STaD: Scaffolded Task Design for Identifying Compositional Skill Gaps in LLMs

Anonymous ACL submission

Abstract

Benchmarks are often used as a standard to understand LLM capabilities in different domains. However, aggregate benchmark scores provide limited insight into compositional skill gaps of LLMs and how to improve it. To make these weaknesses visible, we propose **S**caffolded **Ta**sk **D**esign (STaD) framework. STaD generates controlled variations of benchmark tasks based on the concept of *scaffolding*, which introduces structured, incremental support in a step-by-step manner. Rather than inspecting failures individually, this approach enables systematic and scalable probing of model behavior by identifying the specific reasoning skill compositions they lack. Treating the LLM as a black box, our experiments on six models of varying sizes reveal multiple failure points in three reasoning benchmarks and highlight each model's unique and distinct skill gaps.

1 Introduction

Benchmarks are standardized tests designed to evaluate the strengths and limitations of large language models (LLMs) across domains. However, most existing evaluations often treat complex tasks as monolithic (Ribeiro et al., 2020; Huang and Chang, 2023). Much like a single math test score, an aggregate benchmark score can tell you that a model is "bad at math" without explaining why. Two models may achieve the same overall accuracy on the same benchmarks yet have entirely different skill gaps: one may lack a specific reasoning ability, another may struggle when multiple skills must be combined, and a third may have the necessary knowledge but fail to situate the knowledge in the context of the task.

Understanding such skill gaps requires examining a model's reasoning steps. Several existing approaches aim to make model reasoning more explicit. For instance, Chain-of-Thought (CoT) prompting explicitly elicits intermediate reasoning steps (Wei et al., 2022). While effective for exposing reasoning traces, these methods are difficult to scale or analyze systematically (Huang and Chang, 2023) and may produce unfaithful explanations that do not reflect the model's actual underlying reasoning process (Lanham et al., 2023).

A common theme among methods that elicit reasoning in LLMs is task decomposition (Khot et al., 2022; Dua et al., 2022; Prasad et al., 2024; Jiang et al., 2023; Laban et al., 2025), which breaks complex tasks into simpler sub-tasks and makes intermediate reasoning steps more explicit and configurable. These approaches have primarily been used to help models solve complex tasks or to study model behavior step by step. However, it has rarely been applied as a diagnostic tool to uncover failure patterns.

From a cognitive science perspective, competence is not only what a learner or a model can do independently, but also what it can achieve with appropriate guidance (Vygotsky, 1978). *Scaffolding* provides such targeted support, gradually removed as proficiency develops (Van de Pol et al., 2010). We adapt this idea to LLM evaluation, using scaffolding not to improve performance, but as a diagnostic tool to reveal where and why models fail.

In this work, we propose **S**caffolded **Ta**sk **D**esign (STaD) framework to combine task decomposition and scaffolding to systematically identify the steps at which LLMs fail and the specific reasoning skill compositions they lack. In a nutshell, STaD breaks down benchmark tasks into sub-tasks representing individual reasoning steps; it then creates controlled variations of the original task by providing scaffolding at specific sub-tasks. By measuring performance across scaffolded variations, STaD can pinpoint exactly where the model struggles.

For example, consider a model that fails on the original task. If scaffolding is applied to the first sub-task and the model still fails, the limitation

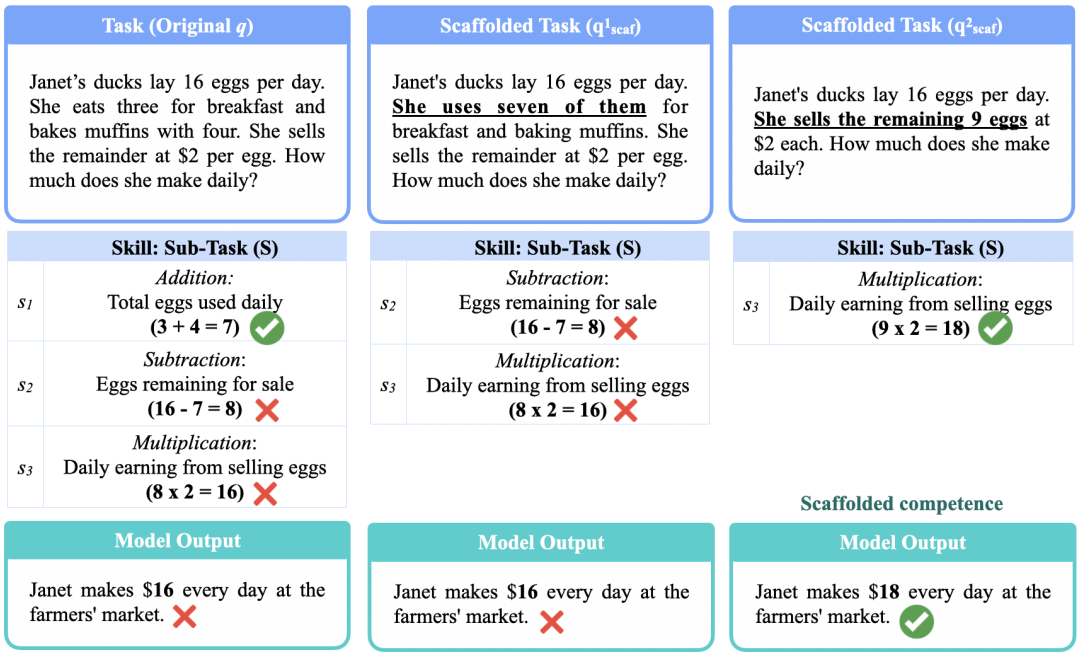


Figure 1: Scaffolded tasks for identifying compositional skill gap *s2* (*Subtraction*: Eggs reamining for sale).

is not confined to that step. If scaffolding is additionally applied to the second sub-task and the model succeeds, we can infer that the weakness lies in reasoning up to the second step. This procedure reveals compositional skill gaps—the particular combination of reasoning skills that the model cannot yet perform independently. An example of scaffolded tasks is shown in Figure 1.

Using this framework, we conduct experiments on three public benchmarks: ToT Arithmetic (Temporal Reasoning) (Fatemi et al., 2024), GSM8K (Grade School Math) (Cobbe et al., 2021), and Math-Hard (Difficult Math Problems) (Hendrycks et al., 2021). Our results show that failures are multifaceted: models with similar overall performance can exhibit very different bottlenecks, some skills are present but fail in context, and compositional skill interaction frequently create challenges. Scaffolding further distinguishes tasks that can reliably solved with targeted support from those that remain fundamentally intractable. STaD provides actionable insights, including: (1) Identifying skill combinations that cause failures; (2) Distinguishing weaknesses that can be mitigated with scaffolding from those that are fundamentally difficult; (3) Highlighting skill-level areas for further training or targeted interventions.

Our main contributions are threefold:

- **Concept**: We formalize *scaffolded competence* for LLMs, capturing not only independent task performance but also the abilities a model can demonstrate with support. This perspective enables a understanding of a model's situated skills within the context of real-world tasks.
- **Framework**: We introduce **S**caffolded **Ta**sk **D**esign (STaD) framework that systematically identifies reasoning and compositional skill gaps in LLMs. The codebase and datasets are provided as supplementary material and will be made publicly available for reproducibility and future research.
- **Empirical Study**: We apply our framework to conduct experiments on three reasoning benchmarks. Our results revealed distinct skill gaps across models and highlighting the value of detailed evaluation beyond aggregate scores.

2 Scaffolded Task Design (STaD)

Given a benchmark, our objective is to design a framework to construct cumulative scaffolded variations of the benchmark tasks that incrementally support reasoning steps. As we will demonstrate, the scaffolded variation enable us to reveal latent competence and identifying compositional skill gaps in the LLM.

At a high level, our **S**caffolded **Ta**sk **D**esign (STaD) framework generates, for each question with ground-truth answer in the benchmark, generates: (1) decomposed reasoning steps with intermediate answers and (2) scaffolded variations

of the original task. STaD then evaluates model performance on both the original and scaffolded test cases to derive a *scaffolded competence* and the *minimal level of scaffolding* required for a model to succeed. We provide the details in the following.

STaD begins by defining a benchmark as a set of questions and corresponding ground-truth answers: $Q = \{q_1, q_2, \ldots, q_n\}$, and $A = \{a_1, a_2, \ldots, a_n\}$, respectively. STaD uses three functional LLM roles in our framework.

Teacher model. The teacher model, f_{teach}, is a high-capacity LLM used to generate scaffolded variations including: (i) decomposing q into minimal reasoning units, (ii) computing intermediate answers, and (iii) producing scaffolded variations.

Judge model. The judge model, f_{judge}, evaluates correctness for both intermediate and final predictions by comparing model outputs to the ground truth a. It returns a binary correctness.

Target model. The target model, f_{target}, is the model we aim to diagnose. It is evaluated on both the original problem q and the generated scaffolded variations q_{scaf}.

2.1 Generating Test Cases

Many benchmark tasks consist of multiple smaller reasoning steps. Decomposing a task into such smaller reasoning steps allows us identify exactly where the model fails. For each task in a benchmark, STaD defines its sub-tasks along with the corresponding intermediate answers, associated higher-level skills, and the set of scaffolding variations as follows.

Task Decomposition. STaD uses the teacher model f_{teach} to decompose the original task q into a sequence of K sub-tasks $q \rightarrow S = \{s_1, s_2, \ldots, s_K\}$, where each s_k is an operation satisfying: s_k contains one logical or computational action; $\bigcup_{k=1}^{K} s_k$ covers all reasoning steps required by q; s_K produces the final answer.

For each sub-task s_k, STaD computes an intermediate result using the teacher model f_{teach}, producing $SA = \{sa_1, \ldots, sa_K\}$. STaD then compares the final sub-task answer sa_K with the ground truth a using the judge model f_{judge}. For this, we define a binary consistency function that returns 1 if the two values are identical and 0 otherwise:

$$\text{Cons}(sa_K, a) = \begin{cases} 1 & \text{if } sa_K = a, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Scaffolded Variations. STaD generates scaffolded variations of each original task q by progressively providing correct intermediate steps using f_{teach}. Specifically, for each $j \in \{1, \ldots, K-1\}$, STaD constructs a scaffolded variation $q^{(j)}_{\text{scaf}}$ by injecting the first j solved sub-tasks into the original task: $q^{(j)}_{\text{scaf}} = g(q,\ \{(s_1, sa_1), \ldots, (s_j, sa_j)\})$. Here, $g(\cdot)$ is a rewriting function that preserves the overall structure of the original task while explicitly providing the solutions to the first j reasoning steps. Each $q^{(j)}_{\text{scaf}}$ is presented as a single instruction indicating that earlier steps are already completed, leaving the model responsible for the remaining steps. This produces a set of scaffolded variations:

$$q_{\text{scaf}} = \{q^{(1)}_{\text{scaf}}, \ldots, q^{(K-1)}_{\text{scaf}}\}. \quad (2)$$

Verification. STaD verifies the consistency of all intermediate variations. For each scaffolded variation $q^{(j)}_{\text{scaf}}$, let $p^{(j)}_{teach} = f_{\text{teach}}(q^{(j)}_{\text{scaf}})$ denote the predictions from the teacher model. STaD computes binary consistency scores $c_j = \text{Cons}(p^{(j)}_{teach}, a)$, where $c_j = 1$ if the prediction matches a, and 0 otherwise (see (1)).

Note that, if all intermediate sub-task answers SA and variations q_{scaf} are correct, each variation should produce the same final answer as the original ground truth a. STaD retains a scaffolded variation for evaluation only if all consistency scores are 1, i.e., $c_j = 1, \forall j \in \{1, \ldots, K-1\}$.

Skill Mapping. To enable getting a higher-level understanding of sub-tasks, STaD maps each sub-task into a set of higher-level skills that reflect shared underlying cognitive or reasoning capabilities. STaD first embeds each task into a semantic vector space and cluster tasks using Agglomerative Clustering (a hierarchical similarity-based algorithm) (Pedregosa et al., 2011). From each cluster, it then selects a representative task whose embedding is closest to the cluster centroid. These representative tasks are then used by f_{teach} to generate a set of higher-level skills: $\mathcal{H} = \{h_1, h_2, \ldots, h_M\}$.

Next, STaD assigns each sub-task s_k of every task to one of the M skills in $\mathcal{H}$. This mapping allows us to abstract from individual sub-tasks to more general cognitive or reasoning skills, enabling evaluation of model performance both at the level of sub-tasks and at the level of reasoning skills.

2.2 Target Model Evaluation

Once the test cases are generated, we evaluate the target model f_{target} on (i) the original task q, and (ii)

a set of scaffolded variations q_{scaf} that incrementally provide guidance at intermediate reasoning steps. These test cases reveal which specific reasoning steps f_{target} handles successfully, and which steps it fails.

Original Task Evaluation. The target model predicts answers for the original benchmark questions: $P_{\text{orig}} = \{p_1, \ldots, p_n\}$. To assess correctness, STaDapplies the consistency function Cons to each prediction, producing an array of scores for all questions: $\text{score}_{\text{orig}} = \Big[\text{Cons}(p_i, a_i)\Big]_{i=1}^{n}$.

Scaffolded Task Evaluation. For each question q_i where the original prediction was incorrect (i.e., $\text{score}^i_{\text{orig}} = 0$), STaD uses the set of scaffolding variations $q^{(1)}_{\text{scaf}}$ to $q^{(K-1)}_{\text{scaf}}$ to find the skill gap.

To assess correctness, it applies the consistency function Cons to each prediction, producing an array of scores for all variations: $\text{score}_{\text{scaf}} = \Big[\text{Cons}(\hat{p}_i, a)\Big]_{i=1}^{K-1}$.

2.3 Minimum Scaffolding Level k

We define the *minimum scaffolding level* k for question q as the smallest number of cumulative scaffolding required for the model to answer a task correctly:

$$k = \min\Big\{j \mid \text{Cons}(\hat{p}^{(j)}, a) = 1 \wedge \text{Cons}(\hat{p}^{(l)}, a) = 0, l < j\Big\}.$$

If no scaffolded variation yields a correct prediction, we set $k = -1$. The minimum scaffolding level provides a quantitative measure of model competence: (i) Tasks are categorized as $k = 0$ if solvable independently, (ii) $k > 0$ if solvable with scaffolding, and (iii) $k = -1$ if unsolvable even with full scaffolding.

2.4 Individual Skill Testing as A Baseline

To assess the effectiveness of scaffolding, we test independent skills by evaluating each sub-task in isolation. Each sub-task is associated with a high-level skill, and the accuracy of each skill is computed as the proportion of its associated sub-tasks that are solved correctly. Low accuracy indicates consistent difficulty and high accuracy indicates proficiency.

3 Experiment Setup

Datasets. We perform skill gap analysis via scaffolding on three benchmark datasets: **ToT (Test of Time) Arithmetic** (Fatemi et al., 2024), **GSM8K (Grade School Math 8K)** (Cobbe et al., 2021), and **Math-Hard** (Hendrycks et al., 2021). We focus on these benchmark datasets since they are popular to assess reasoning in small LMs, these benchmarks have varying difficulty levels, and together they cover broad set of skills. While our study focuses on these datasets, STaD can technically be applied to any benchmark.

Table 1 reports the original and verified dataset sizes and the average number of sub-tasks per example (with min and max). As described in Section 2.1, each example is retained only if the teacher model produces the correct answer for all of its scaffolding variations. The final verified dataset size is indicated by the right arrow in the table.

Table 1: Overview of datasets: original size, final verified size, and average number of sub-tasks (min, max).

Dataset	Size	Avg Sub-Tasks
ToT	1.85K→1.45K	4.59 (2, 6)
GSM8K	1.31K→1.17K	3.92, (2, 7)
Math-Hard	1.32K→773	5.48 (2, 6)

Models. We used GPT-OSS-120B (Agarwal et al., 2025) as the teacher model to generate test cases as it performs well on all three benchmarks. GPT-OSS-120B successfully solved all tasks included in Table 1 (1.45K, 1.17K, and 773 tasks, respectively). We used Llama-3.3-70B (Grattafiori et al., 2024) and Mistral-Large (Karamcheti et al., 2021) as judge models, for verification and evaluation, respectively. To identify skill gaps, we evaluated six target models of varying sizes: Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct (Yang et al., 2025), Granite-3.3-2B-Instruct, Granite-3.3-8B-Instruct (Granite Team, 2024), Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct (Grattafiori et al., 2024).

Implementation Details. We selected the *right* level of granularity in terms of the number of representative tasks and number of high-level skills via careful ablation experiments (see Appendix C). For ToT Arithmetic, we selected 40 representative questions per cluster and identified 20 skills; for GSM8K, 40 questions generated 40 skills; and for Math-Hard, 40 questions generated 20 skills. The full skill lists are provided in Appendix D.

For test case generation, we used a temperature of 0.2 for the teacher models to introduce mild generation diversity. All evaluations with the judge and target models were performed using greedy

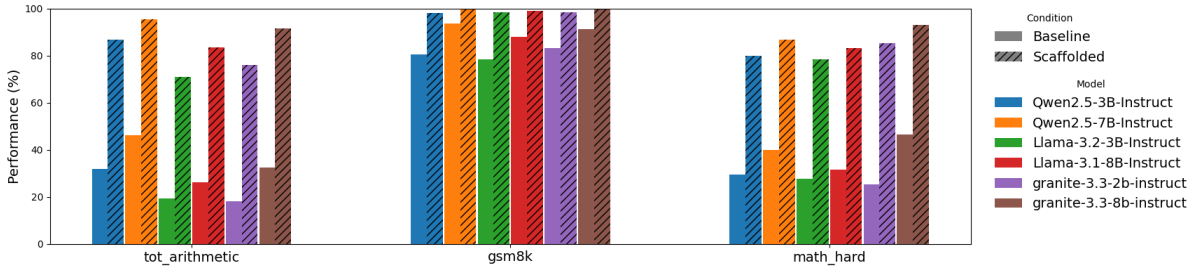


Figure 2: Original vs. scaffolded performance across benchmarks.

decoding to ensure deterministic and consistent evaluations across runs.

4 Results

4.1 Benchmark Evaluation

Figure 2 shows model performance on original tasks and under scaffolding across three benchmarks. A task is counted as successful if any scaffolded variant yields the correct answer. In the original setting, GSM8K performance is strong across models (78.4%-93.62%) compared to 18.16%-46.12% on ToT and 25.36%-46.44% on Math-Hard. Scaffolding consistently improves performance, yielding the largest gains on ToT (avg. +55.05%, min 49.31%, max 59.10%) and Math-Hard (avg. +50.97%, min 46.7%, max 59.75%), while GSM8K reaches near-perfect accuracy with smaller gains (avg. +13.07%, min 6.04%, max 19.97%). Despite these improvements, some tasks remain intractable; for example, even with scaffolding, Llama 3B retains a 29.01% gap on ToT and a 21.6% gap on Math-Hard, reflecting limitations beyond the skills captured in this study.

4.2 Individual Skill Gaps

Table 2 describes independent skill accuracy. Higher accuracy indicates stronger performance, while lower accuracy indicates skill deficiencies.

ToT Arithmetic Models generally perform well on *Structured JSON output* (0.38-0.70), *Arithmetic on durations* (0.42-0.71) and *Multi-format date conversion* (0.42-0.65). Models struggle on *Chronological ordering of events* (0.02-0.14), *Overlap and intersection of time* (0.11-0.22), and *Calendar arithmetic* (0.16-0.23).

GSM8K Models perform strongly across most math skills needed for GSM8K. For Granite2B, only one skill gap appears—*Extracting quantitative data from text*—while all other skills achieve accuracies above 0.5. The primary weaknesses are in *Extracting quantitative data from text* (0.11-0.29) and in *Formulating and solving linear equations* (0.67-0.89). Other skills are generally well-handled by all models.

Math-Hard Skill performance on Math-Hard is less consistent across models. In some skills, Qwen models underperform relative to other families. Examples are *Domain and range of functions* (Qwen: 0.20, Llama and Granite: 0.49-0.55), *Completing the square for optimization* (Qwen: 0.12-0.20, Llama and Granite: 0.46-0.68) and more. *Symmetry counting with Burnside's Lemma* remained challenging for all models (0.18-0.35).

While useful in identifying individual skill gaps, individual skill testing does not reflect how the skill functions in the full context. Skills are not tested with other skills, nor within the situational dependencies of the original problem, so the results may give an optimistic view of actual task performance.

4.3 Single-skill and Compositional-skill bottlenecks

We define the minimum scaffolding set as a *bottleneck set*: the smallest set of reasoning steps that must be completed correctly for the model to make progress. Once these steps are resolved, the remaining steps can be completed on their own. In other words, the bottleneck represents the point in the reasoning steps that prevents the task from being solved. This conceptually answers, "Where does the LLM's reasoning first fail?"

For analysis, we normalize the data by collapsing duplicated skills (e.g., "Arithmetic on durations, Arithmetic on durations" to "Arithmetic on durations") since repeated scaffolding of the same skill reflects persistent difficulty with that skill rather than additional reasoning structure. Figure 3 shows the frequency of the top 5 bottlenecks: single-skill cases ($size = 1$) and compositional-skill cases ($size > 1$). Full results are provided in Table 7.

For ToT, *Unit conversions*, *Natural-language date/time parsing*, *Overlap and intersection of time*,

Table 2: Individual skill accuracy across models and datasets (top 10 most frequent skills). The two lowest-performing skills are highlighted in bold (≤ 0.5).

	Skill	Qwen3B	Qwen7B	Llama3B	Llama8B	granite2B	granite8B
ToT Arithmetic	Structured JSON output	0.55	0.70	0.38	0.42	0.53	0.66
	Arithmetic on durations	0.54	0.71	0.42	0.50	0.55	0.69
	Unit conversion for time spans	0.51	0.70	0.33	0.42	0.37	0.60
	Calendar arithmetic	0.18	0.22	0.19	**0.18**	0.16	0.23
	Natural-language date/time parsing	0.35	0.27	0.23	0.26	0.28	0.27
	Discrete slot counting	0.19	0.29	0.15	0.21	0.16	0.25
	Overlap and intersection of time	**0.11**	**0.18**	**0.13**	0.20	**0.11**	**0.22**
	Multi-format date conversion	0.62	0.65	0.42	0.52	0.55	0.64
	Day-of-week determination	0.44	0.37	0.27	0.36	0.28	0.43
	Chronological ordering of events	**0.02**	**0.04**	**0.08**	**0.14**	**0.08**	**0.10**
GSM8K	Sequential quantity tracking	0.70	0.77	0.74	0.70	0.66	0.77
	Using unit rates to compute totals	0.82	0.82	0.86	0.87	0.75	0.83
	Extracting quantitative data from text	**0.25**	**0.19**	**0.29**	**0.20**	**0.11**	**0.19**
	Summing contributions from multi sources	0.67	0.78	0.77	0.80	0.69	0.89
	Formulating and solving linear equations	**0.49**	**0.50**	**0.42**	**0.48**	0.51	**0.42**
	Multi-digit multiplication and division	0.90	0.85	0.84	0.94	0.73	0.83
	Proportional reasoning	0.77	0.76	0.77	0.85	0.57	0.79
	Calculating a percentage of a quantity	0.85	0.89	0.93	0.96	0.67	0.93
	Performing operations with fractions	0.70	0.80	0.73	0.69	0.60	0.58
	Multiplicative scaling for unknowns	0.81	0.86	0.76	0.73	0.74	0.75
Math-Hard	Multi-step integer/fraction arithmetic	0.45	0.51	0.55	0.60	0.56	0.63
	Translating problems into algebra	**0.24**	**0.23**	**0.31**	**0.31**	**0.34**	**0.34**
	Solving linear equations and systems	0.29	0.31	0.41	0.45	0.46	0.47
	Combinatorial counting	0.36	0.46	0.44	0.42	0.43	0.47
	Quadratic equation techniques	0.30	0.40	0.44	0.51	0.50	0.42
	Modular arithmetic and congruence	0.34	0.44	0.40	**0.35**	**0.39**	0.54
	GCD/LCM via prime factorization	0.38	0.41	0.49	0.51	0.49	0.53
	Probability via counting	0.28	0.46	**0.39**	0.39	0.44	0.49
	Domain and range of functions	**0.20**	**0.20**	0.55	0.49	0.43	0.50
	Arithmetic and geometric series analysis	0.36	0.28	0.43	0.41	0.52	**0.35**

and *Calendar arithmetic* emerge as the most frequent bottlenecks across models. Combining these skills in pairs such as *Overlap/Discrete*, *Natural/-Calendar*, or *Unit/Arithmetic* are bottleneck for most models. For GSM8K, *Using unit rates to compute totals*, *Extracting quantitative data from narrative text*, *Sequential quantity tracking*, *Multi-digit multiplication and division*, and *Performing operations with fractions* are the most frequent skill bottlenecks. These bottlenecks are quite consistent across models. Only few combining bottlenecks were observed for this dataset.

4.4 Mean Minimum Scaffolding per Reasoning Combination

We evaluate skills within their original combinations. Specifically, tasks are grouped by their original skill combinations, and for each task STaD computes the minimum scaffolding level (k) required for the model to produce a correct solution (Section 2.3). Intuitively, the minimum scaffolding level (k) represents the least amount of external guidance needed for success; larger values of k therefore indicate greater difficulty. Table 3 reports results for the most frequent skill combinations in ToT benchmark.

The combination *Overlapping* + *Discrete slots* + *JSON* is consistently challenging across models. This is reflected in both high mean k values and a substantial fraction of intractable tasks. For example, Granite2B fails on 24.61% of these problems even with full scaffolding. Across models, mean k ranges from 2.51 to 3.06, indicating that solving these tasks typically requires two to three levels of explicit guidance (out of four available scaffolding). In practice, this typically means providing 2-3 levels of guidance—for example, first guiding the model through *Overlapping*, then through *Discrete Meeting slots*—after which it can usually handle the remaining *Discrete Meeting slots* and *JSON* components on its own. In contrast, the combination *Unit* + *Arithmetic* + *Arithmetic* + *Unit* + *JSON* reveals model-dependent skill gaps. Qwen3B, Qwen7B, and Granite8B handle this combination with relatively little assistance, exhibiting mean k values between 1.45 and 1.97 and near-zero intractable rates (0-1.63%). However, Llama3B, Llama8B, and Granite2B struggle substantially, with mean k values between 3.19 and 3.68 and intractable rates ranging from 27.86% to 59.01%. This divergence demonstrates that identical composite skills can be trivial for some models while remaining fundamentally difficult for others. A similar pattern is observed for the combination *Unit* + *Midnight* + *Arithmetic* + *Unit* + *JSON*. In GSM8K, most tasks are solved independently

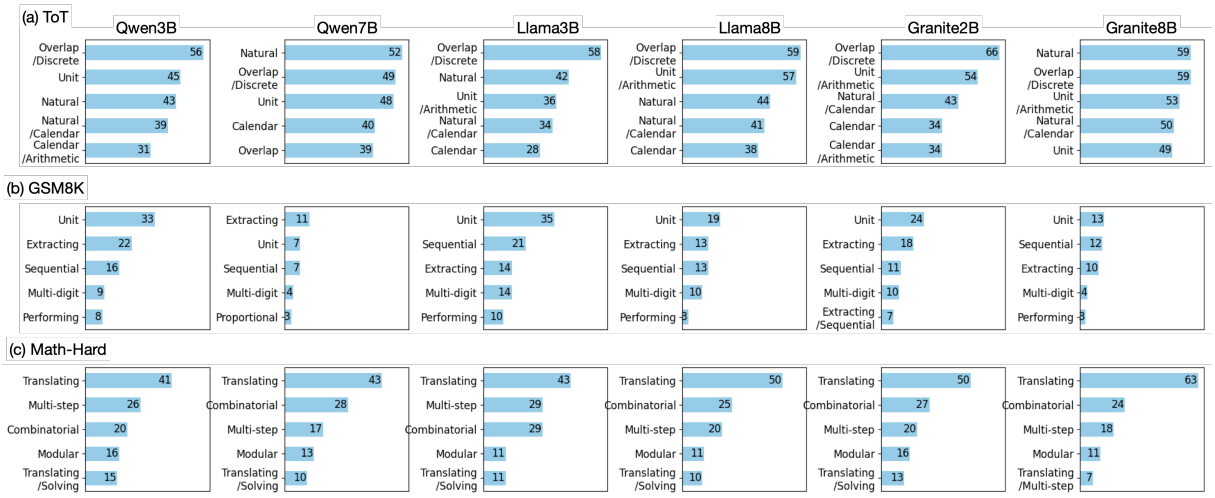


Figure 3: Frequency of compositional-skill bottlenecks in (a) ToT Arithmetic, (b) GSM8K, and (c) Math-Hard. For each model, the top 5 bottlenecks are displayed. Numbers indicate the count of scaffolded cases. Only the first word of each skill is shown.

without any scaffolding: over 90% of problems have $k = 0$ across all models, and almost none remain intractable once scaffolding is introduced. For GSM8K and Math-Hard, however, we were unable to collect enough tasks sharing identical skill combinations to perform a comparable combination-level analysis. This reflects the higher structural diversity of these benchmarks, where problems vary widely in composition rather than recurring around a small set of shared skill combinations.

5 Discussion

Existing probing methods, such as directly asking whether a model can perform a skill, abstract away the context in which a skill must be applied. In real tasks, competence is expressed through the coordinated deployment of multiple skills, situated within the specific demands of the task context. By preserving this context while selectively supporting intermediate reasoning stages, scaffolding enables direct evaluation of situated competence—the model's ability to apply a given skill appropriately within its original task setting.

Comparable performance does not imply identical skill gaps. While overall task scores may be similar, models often exhibit very different skill bottleneck profiles. For example, Llama3B and Granite2B achieve similar ToT performance (19.2 vs. 18.16), but Llama3B tends to require more support on isolated skills (e.g., *Natural-language date/time parsing* and *Calendar arithmetic*) whereas Granite2B's bottlenecks appear when multiple skills are combined (e.g., *Unit conversion / Arithmetic on durations*,*Relative-date reasoning / Calendar arithmetic / Multi-format date conversion*). Comparable trends appear in GSM8K. Qwen3B and Granite2B achieve similar overall accuracy (80.36 vs. 83.08), yet Qwen3B exhibits high individual skill averages (0.74 ± 0.17) while Granite2B shows lower individual skill competence (0.62 ± 0.17), particularly struggling with tasks like *Extracting quantitative data* from text that Qwen3B handles reliably.

Skill Present but Fail in Context. Performance on isolated skill probes systematically overestimates a model's actual reasoning competence when those same skills must be executed in combination. Several abilities that appear relatively well mastered in isolation, such as *Arithmetic on durations*(0.42-0.71), *Unit conversion for time spans* (0.33-0.70), and *Natural-language date/time parsing* (0.23-0.35), *Combinatorial counting* (0.36-0.47) do not emerge as dominant failure skill under individual skill testing. However, scaffolding analysis reveals that these skills become among the most frequent bottlenecks when embedded within tasks that requires sequential reasoning steps, both as single-skill and compositional-skill bottlenecks. More broadly, these results suggest that model reasoning weaknesses are not solely a function of missing skills, but of unstable skill integration. Models may possess the necessary primitives yet lack mechanisms for sequencing, maintaining intermediate representations, or reconciling constraints across skills.

Failures often arise from skill interactions. Compositional skill bottlenecks demonstrate that improving only the upstream skills is insufficient. For example, when *Overlap* and *Discrete* skills are

Table 3: Combinations with higher mean (k) require more explicit guidance. $k = 0$: tasks solved independently; % $k > 0$: % of tasks that can be solved w/ scaffolding; % *Intractable*: tasks that cannot be solved even w/ scaffolding.

Combination (ToT)	#Tasks	Model	Mean k	% $k = 0$	% $k > 0$	% Intractable
Overlapping + Overlapping + Discrete + Discrete + JSON	65	Qwen3B	3.02	6.15	70.76	23.07
		Qwen7B	2.51	6.15	89.23	4.61
		Llama3B	3.06	3.07	75.38	21.53
		Llama8B	2.69	10.76	81.53	7.69
		Granite2B	2.93	1.53	73.84	24.61
		Granite8B	2.79	10.76	81.53	7.69
Unit + Arithmetic + Arithmetic + Unit + JSON	61	Qwen3B	1.90	45.90	52.45	1.63
		Qwen7B	1.45	67.21	32.78	0.0
		Llama3B	3.68	9.83	31.14	59.01
		Llama8B	3.44	13.11	59.01	27.86
		Granite2B	3.19	9.83	59.01	31.14
		Granite8B	1.97	21.31	78.68	0.0
Unit + Midnight + Arithmetic + Unit + JSON	41	Qwen3B	2.32	4.87	90.24	4.87
		Qwen7B	2.18	34.14	65.85	0.00
		Llama3B	3.85	0.0	68.29	31.70
		Llama8B	3.60	0.0	80.48	19.51
		Granite2B	3.82	2.43	68.29	29.26
		Granite8B	2.48	9.75	85.36	4.87

scaffolded together, there are more successful cases than when only *Overlap* is supported. In particular, tasks that require *Overlap + Overlap + Discrete Meeting slots + Discrete Meeting slots + JSON* indicate that, on average, models require support beyond pure *Overlap* reasoning including *Discrete Meeting slots* skills. This suggests that, in compositional tasks, addressing only the upstream skills is unlikely to improve performance, and that failures often stem not from deficiencies in individual skills but from interactions among multiple skills.

Scaffolding Reveals Learnable vs. Intractable Tasks. Scaffolding analysis highlights which tasks can be solved when upstream skills are supported and which remain intractable despite full support. Successful scaffolded cases demonstrate that these tasks are learnable: when the model receives guidance for the necessary intermediate skills, it is able to reach the correct solution. Tasks that remain unsolved, such as those in ToT arithmetic and Math-Hard, point to deeper challenges beyond isolated skill gaps. One contributing factor is that scaffolding was provided only up to the final step, so failures may reflect difficulty in synthesizing intermediate results or completing the last step. Other failures may stem from challenges in understanding the original question or in planning a decomposition strategy, suggesting that intractable tasks require more than just learning individual skills. They demand higher-level reasoning to integrate multiple components.

Complementary Approaches for Diagnosing Model Skill Gaps. Individual skill testing and scaffolding serve complementary roles: the former identifies what a model can do in isolation, while the latter reveals how those capabilities interact during end-to-end reasoning. Decomposition is good for isolating specific skills or reasoning steps that are absent, offering a precise map of the task's underlying components and making localized skill deficits visible. Scaffolding, by contrast, situates these skills within the broader problem context, revealing not only whether a model can apply a skill in practice but also how multiple skills interact. Beyond assessing competence, scaffolding exposes which tasks remain fundamentally intractable and which can be resolved with targeted support, highlighting both the limits of model reasoning and the actionable pathways for improvement.

6 Conclusions

Through applying our framework to three benchmarks and six models, we discovered that not all reasoning failures in LLMs are created equal. Some models stumble on the basics: they cannot perform individual reasoning steps reliably. Other models, while capable of these individual skills, struggle when multiple skills must work together, revealing that composing knowledge is a challenge. Finally, some models appear competent on every step in isolation but fail when they need to apply that knowledge in context. By combining task decomposition with scaffolding, we were able to trace these different failure modes systematically, offering a lens to see beyond aggregate benchmark scores. These insights provide actionable insights: creating targeted synthetic data at the right level of abstraction, refining training strategies, and designing evaluation frameworks that emphasize both individual skills and their composition.

Limitations

Our approach relies on teacher models to generate scaffolded variations of tasks. As a result, the quality of our analysis depends on the teacher model's performance, though in practice we find this provides reliable guidance for multi-step reasoning tasks. We also acknowledge that there are multiple valid ways to decompose a task, and our method addresses only one particular decomposition. Different models or decomposition strategies could yield alternative sub-task sequences, which might lead to different insights about reasoning dependencies.

The method is most effective when multiple questions share similar reasoning steps, allowing aggregation and insight into typical failure modes. In some datasets, such as GSM8K and Math-Hard, there are fewer tasks with identical skill combinations, limiting combinatorial-level analysis. Additionally, our approach is designed for tasks that require multi-step reasoning, and is less applicable to very simple questions (e.g., "What is photosynthesis?") or single-step generation tasks (e.g., "Write a CV for a junior developer"), where sequential scaffolding offers limited benefit.

Importantly, our goal is not to pinpoint single-skill failures, but to understand how combinations of skills interact in reasoning. Cumulative scaffolding helps reveal which combinations are sufficient for success and where dependencies cause breakdowns. While later levels of scaffolding include prior support—making attribution to individual sub-skills less direct—this also allows us to capture higher-order interactions and error propagation effects that isolated evaluations miss.

Declaration on Generative AI

During the preparation of this work, the authors used GPT-5 mini to perform grammar and spelling checks, as well as to assist in generating tables and formatting listings. The authors reviewed and edited all content as needed and take full responsibility for the publication's content.

Acknowledgments

Anonymized.

References

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, and 1 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*.

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

Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. 2022. Successive prompting for decomposing complex questions. *arXiv preprint arXiv:2212.04092*.

Bahare Fatemi, Mehran Kazemi, Anton Tsitsulin, Karishma Malkan, Jinyeong Yim, John Palowitch, Sungyong Seo, Jonathan Halcrow, and Bryan Perozzi. 2024. Test of time: A benchmark for evaluating llms on temporal reasoning. *arXiv preprint arXiv:2406.09170*.

Matt Gardner, Yoav Artzi, Victoria Basmova, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, and 1 others. 2020. Evaluating models' local decision boundaries via contrast sets. *arXiv preprint arXiv:2004.02709*.

IBM Granite Team. 2024. Granite 3.0 language models. *URL: https://github.com/ibm-granite/granite-3.0-language-models*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

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

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the association for computational linguistics: ACL 2023*, pages 1049–1065.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2023. Followbench: A multi-level fine-grained constraints following benchmark for large language models. *Preprint*, arXiv:2310.20410.

Siddharth* Karamcheti, Laurel* Orr, Jason Bolton, Tianyi Zhang, Karan Goel, Avanika Narayan, Rishi Bommasani, Deepak Narayanan, Tatsunori Hashimoto, Dan Jurafsky, Christopher D. Manning, Christopher Potts, Christopher Ré, and Percy Liang. 2021. Mistral - a journey towards reproducible language model training.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2025. Llms get lost in multi-turn conversation. *arXiv preprint arXiv:2505.06120*.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, and 1 others. 2023. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, and 1 others. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. 2024. Adapt: As-needed decomposition and planning with language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4226–4252.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. *arXiv preprint arXiv:2005.04118*.

Janneke Van de Pol, Monique Volman, and Jos Beishuizen. 2010. Scaffolding in teacher–student interaction: A decade of research. *Educational psychology review*, 22(3):271–296.

Lev S Vygotsky. 1978. *Mind in society: The development of higher psychological processes*, volume 86. Harvard university press.

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.

Qwen: An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

A Background and Related Work

Existing benchmarks can be leveraged to gain finer-grained insights by analyzing model performance through skill decomposition and error pattern analysis. Skill decomposition involves breaking a benchmark into subskills or predefined capability categories—either based on the benchmark's domain or task design to identify which aspects a model handles well or poorly (Hendrycks et al., 2021; Gardner et al., 2020). Error pattern analysis examines the distribution of failures across these categories to uncover systematic biases, heuristics, or failure modes hidden by aggregate scores (McCoy et al., 2019). Alternatively, some benchmarks are designed to incrementally increase task complexity, enabling a more direct tracking of skill gaps as the model progresses through levels of difficulty (Jiang et al., 2023)

While model interpretability research often focuses on understanding internal model representations, systematic behavioral analysis systematically probe model behavior across controlled variations to understand functional capabilities and limitations. This treating knowledge/model as blackbox as observable input-output behavior. In this vein, researchers analyze benchmark performance patterns to detect systematic weaknesses, recurring error modes, or inconsistencies across tasks. Such analyses can highlight brittle reasoning, task-specific weaknesses, or generalization failures that may not be visible through internal inspection alone.

To diagnose reasoning failures more precisely, prior work has employed controlled variations of inputs. At the question level, this involves perturbing or rewriting inputs to test robustness and sensitivity. At the answer level, researchers compare model outputs to reference answers to identify consistent error patterns, such as partial reasoning failures, logical inconsistencies, or hallucinations. While this approach provides valuable diagnostic insight, it typically evaluates the task as a whole, which can make it difficult to pinpoint the specific reasoning steps where a model fails.

Building on behavioral and controlled probing approaches, recent work has explored task decomposition to more precisely isolate reasoning gaps (Khot et al., 2022; Dua et al., 2022; Prasad et al., 2024; Jiang et al., 2023; Laban et al., 2025). Many tasks involve multiple cognitive steps, and aggregate performance metrics often obscure which sub-components are responsible for failures.

B Dataset

We perform skill gap analysis via scaffolding on three benchmark datasets: **ToT (Test of Time) Arithmetic**, which evaluates temporal reasoning through arithmetic operations on times and durations (Fatemi et al., 2024); **GSM8K (Grade School Math 8K)**, consisting of grade-school math word problems (Cobbe et al., 2021); and **Math-Hard**, which comprises problems drawn from mathematics competitions, where only the hardest questions (Level 5) are retained (Hendrycks et al., 2021).

C Number of Clusters and Skills

We conducted ablation experiments to find how many representative tasks (clusters) (M) are sufficient to capture task diversity and how many higher-level skills (N) are needed to explain and predict model performance in a more abstract way. We tested the following values: $M \in \{5, 10, 20, 40, 80\}$ and $N \in \{5, 10, 20, 40\}$, resulting in 20 combinations.

We assessed the resulting clusters and skills based on the following criteria:

- **Skill granularity:** whether skills are too broad (providing little insight) or overly task-specific. Figure 6 illustrates skill granularity.
- **"Other" category coverage:** whether the defined skill set sufficiently captures the diversity of tasks. Figure 5 shows the coverage of the *other* category where lower values indicate better coverage of the task space.
- **Skill overlap:** cosine similarity between skills to identify redundancy and ensure distinctiveness. redundancy. Table 4 shows semantic overlap across different skill granularity.
- **Skill distribution:** whether skills are used evenly across tasks or dominated by a few. Higher entropy indicates more balanced assignment. Figure 4 describes skill distribution.

In the following subsection, we describe our analysis for each dataset according to these criteria and explain our decisions regarding the number of clusters and skills. The resulting skills are detailed in Appendix D.

C.1 ToT Arithmetic

Figure 5 shows the coverage of the *other* category where lower values indicate better coverage of the task space. Lower other category indicate better coverage of the task space. We observe that increasing the number of skills consistently reduces the fraction of tasks assigned to *other* However, the improvements exhibit diminishing returns beyond $N = 20$, with only marginal gains at $N = 40$. The lowest *other* category rates occur when $M = 20$ or $M = 40$. **Takeaway: The combinations** $(M, N) = (20, 20), (20, 40), (40, 20), (40, 40)$ **provide the most effective coverage.**

Figure 6 illustrates skill granularity. When $N = 5$, the skills are too high level. For example, a single high-level skill such as "duration arithmetic" covers three distinct subtasks. Increasing to $N = 10$ improves specificity, with skills like "constraint-driven scheduling slot enumeration," but they remain too high-level. At $N = 20$, skill granularity reaches an appropriate level: concrete computation and reasoning steps such as "overlap and intersection of time intervals," "discrete slot counting under constraints," and "structured JSON output generation" are well captured. The same trend holds for $N = 40$, which also effectively captures task skills without unnecessary granularity. **Takeaway:** skill sets of $N \geq 20$ provide sufficient granularity to represent task-level reasoning accurately.

Table 4 shows most skills remain distinct as indicated by low mean similarity. However, higher overlaps occur for larger N, especially with larger cluster sizes M. For example, combinations $(M, N) = (10, 40)$, $(20, 40)$, and $(40, 40)$ exhibit a notable fraction of skill pairs with similarity ≥ 0.78, suggesting some redundancy. **Takeaway:** Using $N = 40$ may introduce redundant skills; smaller skill sets are preferable to maintain distinctiveness.

Figure 4 describes skill distribution. As N increases, the distribution becomes more concentrated with a smaller subset of skills accounting for most task assignments. This trend is reflected in decreasing skill distribution entropy, indicating greater specialization. While specialization improves the ability to capture fine-grained reasoning steps, excessively large N produces a sparse, long-tailed distribution, where many skills are rarely used. In contrast, varying the number of clusters M has minimal impact on the overall distribution. **Takeaway:** Choosing very large N values can lead to underutilized skills and sparse coverage, so a moderate N balances granularity and effective usage.

Based on these observations, we select $N = 20$ **skills and** $M = 40$ **representative clusters** because this configuration (1) achieves low uncovered-task rates, (2) provides sufficiently granular and interpretable skills, and (3) maintains a well-distributed skill usage that avoids long-tailed sparsity.

C.2 GSM8K

In Figure 5, we observe that the values remain consistently low across most configurations. For smaller M (e.g., $M = 5$ or $M = 10$), increasing N beyond 10 can even lead to noticeable degradation. In contrast, $M = 20$ or $M = 40$ exhibit greater stability with minimal variation across different N and consistently lower rates. Increasing M to 80 does not yield further improvement. **Takeaway:** The combinations $(M, N) = (20, 20)$, $(20, 40)$, and $(40, 20)$ provide the most stable and consistently low values.

Figure 7 illustrates skill granularity. When $N = 5$ and $N = 10$, the skills are too high level. For example, a single high-level skill such as "Sequential quantity tracking" covers three distinct subtasks. Increasing to $N = 20$ improves specificity, with skills like "Basic-arithmetic operation," but they remain too high-level. At $N = 50$, skill granularity reaches an appropriate level: concrete computation and reasoning steps such as "determining remaining quantity after consumption an distribution," "using unit rates to compute totals" are well captured. **Takeaway:** skill sets of $N \geq 40$ provide sufficient granularity to represent task-level reasoning accurately.

Table 4 shows combination $(M, N) = (10, 40)$ suggest some redundancy with skill pairs with similarity ≥ 0.78. **Takeaway:** Using $M = 10, N = 40$ may introduce redundant skills; other skill sets are preferable to maintain distinctiveness.

Based on these observations, we select $N = 40$ **skills and** $M = 40$ **representative clusters**.

Table 4: Intra-skill semantic overlap statistics across different skill granularities.

m	n	ToT Arithmetic		GSM8K		Math-Hard	
		Mean Sim.	$\% \geq 0.78$	Mean Sim.	$\% \geq 0.78$	Mean Sim.	$\% \geq 0.78$
5	5	0.370	0.0	0.322	0.0	0.131	0.0
5	10	0.345	0.0	0.268	0.0	0.174	0.0
5	20	0.306	0.0	0.302	0.0	0.151	0.0
5	40	0.276	0.0	0.282	0.0	0.153	0.0
10	5	0.409	0.0	0.346	0.0	0.203	0.0
10	10	0.326	0.0	0.305	0.0	0.194	0.0
10	20	0.278	0.0	0.260	0.0	0.205	0.0
10	40	0.289	0.0	0.280	13.5	0.192	0.0
20	5	0.349	0.0	0.426	0.0	0.272	0.0
20	10	0.328	0.0	0.303	0.0	0.219	0.0
20	20	0.283	0.0	0.265	0.0	0.198	0.0
20	40	0.302	38.5	0.300	0.0	0.172	0.0
40	5	0.257	0.0	0.302	0.0	0.332	0.0
40	10	0.337	0.0	0.262	0.0	0.219	0.0
40	20	0.301	0.0	0.308	0.0	0.170	0.0
40	40	0.301	12.8	0.293	0.0	0.169	0.0
80	5	0.311	0.0	0.351	0.0	0.227	0.0
80	10	0.358	0.0	0.328	0.0	0.243	0.0
80	20	0.291	0.0	0.261	0.0	0.173	0.0
80	40	0.328	0.0	0.235	0.0	0.151	0.0

C.3 Math-Hard

In Figure 5, we observe that this metric exhibits substantial variability. **Takeaway:**The configurations $(M, N) = (40, 5)$, $(20, 5)$, $(80, 5)$, $(40, 10)$, $(40, 20)$, $(5, 40)$, $(80, 40)$ achieve the lowest rate of assignments to the "Other" category. Based on Figure 8, the skill granularity $N \geq 20$ provides the sufficient granularity. Table 4 confirms that the derived skills are not redundant and distinctive. Based on these observations, we select $M = 40$ representative clusters and $N = 20$ skills.

D Skill List

D.1 20 Skills in ToT Arithmetic

1. Natural-language date/time parsing: Extract dates, times, and durations from varied textual expressions and formats.

2. Unit conversion for time spans: Translate hours, minutes, seconds, days, weeks, months, and years into a common unit (e. g. , total seconds) and back.

3. Arithmetic on durations: Add, subtract, multiply, or divide time intervals, including scaling rates to larger quantities.

4. Calendar arithmetic (date addition/subtraction): Compute new dates by adding or removing days, weeks, months, or years, respecting month lengths and leap years.

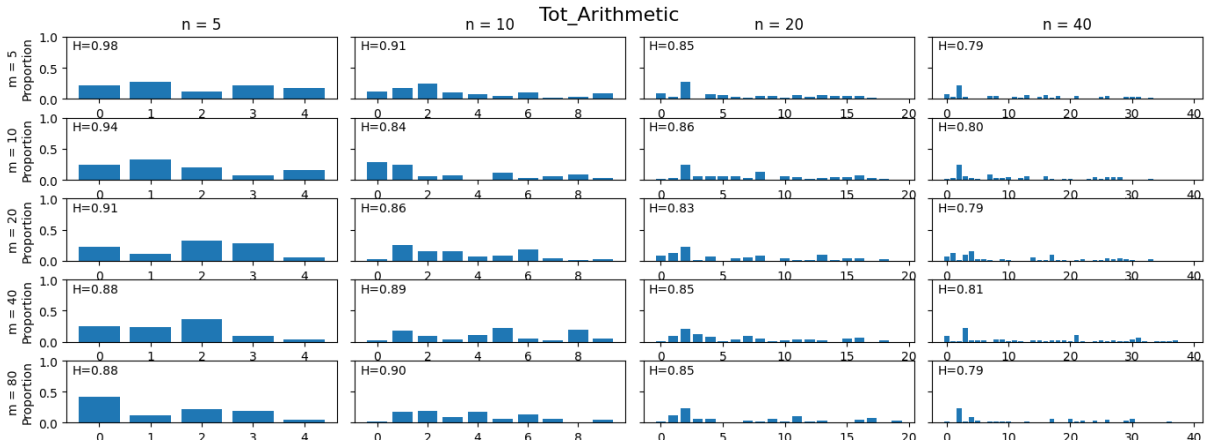
5. Leap-year and month-length handling: Determine the correct number of days in February and other months when performing date calculations.

6. Time-zone conversion: Apply UTC offsets to convert times between zones and adjust the resulting day if needed.

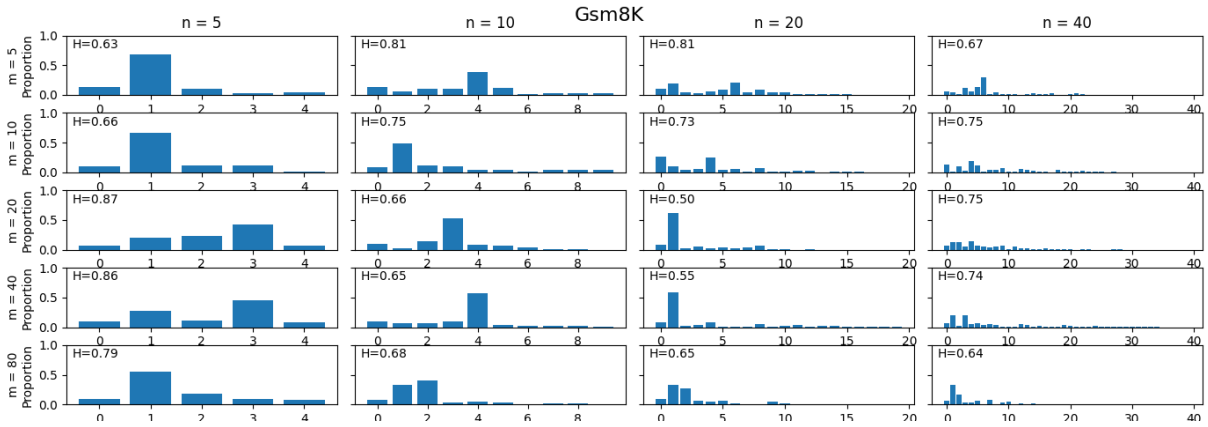
7. Day-of-week determination: Find the weekday for a given date or after a specified offset using modular arithmetic on a 7-day cycle.

8. Relative-date reasoning: Interpret statements like Ẍ days before/after Yör Ẍ weeks earlierẗo locate the target date.

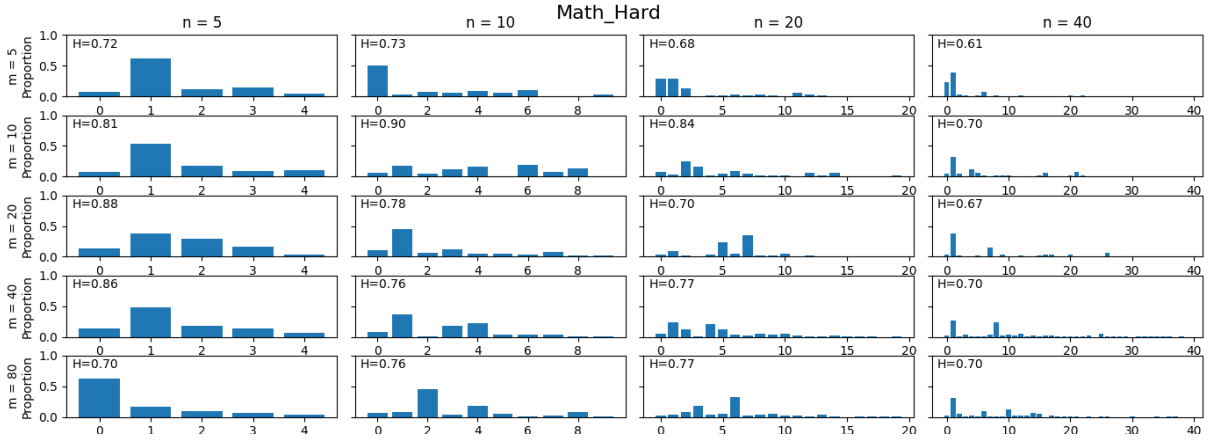
9. Normalization of incomplete timestamps: Fill missing components (e. g. , assume 00: 00: 00 for absent time) to create comparable datetime values.



(a) ToT Arithmetic

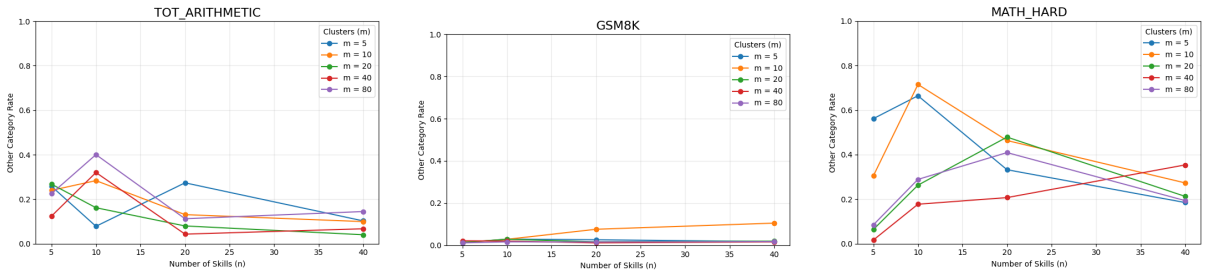


(b) GSM8K



(c) Math-Hard

Figure 4: Skill Distribution Across (m, n) Settings. X-axis skill ID each distinct skill and y axis represent the ratio of the usage. H represents entropy.



(a) ToT Arithmetic (b) GSM8K (c) Math-Hard

Figure 5: Skill distribution across datasets. Each panel shows the normalized usage of skills; H denotes normalized entropy.

10. Chronological ordering of events: Sort a set of dates/times ascending or descending to identify earliest or latest occurrences.
11. Extremum identification: Locate the maximum or minimum date/time among a collection (e. g. , latest exam, earliest activity).
12. Overlap and intersection of time intervals: Determine common free periods among multiple schedules and compute their length.
13. Discrete slot counting under constraints: Enumerate possible meeting or event slots that satisfy granularity rules (e. g. , start on the hour or half-hour).
14. Midnight and date-boundary wrap-around: Correctly handle intervals that cross midnight or span month/year boundaries.
15. Multi-format date conversion: Translate between representations such as m̈m/dd/yyyy, ÿyyy-mm-dd, d̈d-Mon-yyyy, etc.
16. Era (BC/AD) linearization: Convert BC and AD years to a single numeric timeline for comparison.
17. Parsing and applying weekly cycles: Use recurring periods (e. g. , every 19 days) to compute next occurrence from a reference date.
18. Rate-based scaling: Derive per-unit time from a given total and apply it to a different quantity (e. g. , time per box).
19. Structured JSON output generation: Assemble explanations and computed values into the required JSON schema with correct field names.
20. Ambiguity resolution in temporal language: Disambiguate phrases like b̈efore noon, s̈ame day, or ëarliest possible timeẗo select the appropriate interpretation.

D.2 40 Skills in GSM8K

1. Extracting quantitative data from narrative text: Identify all numbers, entities, and relationships described in a word problem.
2. Converting percentages to fractions/decimals: Translate %̈statements into usable numeric forms for calculation.
3. Performing operations with fractions: Add, subtract, multiply, and divide fractional quantities accurately.
4. Calculating a percentage of a quantity: Apply a percent (or its fraction/decimal equivalent) to find part-of-a-whole values.

Skill Granularity Comparison (ToT)

Original Problem
The availability of Brandon is 11 to 12:30 and also from 2:30 to 4:30 and the availability of Levi is from 9 to 11:30 and also from 4:30 to 5. How many possibilities are there for the meeting time if the meeting has to start on the hour or half hour and must be 30 minutes long. Format your answer as a JSON. Eg. JSON = ëxplanation¨ <your step by step solution>, änswer¨:<number>. Do not include units in the JSON value.

Sub-Tasks
{"sub-task": "Find the overlapping time slots between Brandon and Levi"}
{"sub-task": "Determine the possible 30-minute meeting slots within the overlapping time slots that start on the hour or half hour"}
{"sub-task": "Count the total number of possible 30-minute meeting slots"}
{"sub-task": "Format the answer as a JSON"}

Mapped Skills (m=40, n=5)
{"skill": "Duration arithmetic"}
{"skill": "Duration arithmetic"}
{"skill": "Duration arithmetic"}
{"skill": "Structured JSON reporting"}

Mapped Skills (m=40, n=10)
{"skill": "Constraint-driven scheduling slot enumeration"}
{"skill": "Constraint-driven scheduling slot enumeration"}
{"skill": "Constraint-driven scheduling slot enumeration"}
{"skill": "Constraint-driven scheduling slot enumeration"}

Mapped Skills (m=40, n=20)
{"skill": "Overlap and intersection of time intervals"}
{"skill": "Discrete slot counting under constraints"}
{"skill": "Discrete slot counting under constraints"}
{"skill": "Structured JSON output generation"}

Mapped Skills (m=40, n=40)
{"skill": "Finding overlapping free intervals among multiple schedules"}
{"skill": "Counting feasible meeting slots with granularity constraints"}
{"skill": "Counting feasible meeting slots with granularity constraints"}
{"skill": "Formatting results into a prescribed JSON structure"}

Figure 6: ToT Skill Granularity m=40 and where n=5, 10, 20, 40.

5. Proportional reasoning (e. g. , ẗwice as many, ḧalf as much¨): Set up multiplicative relationships based on comparative language.

6. Formulating and solving simple linear equations (one unknown): Turn a verbal condition into an equation and isolate the variable.

7. Formulating and solving basic inequalities: Use ät least, n̈o more than, etc. , to create and solve inequality constraints.

8. Using unit rates to compute totals: Multiply a per-unit value (price, speed, etc.) by a quantity to obtain a total amount.

9. Multi-digit multiplication and division: Carry out accurate arithmetic with two- or three-digit numbers.

10. Sequential quantity tracking: Update a running total through successive additions and subtractions.

11. Converting between time units (minutes hours): Change minutes to hours (or vice-versa) for combined-time calculations.

Skill Granularity Comparison (GSM8K)

Original Problem
Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

Sub-Tasks
{"sub-task": "Determine the total number of eggs produced each day"}
{"sub-task": "Subtract the three eggs Janet eats for breakfast from the total"}
{"sub-task": "Subtract the four eggs used for muffins from the remaining eggs"}
{"sub-task": "Calculate the number of eggs left after breakfast and muffins, which are sold"}
{"sub-task": "Multiply the number of eggs sold by $2 to find the daily earnings at the farmers' market"}

Mapped Skills (m=40, n=5)
{"skill": "Quantitative language comprehension"}
{"skill": "Sequential quantity tracking"}
{"skill": "Sequential quantity tracking"}
{"skill": "Sequential quantity tracking"}
{"skill": "Unit conversion and rate reasoning"}

Mapped Skills (m=40, n=10)
{"skill": "Narrative-to-Math Translation"}
{"skill": "Sequential Quantity Tracking"}
{"skill": "Sequential Quantity Tracking"}
{"skill": "Sequential Quantity Tracking"}
{"skill": "Rate and Proportional Reasoning"}

Mapped Skills (m=40, n=20)
{"skill": "Word-problem interpretation"}
{"skill": "Basic arithmetic operations"}
{"skill": "Basic arithmetic operations"}
{"skill": "Sequential quantity tracking"}
{"skill": "Monetary cost and change calculation"}

Mapped Skills (m=40, n=40)
{"skill": "Extracting quantitative data from narrative text"}
{"skill": "Sequential quantity tracking"}
{"skill": "Sequential quantity tracking"}
{"skill": "Determining remaining quantity after consumption and distribution"}
{"skill": "Using unit rates to compute totals"}

Figure 7: GSM8K Skill Granularity m=40 and where n=5, 10, 20, 40.

12. Area calculation for rectangles and aggregation: Compute length × width and sum multiple areas to find a total.

13. Total cost from per-item price and quantity: Multiply unit price by number of items and sum across categories.

14. Finding change or remaining amount after purchases: Subtract total expense from given money to obtain leftover funds.

15. Rounding up when dividing into packs (ceil operation): Determine the smallest whole number of packs needed to cover a quantity.

16. Applying average rates to determine total output or time: Use Ẍ per houröver a period to compute overall amount.

17. Profit calculation (revenue - cost): Subtract total expenses from total sales to obtain net gain.

Skill Granularity Comparison (Math-Hard)

Original Problem
What is the smallest real number x in the domain of the function $g(x) = sqrt(x-3)^2 - (x-8)^2$?

Sub-Tasks
{"sub-task": "Set the radicand $(x-3)^2 - (x-8)^2$ greater than or equal to zero"}
{"sub-task": "Apply the difference of squares formula to rewrite the radicand as $[(x-3)-(x-8)][(x-3)+(x-8)]$"}
{"sub-task": "Simplify each factor to obtain a linear expression multiplied by a constant"}
{"sub-task": "Solve the resulting linear inequality for x and identify the smallest real x that satisfies it"}

Mapped Skills (m=80, n=5)
{"skill": "Algebraic manipulation of equations and unknown coefficients"}
{"skill": "Algebraic manipulation of equations and unknown coefficients"}
{"skill": "Algebraic manipulation of equations and unknown coefficients"}
{"skill": "Algebraic manipulation of equations and unknown coefficients"}

Mapped Skills (m=80, n=10)
{"skill": "Translating problem conditions into algebraic equations"}
{"skill": "Factoring polynomials and using the factor theorem"}
{"skill": "Factoring polynomials and using the factor theorem"}
{"skill": "Solving systems of linear equations for unknown parameters"}

Mapped Skills (m=80, n=20)
{"skill": "Domain analysis for radicals and rational functions"}
{"skill": "Factoring and completing the square"}
{"skill": "Factoring and completing the square"}
{"skill": "Domain analysis for radicals and rational functions"}

Mapped Skills (m=80, n=40)
{"skill": "Determining domains of radical and rational functions"}
{"skill": "Factoring differences of squares and other algebraic identities"}
{"skill": "Factoring differences of squares and other algebraic identities"}
{"skill": "Determining domains of radical and rational functions"}

Figure 8: Math-Hard Skill Granularity m=80 and where n=5, 10, 20, 40.

18. Applying discount percentages to original prices: Reduce a price by a given percent and compute the discounted amount.

19. Computing savings from price differences over time: Multiply per-unit savings by quantity and by number of periods.

20. Inventory tracking over multiple days: Account for daily usage, additions, and removals to find initial or final stock.

21. Interpreting m̈ore than/ Ïess thanr̈elationships: Translate comparative statements into addition or subtraction equations.

22. Translating p̈ers̈tatements into multiplication: Convert expressions like $ per hourïnto a product of rate and time.

23. Executing mixed-operation word problems with correct order of operations: Apply PEMDAS when several operations appear together.

24. Substitution to solve for unknowns in expressions: Replace a variable with its known value to evaluate an expression.

25. Consistent handling of mixed measurement units: Keep units (gallons, inches, miles, etc.) uniform throughout calculations.

26. Calculating totals from grouped items (e. g. , students, packs): Multiply group size by members per group and sum across groups.

27. Determining remaining quantity after consumption and distribution: Subtract used and given-away amounts from an initial total.

28. Equal sharing (division) of a total among participants: Divide a quantity evenly to find each person's share.

29. Using ät leastc̈onditions to find minimum required values: Set up an inequality and solve for the smallest feasible number.

30. Converting dozens to individual units: Multiply a dozen count by 12 to obtain the exact number of items.

31. Summing contributions from multiple sources to a total: Add separate amounts (e. g. , earnings, donations) to obtain a combined figure.

32. Multiplicative scaling (ẗwice, ẗhree times) for unknown quantities: Represent n̈ timesr̈elationships as multiplication in equations.

33. Distance calculation from speed and time segments: Multiply speed by each time interval and sum the distances.

34. Total weight determination by adding individual item weights: Sum the weights of all objects carried or listed.

35. Using difference statements to isolate unknown quantities: Rearrange ẗotal - known = unknownr̈elationships to solve.

36. Interpreting ẗotal ofs̈tatements to set up equations: Translate ẗhe total is Xïnto an equation linking component parts.

37. Applying p̈er packp̈ricing to compute overall cost: Multiply number of packs by price per pack, accounting for partial packs if needed.

38. Calculating weekly or monthly earnings from hourly wages: Multiply hourly rate by total hours worked in the period.

39. Solving mixture problems (e. g. , temperature blending) with weighted averages: Use weighted-average formulas to find unknown component values.

40. Budget allocation across multiple items under constraints: Distribute a fixed amount among purchases while respecting given limits."

D.3 20 Skills in Math-Hard

1. Translating word problems into algebraic statements: Extract quantities, relationships, and constraints from prose and express them as equations or inequalities.

2. Multi-step arithmetic with integers and fractions: Perform sequences of additions, subtractions, multiplications, and divisions accurately, including reduction of fractions.

3. Combinatorial counting (combinations & permutations): Use binomial coefficients and factorial reasoning to enumerate selections, arrangements, and distributions.

4. Inclusion-exclusion reasoning: Account for overlapping cases by adding and subtracting intersecting counts to obtain correct totals.

5. Probability via counting: Compute probabilities by determining the number of favorable outcomes divided by the total number of equally likely outcomes.

6. Solving linear equations and systems: Isolate variables, substitute, and use elimination or matrix methods to find unknown values.

7. Quadratic equation techniques: Factor, complete the square, or apply the quadratic formula to find real or complex roots.

8. Converting repeating decimals to fractions: Set up algebraic equations for repeating blocks, solve for the unknown, and simplify to lowest terms.

9. Arithmetic and geometric series analysis: Identify first term and common difference/ratio, use sum formulas, and test convergence for infinite series.

10. Modular arithmetic and congruence solving: Work with residues, solve linear congruences, and apply the Chinese remainder theorem when needed.

11. GCD/LCM determination via prime factorization: Decompose integers into primes to compute greatest common divisor and least common multiple efficiently.

12. Absolute-value inequality manipulation: Split into casewise linear inequalities, solve each case, and intersect solution sets.

13. Domain and range of functions (radicals, rationals, absolute values): Impose non-negativity, denominator non-zero, and piecewise analysis to describe permissible inputs and output intervals.

14. Completing the square for optimization: Rewrite quadratic expressions as a perfect square plus constant to locate minima or maxima.

15. Calculus-based optimization: Differentiate area, volume, or other expressions, set derivatives to zero, and verify extremal values.

16. Vector dot and cross product orthogonality: Compute cross products, take dot products, and set results to zero to enforce perpendicularity.

17. Rayleigh quotient / eigenvalue maximization: Recognize quadratic forms, relate them to eigenvalues, and use the largest eigenvalue to obtain maximal ratios.

18. Solving higher-degree polynomials (cubic methods): Apply substitutions, depressed-cubic forms, and Cardano's formula to obtain exact roots.

19. Lattice-point counting under distance constraints: Use integer solutions of circle equations ($x^2 + y^2 = r^2$) to enumerate points satisfying a given distance.

20. Symmetry counting with Burnside's Lemma: Identify group actions (rotations, reflections), count fixed configurations under each, and average to obtain distinct arrangements.

E Prompts

E.1 Generating Test Cases

F Results

F.1 Skill-specific difficulty via marginal contribution

To quantify the contribution of individual skills, we compute marginal contributions based on the minimum scaffolding threshold k. A skill is assigned a marginal contribution if its addition changes from failure to success. Aggregating these contributions across tasks, Table 8 shows which skills are most frequently bottleneck-enabling (that lets the model succeed) in multi-step reasoning.

Sub-Task Decomposition

You will be given a question that requires multiple reasoning or computational steps. Your task is to break down the instruction into explicit, step-by-step sequential segments that detail the actions needed to answer the question. Each segment should represent a distinct actionable operation.
Rules: - Each segment must represent a concrete reasoning or computational action. - Each segment must yield a concrete intermediate result. - Each segment must be non-overlapping and cover all parts of the instruction. - Each segment must represent a single unit of instruction. If a step contains multiple actions (e.g., "add and divide"), split it into separate segments. - Each segment must directly involve computation or logical derivation. - Each segment only describes the action not the solution. - The last segment must represent the final step such that the response to this segment would be the final answer. - Do not include numeric or textual answers inside the segments. Only describe the action needed to obtain them. - Limit the total number of segments to 6 or fewer.
Your output must be a list of these segments in the following JSON format: [{"segment": "[first step]"}, {"segment": "[second step]"}, ...]
Here are some examples:
Example Query: Josh decides to try flipping a house. He buys a house for $80k and then puts in $50k in repairs. This increased the value of the house by 150%. How much profit did he make?
Example Output: [{"segment": "Add the purchase price and repair costs to find total spending"}, {"segment": "Calculate the increase in house value using 150%"}, {"segment": "Find the new value of the house after the increase"}, {"segment": "Subtract total spending from the new value to get the profit"}]
Example Query: It takes Sarah an average of 15 minutes and 36 seconds to solve 2 puzzles. If she wants to solve 12 puzzles at the same rate, it will take her X hours, Y minutes, and Z seconds.
Example Output: [{"segment": "find the time it takes to solve one puzzle"}, {"segment": "Multiply the time for one puzzle by 12 to get the total time for 12 puzzles"}, {"segment": "Convert the total seconds into hours, minutes, and seconds"}]
Now complete the task for the following question: {{ question }}

Sub-Task Answer

You will be given a question and step-by-step sequential segments that detail the reasoning or actions needed to answer the question. Your task is to solve each segment in order. For each segment, provide a clear step-by-step reasoning and the final result for that segment. Your output must be a list of answers where each corresponds to each segment in the following JSON format:
[{"explanation": "[detailed reasoning for the first segment]", "answer": "[final answer to the first segment]"}, {"explanation": "[detailed reasoning for the second segment]", "answer": "[final answer to the second segment]"}, ...]
Example: Josh decides to try flipping a house. He buys a house for 80k and then puts in 50k in repairs. This increased the value of the house by 150
[{"segment": "Total amount Josh spent"}, {"segment": "Increase in house value"}, {"segment": "New value of the house"}, {"segment": "Profit"}, {"segment": "Format the final result as a JSON"}]
Output: [{"explanation": "Josh spent 80k to buy the house and 50k on repairs. Adding them gives 80k + 50k = 130k.", "answer": "130k"}, {"explanation": "The original value was 80k. A 150{"explanation": "The new value is original value + increase = 80k + 120k = 200k.", "answer": "200k"}, {"explanation": "Profit = New value - Total spent = 200k - 130k = 70k.", "answer": "70k"}, {"explanation": "Format the final result as a JSON.", "answer": "{ëxplanation̈: J̈osh spent 80k to buy the house and 50k on repairs, totaling 130k. The house increased by 150]
Now complete the task for the following question: question sequential$_segments$
Only output the JSON array, strictly following the format.

Consistency

You are given a ground truth answer and a model answer. Your task is to decide whether the two answers are equivalent in value.
Rating Guidelines 1. Ignore formatting differences (e.g., 2, "2", 2.0, answer: 2 should all be treated as the same). 2. Treat numbers written as words (e.g., two, forty-five) as equivalent to their numeric forms. 3. Units must be considered: 2 kg is not equal to 2 g, but 2000 g is equal to 2 kg. 4. Time expressions should be normalized: treat equivalent times as the same value even if expressed differently (e.g., "7:00", "7 am", "07:00", or "7 o'clock" all mean the same; "1 pm–3 pm" = "13:00–15:00"). Overlapping time intervals must match in value, regardless of format. 5. If either the model output or ground-truth answer is empty, missing, or unspecified, return a score of 0 regardless of other conditions.
Model Output {{validator_final_answer}}
Ground-truth Answer {{ground_truth}}
Scoring Criteria * Score 1: If the answers are equivalent in value. * Score 0: If the answers are different in value.
Return your rating as a json of the form {"score": your_score, "justification": your_justification}. No additional explanation, text, or formatting outside the JSON.

Scaffolded Variation

You will be given a question along with a partially completed step-by-step solution. Your task is to rewrite the original question by incorporating the parts of the solution that have already been completed, so the rewritten question reflects that those steps are done. The new question should only require solving for the remaining steps. When rewriting, preserve the structure and wording of the original question as much as possible—only revise or replace the parts that are directly affected by the completed steps. Use the completed work to inform the new phrasing, as if someone is picking up the problem mid-way with that progress already understood. The rewritten question must be a single line without any line breaks.
Example Original Question: Josh decides to try flipping a house. He buys a house for $80k and then puts in $50k in repairs. This increased the value of the house by 150%. How much profit did he make?
Solved Segments: {"segment": "Calculate total amount Josh spent", "answer": "130K"}, {"segment": "Calculate increase in house value", "answer": "120K"},
Rewritten Question: Josh decides to try flipping a house. He spent a total of $130k for the house and the repairs. This increased the value of the house by $120K. When the original value of the house was $80k, how much profit did he make?
Now rewrite the following question using the completed steps provided. Reply in the following format: Rewritten Question: <your rewritten version here>
Original Question: {{ question }}
Solved Segments: {{ solved_sequential_segments }}

Skill Generation

You will be given a set of representative questions on {{category}}.
Your goal is to identify {{num_of_skills}} distinct skills that a learner (or model) must possess to correctly solve problems in this category. These skills will be used to map fine-grained sub-tasks to higher-level capabilities in order to analyze model performance.
Each skill should: - Represent a specific cognitive or procedural competency, not a topic name (e.g., "Simplifying algebraic fractions" instead of "Rational expressions"). - Cover a range of difficulty, from foundational to advanced subskills. - Be granular enough that multiple skills may be needed to solve a single question. - Avoid redundancy (each skill should describe a unique capability).
Example: Representative Questions (abridged): 1. Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market? Steps to solve: "Determine the total number of eggs produced each day", "Subtract the three eggs Janet eats for breakfast from the total", "Subtract the four eggs used for muffins from the remaining eggs", "Calculate the number of eggs left after breakfast and muffins, which are sold", "Multiply the number of eggs sold by $2 to find the daily earnings at the farmers' market." 2. Henry and 3 of his friends order 7 pizzas for lunch. Each pizza is cut into 8 slices. If Henry and his friends want to share the pizzas equally, how many slices can each of them have? Steps to solve: "Multiply the number of pizzas (7) by the number of slices per pizza (8) to find the total number of slices", "Count the total number of people sharing the pizza (Henry plus 3 friends) to determine the number of recipients", "Divide the total number of slices by the number of people to find how many slices each person receives." 3. Jame will turn 27 in 5 years. In 8 years his cousin will be 5 years younger than twice his age. How many years separate the age of the two now? Steps to solve: "Subtract 5 from 27 to obtain Jame's current age", "Add 8 to Jame's current age to find Jame's age in 8 years", "Formulate cousin's age in 8 years as (twice Jame's age in 8 years) minus 5", "Calculate cousin's age in 8 years using the value from step 2", "Subtract 8 from cousin's age in 8 years to get cousin's current age", "Compute the absolute difference between cousin's current age and Jame's current age."
Example Skills (abridged): 1. Quantitative interpretation of word problems: extracting relevant numbers, entities, and constraints from text. 2. Basic arithmetic operations: addition, subtraction, multiplication, and division in multi-step contexts. 3. Sequential quantity tracking: updating remaining or accumulated values across successive steps. 4. Equal sharing and unit-rate calculations: dividing quantities evenly and applying per-unit values. 5. Temporal and relational reasoning: shifting values across time and forming linear relationships from verbal descriptions.
Now perform this task: Using the representative questions below, infer and list {{num_of_skills}} well-defined skills required to solve the full set of problems.
Focus on the underlying reasoning and computation abilities shared across questions, rather than restating the solution steps or question content.
Representative Questions: {{question_clusters}}
Output Format: - Return a numbered list of {{num_of_skills}} skills, each described in one concise sentence. - Do not include solutions, examples, or references to specific questions. - Each item should be in the format: 1. Skill name: brief explanation.

Skill Mapping

You are a classification model. You will be given a problem and its associated sub-tasks. For each sub-task, classify it into exactly one of the predefined skills listed below.
Skills (skill ID: description): {{skills}}
Output requirements: - Return a JSON array with one object per sub-task. - Each object must contain only the selected skill ID from the list. - Do not include any extra text, explanation, or formatting.
Example format: [{"skill": skill id from the list}, {"skill": skill id from the list}, {"skill": skill id from the list}, ...]
Problem: {{ question }}
Sub-Tasks: {{ sub-tasks }}

Skill (ToT Arithmetic)	**Qwen3B**	**Qwen7B**	**Llama3B**	**Llama8B**	**granite2B**	**granite8B**
Structured JSON output	0.55	0.70	0.38	0.42	0.53	0.66
Arithmetic on durations	0.54	0.71	0.42	0.50	0.55	0.69
Unit conversion for time spans	0.51	0.70	0.33	0.42	0.37	0.60
Calendar arithmetic	0.18	0.22	0.19	**0.18**	0.16	0.23
Natural-language date/time parsing	0.35	0.27	0.23	0.26	0.28	0.27
Discrete slot counting	0.19	0.29	0.15	0.21	0.16	0.25
Overlap and intersection of time	**0.11**	**0.18**	**0.13**	0.20	**0.11**	**0.22**
Multi-format date conversion	0.62	0.65	0.42	0.52	0.55	0.64
Day-of-week determination	0.44	0.37	0.27	0.36	0.28	0.43
Chronological ordering of events	**0.02**	**0.04**	**0.08**	**0.14**	**0.08**	**0.10**
Extremum identification	0.26	0.19	0.30	0.32	0.26	0.31
Normalization of incomplete timestamps	0.39	0.47	0.63	0.60	0.37	0.49
Midnight and date-boundary wrap-around	0.28	0.35	0.19	0.28	0.30	**0.22**
Time-zone conversion	0.40	0.62	0.55	0.66	0.36	0.57
Relative-date reasoning	0.29	0.44	0.20	0.31	0.12	0.24
Average (std)	0.34 (0.17)	0.41 (0.22)	0.30 (0.16)	0.36 (0.16)	0.30 (0.16)	0.39 (0.20)

Skill (GSM8K)	**Qwen3B**	**Qwen7B**	**Llama3B**	**Llama8B**	**granite2B**	**granite8B**
Sequential quantity tracking	0.70	0.77	0.74	0.70	0.66	0.77
Using unit rates to compute totals	0.82	0.82	0.86	0.87	0.75	0.83
Extracting quantitative data from text	**0.25**	**0.19**	**0.29**	**0.20**	**0.11**	**0.19**
Summing contributions from multi sources	0.67	0.78	0.77	0.80	0.69	0.89
Formulating and solving linear equations	**0.49**	**0.50**	**0.42**	**0.48**	0.51	**0.42**
Multi-digit multiplication and division	0.90	0.85	0.84	0.94	0.73	0.83
Proportional reasoning	0.77	0.76	0.77	0.85	0.57	0.79
Calculating a percentage of a quantity	0.85	0.89	0.93	0.96	0.67	0.93
Performing operations with fractions	0.70	0.80	0.73	0.69	0.60	0.58
Multiplicative scaling for unknowns	0.81	0.86	0.76	0.73	0.74	0.75
Remaining amount after purchases	0.76	0.89	0.86	0.84	0.83	0.82
Total cost from price and quantity	0.88	0.91	0.79	0.80	0.67	1.00
Isolate unknown quantities	0.86	1.00	0.76	0.73	0.68	0.71
Equal sharing of a total	0.89	1.00	0.74	0.79	0.59	0.83
Determining remaining quantity	0.82	1.00	0.90	0.80	**0.50**	1.00
Average (std)	0.74 (0.17)	0.80 (0.21)	0.74 (0.17)	0.75 (0.19)	0.62 (0.17)	0.76 (0.22)

Skill (Math-Hard)	**Qwen3B**	**Qwen7B**	**Llama3B**	**Llama8B**	**granite2B**	**granite8B**
Multi-step integer/fraction arithmetic	0.45	0.51	0.55	0.60	0.56	0.63
Translating problems into algebra	0.24	0.23	0.31	**0.31**	**0.34**	0.34
Solving linear equations and systems	0.29	0.31	0.41	0.45	0.46	0.47
Combinatorial counting	0.36	0.46	0.44	0.42	0.43	0.47
Quadratic equation techniques	0.30	0.40	0.44	0.51	0.50	0.42
Modular arithmetic and congruence	0.34	0.44	0.40	0.35	0.39	0.54
GCD/LCM via prime factorization	0.38	0.41	0.49	0.51	0.49	0.53
Probability via counting	0.28	0.46	0.39	0.39	0.44	0.49
Domain and range of functions	0.20	0.20	0.55	0.49	0.43	0.50
Arithmetic and geometric series analysis	0.36	0.28	0.43	0.41	0.52	0.35
Completing the square for optimization	0.20	**0.12**	0.68	0.46	0.46	0.62
Calculus-based optimization	0.23	0.35	0.39	**0.27**	0.35	0.36
Absolute-value inequality manipulation	**0.17**	0.30	0.35	0.38	0.45	**0.17**
Symmetry counting w/ Burnside's Lemma	0.28	**0.19**	**0.22**	0.35	**0.18**	0.33
Solving higher-degree polynomials	**0.09**	0.24	**0.30**	0.35	0.39	**0.30**
Average (std)	0.28 (0.09)	0.33 (0.12)	0.42 (0.11)	0.42 (0.09)	0.43 (0.09)	0.43 (0.13)

Table 5: Individual skill accuracy across models and datasets (Top-15 most frequent skills). The two lowest-performing skills are highlighted in bold.

Table 6: Skill combinations with higher mean minimum scaffolding (k) require more explicit guidance and are therefore more difficult for the model. $k = 0$ indicates tasks solved independently; % $k > 0$ are % of tasks that can be solved with scaffolding; % *Intractable* indicates tasks that cannot be solved even with full scaffolding.

Combination (ToT)	#Tasks	Model	Mean k	% $k = 0$	% $k > 0$	% Intractable
Overlapping + Overlapping + Meeting slots + Meeting slots + JSON	65	Qwen3B	3.02	6.15	70.76	23.07
		Qwen7B	2.51	6.15	89.23	4.61
		Llama3B	3.06	3.07	75.38	21.53
		Llama8B	2.69	10.76	81.53	7.69
		Granite2B	2.93	1.53	73.84	24.61
		Granite8B	2.79	10.76	81.53	7.69
Unit + Duration + Duration + Unit + JSON	61	Qwen3B	1.90	45.90	52.45	1.63
		Qwen7B	1.45	67.21	32.78	0.0
		Llama3B	3.68	9.83	31.14	59.01
		Llama8B	3.44	13.11	59.01	27.86
		Granite2B	3.19	9.83	59.01	31.14
		Granite8B	1.97	21.31	78.68	0.0
Unit + Date-Boundary + Duration + Unit + JSON	41	Qwen3B	2.32	4.87	90.24	4.87
		Qwen7B	2.18	34.14	65.85	0.00
		Llama3B	3.85	0.0	68.29	31.70
		Llama8B	3.60	0.0	80.48	19.51
		Granite2B	3.82	2.43	68.29	29.26
		Granite8B	2.48	9.75	85.36	4.87

Combination (GMS8k)	#Tasks	Model	Mean k	% $k = 0$	% $k > 0$	% Intractable
Unit-Total + Unit-Total + Unit-Total + Sum	15	Qwen3B	0.00	93.33	0.00	6.66
		Qwen7B	0.00	100.00	0.00	0.00
		Llama3B	1.33	80.00	20.00	0.00
		Llama8B	1.00	93.33	6.66	0.00
		Granite2B	1.00	93.33	6.66	0.00
		Granite8B	2.00	93.33	6.66	0.00
Quantity-Tracking + Quantity-Tracking	14	Qwen3B	1.00	92.85	7.14	0.00
		Qwen7B	0.00	100.00	0.00	0.00
		Llama3B	1.00	85.71	14.28	0.00
		Llama8B	0.00	100.00	0.00	0.00
		Granite2B	1.00	92.85	7.14	0.00
		Granite8B	1.00	92.85	7.14	0.00
Extracting + Proportional + Quantity-Tracking	10	Qwen3B	0.00	100.00	0.00	0.00
		Qwen7B	0.00	100.00	0.00	0.00
		Llama3B	0.00	100.00	0.00	0.00
		Llama8B	0.00	100.00	0.00	0.00
		Granite2B	0.00	100.00	0.00	0.00
		Granite8B	1.00	90.00	10.00	0.00

Combination (Math-Hard)	#Tasks	Model	Mean k	% $k = 0$	% $k > 0$	% Intractable
Arithmetic + Arithmetic + Arithmetic + Arithmetic + Arithmetic	12	Qwen3B	2.14	33.33	58.33	8.33
		Qwen7B	2.5	75.0	16.66	8.33
		Llama3B	1.71	33.33	58.33	8.33
		Llama8B	1.00	41.66	41.66	16.66
		Granite2B	2.50	41.66	50.00	8.33
		Granite8B	2.00	75.00	25.00	0.00
Combinatorics + Combinatorics + Combinatorics + Combinatorics + Combinatorics	11	Qwen3B	1.8	36.36	45.45	18.18
		Qwen7B	2.42	27.27	63.63	9.09
		Llama3B	2.62	18.18	72.72	9.09
		Llama8B	2.16	18.18	54.54	27.27
		Granite2B	1.88	9.09	81.81	9.09
		Granite8B	1.75	63.63	36.36	0.00
Congruences + Congruences + Congruences + Congruences + Congruences + Congruences	9	Qwen3B	2.88	0.00	100.00	0.00
		Qwen7B	2.50	11.11	88.88	0.00
		Llama3B	3.14	22.22	77.77	0.00
		Llama8B	3.00	11.11	77.77	11.11
		Granite2B	3.00	0.00	88.88	11.11
		Granite8B	3.00	22.22	77.77	0.00

Table 7: Frequency of single-skill and compositional-skill bottlenecks. Numbers indicate count of scaffolded cases. Only the first word of each skill is shown.

	Skill (ToT)	Qwen3B	Qwen7B	Llama3B	Llama8B	Granite2B	Granite8B
Single-skill	**ToT Arithmetic**						
	Unit	**45**	**48**	10	19	16	49
	Natural	**43**	**52**	**42**	**44**	30	**59**
	Overlap	27	39	22	28	20	28
	Calendar	26	40	28	38	34	36
	Day-of-week	25	5	20	16	21	17
	Arithmetic	18	20	3	12	7	28
	Time-zone	15	6	13	11	15	15
	Relative	10	6	2	12	5	7
	Normalization	7	12	6	8	6	6
	Multi-format	4	7	5	6	5	4
	GSM8K						
	Unit	33	7	35	19	24	13
	Extracting	22	11	14	13	18	10
	Sequential	16	7	21	13	11	12
	Multi-digit	9	4	14	10	10	4
	Performing	8	-	10	3	4	3
	Proportional	8	3	6	5	6	5
	Calculating	7	2	10	7	5	3
	Total	6	3	6	2	6	-
	Formulating	4	-	8	2	3	6
	Summing	3	-	-	2	-	-
	Math-Hard						
	Translating	41	43	43	50	50	63
	Multi-step	26	17	29	20	20	18
	Combinatorial	20	28	29	25	27	24
	Modular	16	13	11	11	16	11
	Probability	9	7	6	7	11	6
	Solving	8	6	3	6	7	0
	Quadratic	4	4	3	4	3	0
	GCD/LCM	4	5	4	4	3	5
	Arithmetic	3	4	3	4	9	5
	Domain	3	6	3	6	5	4
Compositional-skill ($n = 2$)	**ToT Arithmetic**						
	Overlap / Discrete	**56**	**49**	**58**	**59**	**66**	**59**
	Natural / Calendar	39	30	34	41	**43**	50
	Calendar / Arithmetic	31	25	21	25	34	35
	Unit / Arithmetic	30	17	36	**57**	**54**	**53**
	Multi / Calendar	23	15	15	17	19	16
	Relative / Calendar	15	14	12	13	20	21
	Natural / Arithmetic	12	4	13	13	9	11
	Unit / Midnight	12	9	10	4	12	13
	Natural / Overlap	9	25	6	12	8	11
	Unit / Calendar	7	9	11	8	10	13
	GSM8K						
	Extracting / Sequential	5	2	5	2	7	3
	Unit / Summing	3	-	-	-	2	-
	Extracting / Multiplicative	3	-	3	-	-	-
	Extracting / Proportional	2	-	2	3	2	-
	Unit / Proportional	2	-	2	-	2	-
	Unit / Sequential	2	-	-	-	-	-
	Sequential / Proportional	2	2	-	-	2	2
	Math-Hard						
	Translating / Solving	15	10	11	10	13	4
	Translating / Multi-step	6	6	5	10	10	7
	Domain / Quadratic	2	2	0	2	0	0
	Arithmetic / Translating	2	2	3	3	3	3
	Combinatorial / Translating	2	2	2	4	5	2
	Modular / Multi-step	0	2	2	0	0	0
	Probability / Combinatorial	2	2	3	3	2	2
	Translating / Quadratic	0	4	4	4	5	0
	Multi-step / GCD/LCM	0	2	0	0	0	0
	Calculus / Translating	0	0	0	0	3	2
Compositional-skill ($n >= 3$)	**ToT Arithmetic**						
	Natural / Overlap / Discrete	29	29	**38**	26	42	32
	Unit / Midnight / Arithmetic	14	11	27	29	26	18
	Natural / Calendar / Multi	5	2	4	5	4	7
	Unit / Arithmetic / Midnight	4	3	5	7	3	4
	Relative / Calendar / Multi	4	2	9	3	4	3
	Natural / Arithmetic / Day-of-week	4	4	5	4	0	4
	Unit / Calendar / Multi	3	2	2	4	3	2
	Arithmetic / Unit / Midnight	3	3	3	7	3	4
	Unit / Time-zone / Arithmetic / Midnight	2	3	2	3	2	3
	Natural / Calendar / Unit / Arithmetic	2	2	2	2	2	2
	Math-Hard						
	Translating / Multi-step / Solving	0	2	2	2	3	2

Skill (ToT Arithmetic)	Qwen3B	Qwen7B	Llama3B	Llama8B	granite2B	granite8B
Structured JSON output	0	0	1	0	0	1
Arithmetic on durations	104	74	58	83	93	141
Unit conversion for time spans	76	78	88	113	89	82
Calendar arithmetic	142	139	125	148	170	183
Natural-language date/time parsing	50	61	45	50	41	68
Discrete slot counting	99	93	112	97	123	105
Overlap and intersection of time	45	68	32	44	33	44
Multi-format date conversion	31	27	34	33	24	19
Day-of-week determination	37	10	38	34	41	24
Chronological ordering of events	48	30	44	43	51	40
Extremum identification	10	2	31	35	26	17
Normalization of incomplete timestamps	39	58	22	37	21	32
Midnight and date-boundary wrap-around	32	27	22	22	24	31
Time-zone conversion	16	7	15	14	18	19
Relative-date reasoning	14	7	11	22	19	11

Table 8: Marginal Contribution show which skills are most frequently bottleneck-enabling.