
	$\text{STIM}_{\max}$	0.410
	Random	0.209
Counting	Local	0.195
	Mid	0.240
	Long	0.240
	$\text{STIM}_{\max}$	0.215
	Random	0.103
Capitalization	Local	0.180
	Mid	0.155
	Long	0.150
	$\text{STIM}_{\max}$	0.175
	Random	0.100
All Tasks	Local	0.298
	Mid	0.288
	Long	0.309
	$\text{STIM}_{\max}$	0.314
	Random	0.143

(a) P@1

Task	Method	R@1
Applied Math	Local	0.400
	Mid	0.305
	Long	0.350
	$\text{STIM}_{\max}$	0.405
	Random	0.129
Formula Only	Local	0.345
	Mid	0.375
	Long	0.400
	$\text{STIM}_{\max}$	0.375
	Random	0.155
Counting	Local	0.185
	Mid	0.220
	Long	0.220
	$\text{STIM}_{\max}$	0.200
	Random	0.130
Capitalization	Local	0.180
	Mid	0.155
	Long	0.145
	$\text{STIM}_{\max}$	0.175
	Random	0.110
All Tasks	Local	0.278
	Mid	0.264
	Long	0.279
	$\text{STIM}_{\max}$	0.289
	Random	0.131

(b) R@1

Table 11: Precision (P@1) and Recall (R@1) across tasks and methods.

Prompt Template for Applied Math, Base, CoT

```
Instruction: Answer the given question. You will end your response with a
sentence in the format of 'So the answer is <number>.'
Question: There are 15 trees in the grove. Grove workers will plant trees
in the grove today. After they are done, there will be 21 trees. How many
trees did the grove workers plant today?
Answer: There are 15 trees originally. Then there were 21 trees after some
more were planted. So there must have been 21 - 15 = 6. So the answer is
6.

[...7 more examples]

Instruction: Answer the given question. You will end your response with a
sentence in the format of 'So the answer is <number>.'
Question: John drives for 3 hours at a speed of 60 mph and then turns
around because he realizes he forgot something very important at home. He
tries to get home in 4 hours but spends the first 2 hours in standstill
traffic. He spends the next half-hour driving at a speed of 30mph, before
being able to drive the remaining time of the 4 hours going at 80 mph. How
far is he from home at the end of those 4 hours?
Answer:
```

Table 12: Prompt of calculating math word problems (CoT setting)

Prompt Template for Applied Math, Longtail, CoT

```
Instruction: Assuming that all numbers are in base-2 where the digits
are "01". Answer the given question. You will end your response with a
sentence in the format of 'So the answer is <number>.'
Question: There are 1111 trees in the grove. Grove workers will plant
trees in the grove today. After they are done, there will be 10101 trees.
How many trees did the grove workers plant today?
Answer: There are 1111 trees originally. Then there were 10101 trees after
some more were planted. So there must have been 10101 - 1111 = 110. So
the answer is 110.

[...7 more examples]

Instruction: Assuming that all numbers are in base-2 where the digits
are "01". Answer the given question. You will end your response with a
sentence in the format of 'So the answer is <number>.'
Question: John drives for 11 hours at a speed of 111100 mph and then turns
around because he realizes he forgot something very important at home. He
tries to get home in 100 hours but spends the first 10 hours in standstill
traffic. He spends the next half-hour driving at a speed of 11110mph,
before being able to drive the remaining time of the 100 hours going at
1010000 mph. How far is he from home at the end of those 100 hours?
Answer:
```

Table 13: Prompt of implementing base-2 calculation in math word problems (CoT setting)

Prompt Template for Formula Calculation, Base, Direct

```
Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <number>.'
Question: What is 32 + 42 - 35 equal to?
Answer: So the answer is 39.

[...7 more examples]

Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <number>.'
Question: What is (16 - 3 - 4) * 2 equal to?
Answer:
```

Table 14: Prompt of formula calculation for base-10 (Direct setting)

Prompt Template for Formula Calculation, Longtail, Direct

```
Instruction: Assuming that all numbers are in base-2 where the digits are "01". Answer the given question. You will end your response with a sentence in the format of 'So the answer is <number>.'
Question: What is 100000 + 101010 - 100011 equal to?
Answer: So the answer is 100111.

[...7 more examples]

Instruction: Assuming that all numbers are in base-2 where the digits are "01". Answer the given question. You will end your response with a sentence in the format of 'So the answer is <number>.'
Question: What is (10010110 - 111100 / 1100100 * 10010110) * 110 equal to?
Answer:
```

Table 15: Prompt of formula calculation in base-2 (Direct setting)

Prompt Template for Formula Calculation, Base, CoT

```
Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <number>.'
Question: What is 32 + 42 - 35 equal to?
Answer: To calculate 32 + 42 - 35, we need to first calculate 32 + 42. 32 + 42 = 74. Then we need to calculate 74 - 35. 74 - 35 = 39. So the answer is 39.

[...7 more examples]

Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <number>.'
Question: What is (16 - 3 - 4) * 2 equal to?
Answer:
```

Table 16: Prompt of formula calculation in base-10 (CoT setting)

Prompt Template for Formula Calculation, Longtail, CoT

Instruction: Assuming that all numbers are in base-2 where the digits are "01". Answer the given question. You will end your response with a sentence in the format of 'So the answer is <number>.'
Question: What is 100000 + 101010 - 100011 equal to?
Answer: To calculate 100000 + 101010 - 100011, we need to first calculate 100000 + 101010. 100000 + 101010 = 1001010. Then we need to calculate 1001010 - 100011. 1001010 - 100011 = 100111. So the answer is 100111.

[...7 more examples]

Instruction: Assuming that all numbers are in base-2 where the digits are "01". Answer the given question. You will end your response with a sentence in the format of 'So the answer is <number>.'
Question: What is (10010110 - 111100 / 1100100 * 10010110) * 110 equal to?
Answer:

Table 17: Prompt of formula calculation in base-2 (CoT setting)

Prompt Template for Counting, Base, Direct

Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <number>.'
Question: Here is a list: [orange, orange, orange, orange, orange, orange, orange, apple, orange, orange]. How many times does 'apple' appear on it?
Answer: So the answer is 1.

[...7 more examples]

Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <number>.'
Question: Here is a list: [pear, pear, pear, pear, pear, pear, pear, apple, pear, pear]. How many times does 'apple' appear on it?
Answer:

Table 18: Prompt of counting common fruits (Direct setting)

Prompt Template for Counting, Longtail, Direct

Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <number>.'
Question: Here is a list: [pitanga, pitanga, pitanga, pitanga, pitanga, pitanga, pitanga, ugli, pitanga, pitanga]. How many times does 'ugli' appear on it?
Answer: So the answer is 1.

[...7 more examples]

Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <number>.'
Question: Here is a list: [ugli, ugli, ugli, pitanga, pitanga, pitanga, ugli, ugli, pitanga, pitanga]. How many times does 'ugli' appear on it?
Answer:

Table 19: Prompt of counting uncommon fruits (Direct setting)

Prompt for Counting, Base, CoT

Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <number>.'
Question: Here is a list: [orange, orange, orange, orange, orange, orange, orange, apple, orange, orange]. How many times does 'apple' appear on it?
Answer: Let's think step by step. To determine how many times the symbol 'apple' appears in the list [orange, orange, orange, orange, orange, orange, orange, apple, orange, orange], we can simply count the occurrences of 'apple' within the list. Looking at the list, we see: - There are eight 'orange' symbols. - There is one 'apple' symbol. So, 'apple' appears once in the list. So the answer is 1.

[...7 more examples]

Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <number>.'
Question: Here is a list: [apple, apple, apple, orange, orange, orange, orange, apple, orange, orange]. How many times does 'apple' appear on it?
Answer:

Table 20: Prompt of counting common fruits (CoT setting)

Prompt Template for Counting, Longtail, COT

Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <number>.'
Question: Here is a list: [pequi, pequi, pequi, pequi, pequi, pequi, pequi, keule, pequi, pequi]. How many times does 'keule' appear on it?
Answer: Let's think step by step. To determine how many times the symbol 'keule' appears in the list [pequi, pequi, pequi, pequi, pequi, pequi, pequi, keule, pequi, pequi], we can simply count the occurrences of 'keule' within the list. Looking at the list, we see: - There are eight 'pequi' symbols. - There is one 'keule' symbol. So, 'keule' appears once in the list. So the answer is 1.

[...7 more examples]

Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <number>.'
Question: Here is a list: [keule, pequi, pequi, keule, pequi, keule, keule, pequi, keule, pequi]. How many times does 'keule' appear on it?
Answer:

Table 21: Prompt of counting uncommon fruits (CoT setting)

Prompt Template for Capitalization, Base, Direct

Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <string>.'
Question: Here is a string: "cartoons for victory". Change the format of the string so that it can be a title.
Answer: So the answer is Cartoons for Victory.

[...Four more examples]

Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <string>.'
Question: Here is a string: "simple explanation of work ideas". Change the format of the string so that it can be a title.
Answer:

Table 22: Prompt of changing title in capitalization task (Direct setting)

Prompt Template for Capitalization, Longtail, Direct

```
Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <string>.'
Question: Here is a string: "cartoons for victory". Change the format of the string so that only the first letter of the last word is capitalized.
Answer: So the answer is cartoons for Victory.

[...Four more examples]

Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <string>.'
Question: Here is a string: "simple explanation of work ideas". Change the format of the string so that only the first letter of the last word is capitalized.
Answer:
```

Table 23: Prompt of capitalizing the first letter of the last word in the string (Direct setting)

Prompt Template for Capitalization, Base, CoT

```
Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <string>.'
Question: Here is a string: 'cartoons for victory'. Change the format of the string so that it can be a title.
Answer: Let's think step by step. To convert the string into a proper title, we need to capitalize the major words, but do not capitalize short conjunctions unless they are the first or last word. We can traverse each word iteratively. 'cartoons' becomes 'Cartoons'. 'for' becomes 'for'. 'victory' becomes 'Victory'. So the answer is Cartoons for Victory.

[...Four more examples]

Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <string>.'
Question: Here is a string: the roots of hinduism: the early aryans and the indus civilization'. Change the format of the string so that it can be a title.
Answer:
```

Table 24: Prompt of changing title in capitalization task (CoT setting)

Prompt Template for Capitalization, Longtail, CoT

Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <string>.'
Question: Here is a string: "cartoons for victory". Change the format of the string so that only the first letter of the last word is capitalized.
Answer: Let's think step by step. The last word in the string is 'victory', so we capitalize the first letter of it and it becomes 'Victory'. So the answer is cartoons for Victory.

[...Four more examples]

Instruction: Answer the given question. You will end your response with a sentence in the format of 'So the answer is <string>.'
Question: Here is a string: "the roots of hinduism: the early aryans and the indus civilization". Change the format of the string so that only the first letter of the last word is capitalized.
Answer:

Table 25: Prompt of capitalizing the first letter of the last word in the string (CoT setting)