# †DAGGER: Distractor-Aware Graph Generation for Executable Reasoning in Math Problems

**Zabir Al Nazi** [1], **Shubhashis Roy Dipta** [2], **Sudipta Kar** [3]
[1]University of California, Riverside
[2]University of Maryland, Baltimore County
[3]Oracle Health AI
znazi002@ucr.edu, sroydip1@umbc.edu, sudipta.k.kar@oracle.com

## Abstract

Chain-of-Thought (CoT) prompting is widely adopted for mathematical problem solving, including in low-resource languages, yet its robustness under irrelevant context - a known vulnerability in English - remains unexamined in such settings. To systematically probe this challenge, we introduce **DistractMath-BN**, a Bangla benchmark that augments MGSM and MSVAMP with semantically coherent but computationally irrelevant information. Evaluating seven models ranging from 3B to 12B parameters, we observe substantial performance degradation under distractors: standard models drop up to 41 points, while reasoning-specialized models decline 14–20 points despite consuming 5× more tokens. We propose †DAGGER, which reformulates mathematical problem solving as executable computational graph generation with dedicated modeling of distractor nodes. Fine-tuning Gemma-3 models using Supervised Fine-Tuning followed by Group Relative Policy Optimization achieves comparable aggregate accuracy across original and distractor-augmented splits while using 89% fewer tokens than reasoning models. Importantly, this robustness emerges without explicit training on distractor-augmented examples. Our results suggest that enforcing structured intermediate representations can improve robustness and inference efficiency in mathematical understanding compared to free-form reasoning, particularly in noisy, low-resource settings.[1]

## 1 Introduction

Chain-of-Thought (CoT) prompting has become a standard approach for mathematical word problem (MWP) solving in large language models (LLMs), enabling models to generate intermediate reasoning steps prior to producing final answers (Wei et al., 2022; Cheng et al., 2025). This paradigm has been widely adopted across languages and datasets, including recent work demonstrating that CoT-based prompting provides strong baselines for Bangla mathematical reasoning benchmarks (Shi et al., 2022; Ahmed et al., 2025). Reasoning-specialized models further extend this paradigm by allocating additional computation to multi-step reasoning through longer or iterative generations (Yang et al., 2025a; Luong et al., 2025) but at substantially higher token budgets.

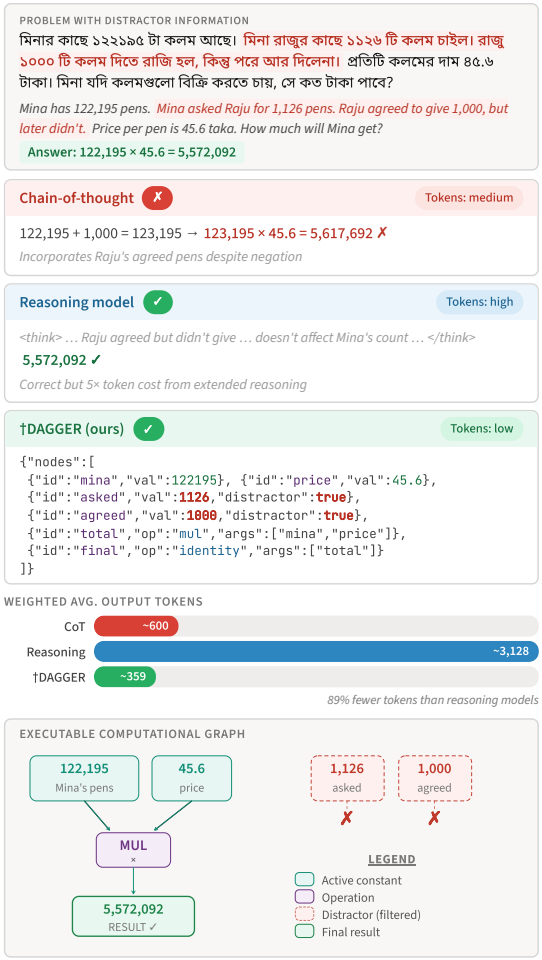



Figure 1: Comparing CoT, reasoning, and †DAGGER approaches on a distractor-augmented Bangla MWP.

Despite these advances, mathematical problem solving outside curated benchmarks frequently in-

[1]https://dipta007.github.io/DAGGER/



*Findings of the Association for Computational Linguistics: EMNLP 2026*, pages 48423–48446
October 24-29, 2026 ©2026 Association for Computational Linguistics

volves semantically coherent but computationally irrelevant information - *distractors* - such as quantities about related entities, attributes in orthogonal dimensions, or events that are negated or hypothetical (Fig. 1). Prior work has established this vulnerability in English: Shi et al. (2023) showed irrelevant sentences degrade reasoning significantly, and subsequent studies confirm this fragility under adversarial settings (Li et al., 2023; Yang et al., 2025b). However, distractor robustness in low-resource languages remains under-examined, and the *computational cost* of achieving robustness is largely overlooked. Reasoning-specialized models trained with reinforcement learning (Guo et al., 2025) show improved robustness but require 10-100× more tokens at inference (Snell et al., 2025), raising a practical question: ***can we achieve distractor robustness without the computational overhead of extended reasoning, particularly in resource-constrained settings where both accuracy and efficiency matter?***

**Research Questions** Low-resource languages like Bangla show greater vulnerability to irrelevant context during math reasoning (Sobhani et al., 2025; Prama et al., 2025). While CoT prompting improves English math reasoning (Wei et al., 2022), it exhibits brittleness to distractors (Shi et al., 2023). Reasoning models trained using Group Relative Policy Optimization (GRPO) achieve substantial gains in mathematical problem-solving (Guo et al., 2025) but require more tokens (Snell et al., 2025), raising deployment concerns.

Recent works suggest reasoning traces produced by reasoning models can be structured as ***graphs*** (Minegishi et al., 2025), potentially enabling more efficient reasoning (Yu et al., 2024). These observations motivate us to investigate the following research questions. **RQ1**: whether CoT-based reasoning remains robust on distractor-augmented Bangla MWPs; **RQ2**: whether structured reasoning frameworks can be distractor-aware ***without targeted*** distractor-type training; and **RQ3**: the computational trade-offs between robustness and token efficiency.

**Contributions** We make the following contributions in this paper.

- We introduce DistractMath-BN, a novel Bangla benchmark of 3,685 distractor-augmented problems derived from MGSM and MSVAMP validated by GPT and human expert. The benchmark systematically incorporates three classes of interference - Related Entity Distractors, Orthogonal Attribute Distractors, and Null-Effect Event Distractors *(details in Section 3, examples in Appendix A.11)*.
- Thorough evaluation of seven open-weight models ranging from 3B to 12B parameters, we show that CoT-based reasoning degrades substantially in the presence of distractors.
- Our proposed framework †DAGGER reformulates mathematical problem solving as executable computational graph generation with explicit modeling of distractor nodes.
- We show substantial accuracy and token efficiency over long-form CoT by fine-tuning Gemma-3 (SFT+GRPO) using †DAGGER framework for improved graph generation.

Overall, our findings indicate that robustness to distractor information is not guaranteed by longer reasoning traces alone. Instead, enforcing structured intermediate representations provides a practical mechanism for improving both robustness and inference efficiency in mathematical reasoning, particularly in noisy, low-resource settings.

## 2 Related Work

**Chain-of-Thought Reasoning** CoT prompting (Wei et al., 2022) enables large language models (LLMs) to generate intermediate reasoning steps, substantially improving performance on mathematical word problems. Subsequent variants such as self-consistency (Wang et al., 2023), and least-to-most prompting (Zhou et al., 2023) further enhance reasoning accuracy. In multilingual and low-resource settings, recent studies report that CoT-based prompting remains a strong baseline for Bangla mathematical reasoning benchmarks (Ahmed et al., 2025).

**Reasoning Robustness** Prior work on mathematical problem solving has shown that, while CoT prompting can improve accuracy on math word problems, the resulting step-by-step rationales are often brittle and not reliably indicative of the model's underlying computation - models may produce convincing but unfaithful explanations and still fail at exact or algorithmic calculation (Turpin et al., 2023; Sprague et al., 2024). More closely related are studies showing that irrelevant or misleading context can significantly degrade model performance in question answering

and reasoning tasks (Shi et al., 2023; Roy Dipta and Ferraro, 2025; Nazi et al., 2026).

**Structured Reasoning** Program-Aided Language Models (PAL) (Gao et al., 2023) and Program-of-Thoughts (PoT) (Chen et al., 2022) translate reasoning into executable programs, offloading computation to external interpreters such as Python. While effective for numerical accuracy, these methods generate code sequentially and treat all input tokens uniformly, offering limited mechanisms to isolate computationally irrelevant context. Related work on graph-based reasoning and planning (Yao et al., 2023; Besta et al., 2024) focuses primarily on knowledge graphs or abstract decision processes rather than executable computational structures for math. In particular, Graph-of-Thoughts (Besta et al., 2024) models reasoning as a graph over alternative *thought strategies* explored via multiple LLM calls (∼6 per problem), whereas †DAGGER generates a single executable *computational* graph in one call, with explicit distractor annotations absent from all prior approaches. Particularly, †DAGGER generates executable computational graphs with dedicated modeling of distractor nodes, enabling structural separation between relevant computation and irrelevant information prior to execution.

**Bangla Mathematical Reasoning** MGSM (Shi et al., 2022) and MSVAMP (Patel et al., 2021) provide multilingual benchmarks for mathematical reasoning, and recent Bangla work builds instruction-tuned models (Zehady et al., 2026) and a curriculum-GRPO math reasoner (Dipta et al., 2026b) evaluated on their Bangla splits. We extend these benchmarks with systematic distractor augmentation enabling controlled evaluation of reasoning robustness under semantically plausible but computationally irrelevant context in low-resource language *(Bangla)*.

## 3 DISTRACTMATH-BN Dataset

To systematically evaluate the robustness of mathematical reasoning under realistic noise conditions, we construct DISTRACTMATH-BN, a distractor-augmented benchmark derived from Bangla subset of MGSM *(250 problems)* and MSVAMP *(1000 problems)*. For each problem in the source datasets, we generate *three* distinct augmentation variants, each targeting a different cognitive inter-

| Type | Problem Text |
|---|---|
| **Orig. Ans: 8** | জুলিয়া ① সোমবার ③ ১৮ ② জন ছেলেমেয়ের ② সঙ্গে ট্যাগ খেলেন। মঙ্গলবার ③ তিনি ১০ ② জন ছেলেমেয়ের সঙ্গে ট্যাগ খেলেন। সোমবার তিনি কতজন ছেলেমেয়ের সঙ্গে আরও খেলেন ?<br>*Julia ① played tag with 18 ② children ② on Mon ③. On Tue ③ she played with 10 ② children. How many more on Mon?* |
| ① **RED** Related Entity | মঙ্গলবার ③ মাঠে আরও ১৫ জন ছেলেমেয়ে ② খেলতে এসেছিল। তার ছোট বোন বুধবার ১২ জন ছেলেমেয়ের ② সঙ্গে লুকোচুরি খেলেছিল।<br>*15 more children ② came on Tue ③. Her sister played with 12 children ② on Wed.*<br>↪ ② *same item (children),* ③ *same time (Tue); different actors* |
| ② **OAD** Ortho. Attrib. | সোমবার ③ খেলতে ১ ঘণ্টা সময় লেগেছিল। মঙ্গলবার ③ খেলতে ৪৫ মিনিট সময় লেগেছিল।<br>*It took 1 hour on Mon ③. It took 45 min on Tue ③.*<br>↪ ③ *same events (Mon/Tue); orthogonal dim. (time ⊥ count)* |
| ③ **NEED** Null-Eff. Event | জুলিয়া ① ভেবেছিলেন আরও ৫ জন ছেলেমেয়েকে সোমবার ③ খেলায় ডাকবেন, কিন্তু পরে আর ডাকলেন না । যদি সোমবার বৃষ্টি না হতো, আরও ৩ জন যোগ দিত, কিন্তু শেষ পর্যন্ত আসেনি ।<br>*Julia ① thought of inviting 5 more on Mon ③, but didn't . If no rain on Mon, 3 would've joined, but didn't come .*<br>↪ ① *same subject (Julia),* ③ *same time (Mon);* **negation** *cancels effects* |

subj. ① entity ② event ③ negation red = distractor **bold** = new val.

Table 1: Distractor augmentation (DISTRACTMATH-BN). Markers ① subject, ② entity, ③ event link distractors to original anchors. Answer (18−10=8) is unchanged.

ference mechanism.

**Related Entity Distractors (RED)** augments problems with numerical information about the same object type as the query target, but associated with different entities or contexts. These sentences introduce quantities pertaining to related individuals, locations, or situations that are irrelevant to solving the problem. The distractors test whether models can distinguish between numerically relevant facts about the focal entity versus similar facts about peripheral entities (details in A.10.1).

**Orthogonal Attribute Distractors (OAD)** adds supplementary properties to objects already present in the problem, where these properties are measured in dimensions orthogonal to the queried attribute. This tests whether models can identify which numerical attributes contribute to the solution when multiple properties of identical entities are specified (details in A.10.2).

**Null-Effect Event Distractors (NEED)** introduces descriptions of events involving the problem's main entities that have zero net impact on the final answer. These include actions that were planned but not executed, operations that were performed then reversed, or counterfactual sce-

narios explicitly marked as unrealized. NEED tests whether models correctly interpret linguistic markers of negation, cancellation, and hypotheticality when processing event sequences (details in A.10.3).

**Design Principles and Ecological Validity** Our distractors are *structurally anchored* to the original problem unlike prior work that evaluates robustness using randomly inserted or semantically unrelated sentences (Shi et al., 2023). As illustrated in Table 1, each distractor sentence is grounded in existing entities, subjects, or events from the problem. This closely aligns with how real-world math problems naturally contain plausible but computationally irrelevant information. This ensures ***naturalness of the augmented problems***: because distractors reference existing entities, events, and attributes rather than introducing unrelated context, they are semantically integrated into the problem and cannot be filtered through surface-level heuristics, unlike randomly appended irrelevant sentences. To preserve problem integrity, our generation pipeline uses structured JSON prompting (Apps. A.10) that instructs the model to produce *only candidate distractor sentences with specified insertion positions*, rather than paraphrasing or rewriting the original problem. This ensures the core problem text and its solution remain untouched while the augmented problem reads as a natural, coherent Bangla math problem. Full examples across both MGSM and MSVAMP for all three distractor types are provided in Table A.11.

**Quality Assurance** Each augmented problem undergoes a two-stage verification pipeline: (1) automated validation using `GPT-4.1` to confirm answer preservation (prompt in A.10.4), and (2) human expert review by a native Bangla speaker with mathematical training. Only samples passing both stages are retained, yielding 738 verified augmented problems for MGSM and 2947 for MSVAMP.

We report the distribution of distractor nodes across training and evaluation data in Table A.7: augmented evaluation problems contain 2.7 distractors on average, while training data averages only 0.21 - these are naturally occurring distractors identified by the teacher model (`GPT-4.1`) during graph generation, not artificially injected. No distractor augmentation is applied to training data; this gap is intentional, as it allows us to test whether learning to model distractor nodes through the graph structure alone, refined via GRPO, is sufficient to induce robustness under substantially heavier distractor loads than seen during training.

## 4 Methodology

### 4.1 Computational Graph Framework

CoT prompting and code generation (PoT/PAL) have no mechanism in their output to distinguish relevant quantities from distractors - distractors can be silently incorporated without the model ever committing to their relevance (Section 6). Reasoning-specialized models reduce this through extended computation, and their traces have been shown to encode implicit graph-like structures (Minegishi et al., 2025). We propose making this structure explicit: the model constructs an executable directed acyclic graph that assigns each identified quantity a relevance annotation, followed by deterministic execution over only the relevant subgraph.

We reformulate mathematical word problem solving as a structured graph generation task. Given a problem statement $\mathcal{P}$, our approach instructs a language model to construct a directed acyclic graph $\mathcal{G} = (V, E)$ (prompt in A.10.5). Each node $v$ is assigned a unique identifier and includes a boolean attribute $\texttt{distractor}(v)$ that indicates whether the node contributes to the solution path. Edges $e \in E$ encode data dependencies, where an edge from node $u$ to $v$ indicates that $v$ consumes the output of $u$ as input. The graph structure enforces a topological ordering that enables deterministic execution.

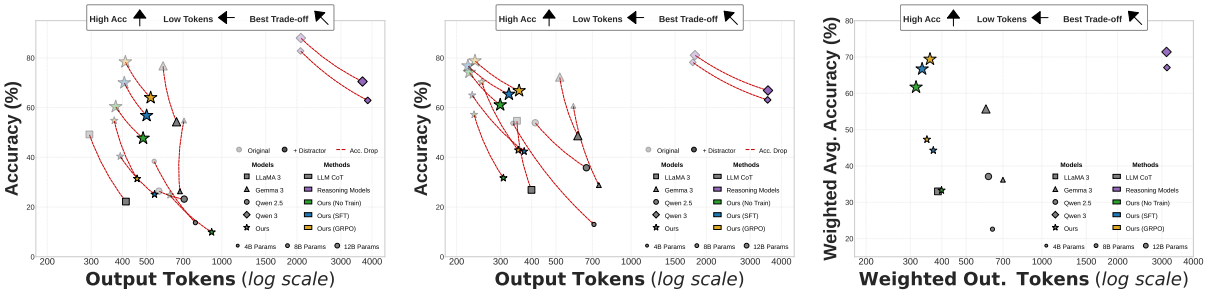
A distinguished node $v_{\text{final}} \in V$ is designated as the final result, representing the problem's solution. Graph execution proceeds through recursive computation: starting from $v_{\text{final}}$, we traverse backwards through dependencies, computing each node's value from its inputs until all constant nodes are reached (Geh and Mauá, 2021). This execution model provides verifiable correctness since each computational step is explicitly represented and can be independently validated (A.8).

The graph representation offers several advantages over sequential reasoning chains. The explicit `distractor` annotation enables the model to identify irrelevant numerical information during construction, preventing error propagation. The structured format separates problem understand-

| | LLM (5-shot CoT) | | | | | | | | | | Reasoning Models | | | | †DAGGER (Gemma 3) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Qwen 2.5 | | | | LLaMA 3 | | Gemma 3 | | | | Qwen 3 | | | | Gemma 3 w/o train. | | | | Gemma 3 SFT | | | | Gemma 3 GRPO | | | |
| | 3B | | 7B | | 8B | | 4B | | 12B | | 4B | | 8B | | 4B | | 12B | | 4B | | 12B | | 4B | | 12B | |
| **Metric** | MG | MS | MG | MS | MG | MS | MG | MS | MG | MS | MG | MS | MG | MS | MG | MS | MG | MS | MG | MS | MG | MS | MG | MS | MG | MS |
| Accuracy (Original) | 38.4 | 53.7 | 26.4 | 53.9 | 49.2 | 54.7 | 54.8 | 60.7 | 76.8 | 72.3 | **82.8** | 78.2 | **88.0** | **81.1** | 24.8 | 57.2 | 60.4 | 74.2 | 40.4 | 65.0 | 70.0 | 76.8 | 54.8 | 70.3 | 78.4 | **78.8** |
| Accuracy (+Distractor) | 13.7 | 13.0 | 23.2 | 35.8 | 22.2 | 26.9 | 26.3 | 28.8 | 54.3 | 48.7 | 62.9 | 63.1 | **70.5** | **66.9** | 9.9 | 31.7 | 47.7 | 61.1 | 25.1 | 42.4 | 56.8 | **65.4** | 31.4 | 42.9 | 64.0 | **66.8** |
| Δ Accuracy (↓) | 24.7 | 40.7 | **3.2** | 18.1 | 27.0 | 27.8 | 28.5 | 31.9 | 22.5 | 23.6 | 19.9 | 15.1 | 17.5 | 14.2 | 14.9 | 25.5 | **12.7** | 13.1 | 15.3 | 22.7 | 13.2 | **11.4** | 23.4 | 27.4 | 14.4 | **12.0** |
| **Weighted Avg. Acc.** | 22.6 | | 37.1 | | 33.0 | | 36.2 | | 55.7 | | **67.1** | | **71.4** | | 33.3 | | 61.7 | | 44.3 | | **66.7** | | 47.3 | | **69.4** | |
| Tokens (Original) | 536 | 336 | 560 | 413 | 295 | 349 | 706 | 587 | 582 | 518 | 2068 | 1767 | 2074 | 1802 | 620 | **235** | 376 | **225** | 392 | **231** | 407 | **222** | 371 | 252 | 411 | 237 |
| Tokens (+Distractor) (↑) | 784 | 710 | 707 | 662 | 413 | 399 | 680 | 743 | 659 | 612 | 3842 | 3522 | 3662 | 3534 | 911 | **308** | 484 | **299** | 538 | 373 | 500 | **324** | 458 | **353** | 519 | 356 |
| **Weighted Out. Tokens** | 636 | | 613 | | 386 | | 700 | | 599 | | 3140 | | 3128 | | 399 | | **316** | | 370 | | **334** | | **349** | | **359** | |

MG = MGSM, MS = MSVAMP; w/o train. = without training; GRPO = SFT→GRPO; ↓ = lower is better, ↑ = higher indicates longer generation. Top 4 results per row are shown in bold.

Table 2: **Impact of Distractor Augmentation on Mathematical Reasoning across MGSM and MSVAMP (Bangla).** Color coding distinguishes 3 model categories. Weighted metrics provide overall performance.




(a) **MGSM** (b) **MSVAMP** (c) **Weighted Average***

Figure 2: **Accuracy vs. Efficiency.** We compare our method (★) against CoT baselines and reasoning models. Dashed arrows in **(a)** and **(b)** show performance drops due to distractors. Marker size indicates model parameter count. Overall, our method approaches reasoning model accuracy with significantly lower computational cost.

ing from computation, allowing errors to be localized to specific operations. Finally, the representation's token budget scales with problem complexity rather than explanation verbosity as shown in Section 6.5.

### 4.2 Training Pipeline

We leverage `GPT-4.1` as a teacher model to generate computational graph representations for training. For SFT, we curate 3K examples: 1.5K from the SOMADHAN dataset and 1.5K from numina-math-cot-bn (Paul et al., 2025; LI et al., 2024; Rahman, 2024) *without any distractor augmentation*. Each problem is processed by `GPT-4.1` to generate structured graph outputs, which are subsequently executed using our graph interpreter (Algorithm A.8 in Apps.) to verify computational correctness. Only verified samples are retained for training. For GRPO, we use 3K questions from numina-math-cot-bn without pre-generated solutions, allowing the model to explore multiple solution paths during training. The GRPO reward function $R(g, y)$ for a generated graph $g$ and ground truth $y$ is defined as

$$R(g, y) = 0.5\,\mathbb{I}_{\text{fmt}} + 0.5\,\mathbb{I}_{\text{exec}} + \mathbb{I}_{\text{acc}}(\text{exec}(g), y)\,,$$

where $\mathbb{I}_{\text{fmt}}$, $\mathbb{I}_{\text{exec}}$, and $\mathbb{I}_{\text{acc}}$ are indicator functions corresponding to (i) a valid JSON schema with at least three nodes[2], (ii) successful topological execution, and (iii) an exact numerical match between the executed graph output and $y$, respectively. A natural concern with GRPO training is reward hacking (Guo et al., 2025; Dipta et al., 2026a,c) - whether the model can achieve high reward without genuine reasoning. Our reward design mitigates this: $\mathbb{I}_{\text{acc}}$ is computed by *executing* the generated graph, so the model cannot reach the correct answer without first producing a structurally valid and executable computational path. A model that satisfies format and execution checks but produces incorrect answers receives at most $0.5 + 0.5 = 1.0$ out of a maximum reward of 2.0, providing limited learning signal without genuine reasoning improvement. We verify that GRPO does not collapse graphs into trivial structures through reward hacking in Appendix A.4.

[2]This constraint prevents models from taking shortcuts by answering the question with a single node.

We structure training as SFT followed by GRPO because they address complementary bottlenecks. SFT requires ground-truth graph annotations from a teacher model, limiting scale to 3K examples, but teaches the model to produce well-formed executable graphs. GRPO needs only final numerical answers, allowing exploration of diverse solution paths through multi-sample generation - but converges poorly without a strong initialization (Dipta et al., 2026b). SFT therefore provides the structural initialization that allows GRPO to focus on reasoning quality - an effect confirmed by the ablation in Appendix A.2, where SFT→GRPO consistently outperforms Base→GRPO.

## 5 Experimental Setup

**Models** We evaluate three model categories on DistractMath-BN: **(1)** standard LLMs with 5-shot CoT prompting: Qwen 2.5 (3B/7B), LLaMA 3 8B, Gemma 3 (4B/12B); **(2)** reasoning-specialized models with extended thinking budgets (up to 7,168 output tokens): Qwen 3 (4B/8B); **(3)** †Dagger-trained Gemma-3 (4B/12B) using SFT followed by GRPO. Training details are in Apps. A.6.

**Supplementary Evaluation** To assess generality beyond fine-tuned models and Bangla, we additionally evaluate GPT-4.1 in few-shot settings with four prompting strategies (CoT (Wei et al., 2022), PoT (Chen et al., 2022), GoT (Besta et al., 2024), †Dagger) on distractor-augmented Bangla data, and with CoT and †Dagger on original MGSM for Telugu, Thai, English, and Spanish (Sec. 6.4).

**Metrics** We report accuracy on each evaluation split, accuracy drop ($\Delta$ = original − augmented) as a robustness measure, and average output tokens as an efficiency measure. We compute weighted average accuracy across all four Bangla splits following dataset-size weighting (Apps. A.1). For †Dagger, we additionally report execution rate (fraction of generated graphs that execute successfully). Failed executions are considered incorrect.

## 6 Results and Analysis

### 6.1 Distractor Impact on CoT and Reasoning Models

Table 2 presents the full evaluation on DistractMath-BN. We organize analysis around two findings.

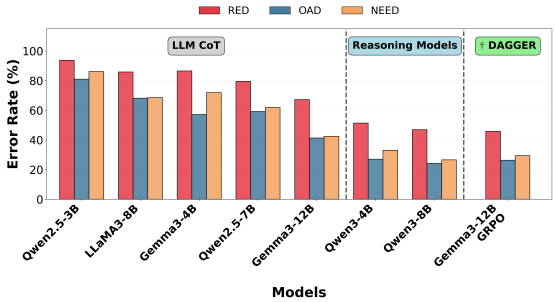



Figure 3: Error rates across distractor types for different model categories.

**CoT Prompting Fails Catastrophically** Standard 5-shot CoT prompting shows severe degradation under distractors: accuracy drops up to 40.7 points, with Qwen 2.5 (3B) falling from 53.7% to 13.0% on MSVAMP. LLaMA 3 8B shows a notably small drop of 3.2 points on MGSM, likely attributable to its already low baseline (26.4%). Figure 3 reveals vulnerabilities across all distractor categories. Related Entity Distractors (RED) are most disruptive (46–94% error rate), indicating persistent difficulties in entity-level disambiguation: models cannot reliably distinguish quantities about the focal entity from those about peripheral entities. Orthogonal Attribute (24–81%) and Null-Effect (27–86%) distractors also cause substantial errors, confirming that the vulnerability is systematic rather than category-specific.

**Reasoning Models: Robust but Expensive** Reasoning-specialized models perform substantially better: Qwen 3-8B achieves 71.4% weighted accuracy, the highest among all evaluated models. However, they still degrade 14.2–19.9 points and consume ∼5.2× more tokens than standard models (3,128 vs. ∼600 tokens per problem). This establishes a key tension: achieving distractor robustness through extended reasoning comes at significant computational cost, motivating the search for more efficient alternatives, particularly for low-resource languages.

### 6.2 †Dagger: Executable Graph Reasoning for Efficiency

†Dagger addresses the robustness–efficiency tension identified above. Figure 2 shows that Gemma-3 12B with SFT→GRPO achieves 69.4% weighted accuracy using only 359 tokens per problem.

**Robustness** The 12B GRPO model drops only 12.0–14.4 points under distractors, compared to 14.2–17.5 for Qwen 3-8B and up to 40.7 for

| | 4B | | 12B | |
|---|---|---|---|---|
| **Annot.** | **Orig.** | **+Dist.** | **Orig.** | **+Dist.** |
| w/o | 64.9 | 39.5 | 76.2 | 60.4 |
| w/ | 65.0 | 42.4 | 76.8 | 65.4 |

Table 3: Ablation: effect of distractor node annotations on MSVAMP accuracy (%). Annotations provide minimal impact on clean data but yield +2.9 (4B) and +5.0 (12B) gains under distractor augmentation.

standard CoT models. This improved robustness emerges from the structured representation: the computational graph requires the model to commit to specific nodes and dependencies, rather than processing all tokens uniformly. Importantly, †DAGGER is not trained on distractor-augmented examples (Appendix A.7): training data averages only 0.21 naturally present distractor nodes versus 2.7 in evaluation, suggesting that the graph structure itself provides an inductive bias for distractor filtering.

**Efficiency** †DAGGER uses 89% fewer tokens than reasoning models (359 vs. 3,128) by encoding mathematical relationships as graph edges rather than verbose natural language explanations.

**Accuracy–Efficiency Trade-off** We note that Qwen 3-8B achieves higher absolute accuracy (71.4% vs. 69.4%). However, Qwen 3 is a reasoning-specialized model that underwent extensive RL training specifically for mathematical reasoning (Yang et al., 2025a) and relies on extended test-time computation. Our approach applies only 3K SFT + 3K GRPO examples to Gemma-3, a general-purpose model. The small accuracy gap comes with $\sim$9$\times$ fewer tokens and better robustness, representing a practical trade-off for resource-constrained deployment.

**Capacity Threshold** Graph-based reasoning requires sufficient model capacity to be effective. Without any training, †DAGGER achieves 33.3% weighted accuracy at 4B but 61.7% at 12B - the latter exceeding all the CoT baselines - indicating that larger models can leverage the graph structure even in a few-shot prompting setup. With SFT and GRPO, the 4B model improves to 47.3% but still lags substantially behind the 12B model (69.4%), suggesting a practical lower bound on model size for constructing well-formed computational graphs, mirroring the capacity floor reported for Bangla math reasoning at smaller scales (Dipta et al., 2026b).

| | CoT | PoT | GoT | †D. |
|---|---|---|---|---|
| DISTRACTMATH-BN (MGSM) | | | | |
| Accuracy | 64.9 | 56.0 | 66.8 | **69.2** |
| Exec. Rate | — | 94.8 | — | 98.4 |
| DISTRACTMATH-BN (MSVAMP) | | | | |
| Accuracy | 63.2 | 49.6 | 58.6 | **65.2** |
| Exec. Rate | — | 96.4 | — | 99.7 |

Table 4: GPT-4.1 results on distractor-augmented DISTRACTMATH-BN.

**Effect of Distractor Annotations** To isolate the contribution of explicit distractor node annotations beyond graph structure alone, we perform ablation experiment by training models using SFT with identical setup but remove all distractor annotations from the graph schema (Table 3).

On clean data, distractor annotations have minimal effect (+0.1/+0.6), as expected since original problems contain few distractors. Under distractor augmentation, annotations yield clear gains: +2.9 for 4B and +5.0 for 12B. The benefit increases with model scale, suggesting larger models better leverage the annotation signal. We note that consistent trends in Table 2 between SFT and GRPO suggest a similar pattern would hold after policy optimization.

### 6.3 Comparison with Structured Reasoning

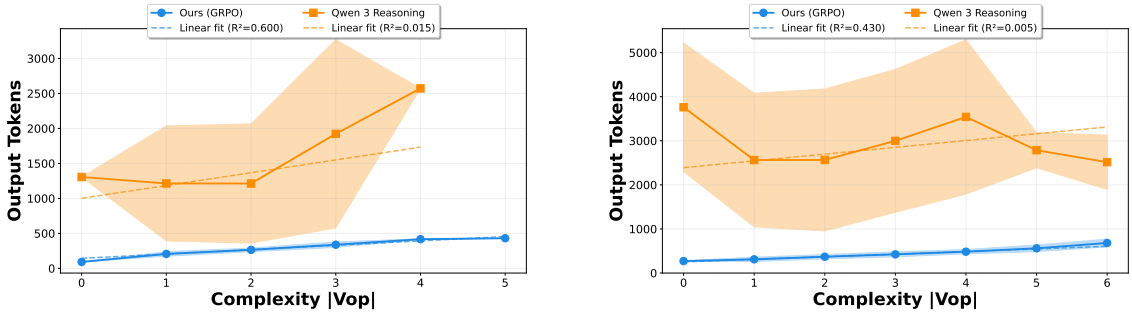
To contextualize †DAGGER against alternative structured approaches, we evaluate GPT-4.1 with CoT, PoT, GoT, and †DAGGER on distractor-augmented DISTRACTMATH-BN (Table 4).

†DAGGER achieves the highest accuracy on both datasets. Notably, GoT requires $\sim$6 API calls per problem while all other methods use a single call, yet still underperforms †DAGGER by 2.4 and 6.6 points on MGSM and MSVAMP respectively. PoT underperforms significantly despite high execution rates (94–96%), confirming that successful code execution does not guarantee correct reasoning under distractors (Xu et al., 2025): sequential code generation, like free-form CoT, processes all input tokens uniformly. GoT's multi-call approach provides some robustness through exploration of alternative paths, but at substantially higher cost and without explicit distractor filtering.

### 6.4 Low-Resource Language Results

To assess whether †DAGGER's benefits extend beyond Bangla and fine-tuned models, we evaluate GPT-4.1 on MGSM across four languages (Table 5).

†DAGGER consistently shows strong perfor-




(a) MSVAMP-BN (original)

(b) MSVAMP-BN + distractors (DISTRACTMATH-BN)

Figure 4: Output token usage as a function of computational complexity ($|V_{\text{op}}|$).

| Language | CoT (5-shot) | †DAGGER (3-shot) |
|---|---|---|
| English | 78.4 | **89.2** |
| Spanish | 75.2 | **85.6** |
| Telugu | 74.4 | **78.8** |
| Thai | 78.4 | **87.6** |

Table 5: GPT-4.1 accuracy (%) on MGSM across four languages. Execution rates for †DAGGER exceed 93% in all cases.

mance compared to CoT across all four languages, with improvements ranging from +4.4 on Telugu to +10.8 on English. High execution rates (>93%) across all languages indicate that graph-based representations maintain structural validity across diverse linguistic contexts. However, the CoT results are sensitive to prompt choice: Table 5 reports the original MGSM 5-shot ICL setup (78.4%), while alternate configurations obtain 80.0–89.6% accuracy (Appendix A.3). Thus, Table 5 represents a fixed-prompt comparison.

These results suggest that the structured graph representation generalizes beyond Bangla, though the primary contribution of this work remains the systematic evaluation of distractor robustness and efficiency in the Bangla setting.

### 6.5 Token Scaling with Computational Complexity

We investigate whether output length in †DAGGER scales with problem complexity rather than explanation verbosity. We define complexity as $|V_{\text{op}}|$, the number of operation nodes (excluding `const` and `identity`) in the ground-truth computational graph produced by the teacher model. This directly reflects how many arithmetic steps are needed to solve a problem.

We compare †DAGGER (Gemma-3-12B, SFT→GRPO) against Qwen-3-8B Reasoning on MSVAMP-BN, both on original and distractor-augmented splits. Each baseline output is assigned the $|V_{\text{op}}|$ of its corresponding ground-truth graph. The analysis is restricted to correctly solved instances (Figures 4a and 4b).

On the original split, †DAGGER output length correlates well with $|V_{\text{op}}|$ ($R^2$=0.600), staying in the hundreds of tokens with low variance. Qwen-3-8B shows almost no correlation ($R^2$=0.015), spanning thousands of tokens regardless of complexity. Under distractor augmentation, the pattern persists: †DAGGER maintains meaningful correlation ($R^2$=0.430) while the reasoning baseline remains near-zero ($R^2$=0.005). These results show that graph-output length is associated with the number of operation nodes in the same generated graph, whereas free-form reasoning length is largely independent of problem complexity.

## 7 Discussion

Standard CoT prompting is brittle under distractors, dropping up to 40.7 points on Bangla MWPs across all distractor categories (Figure 3). Reasoning models reduce this gap but still degrade 14–20 points at ∼9× the token cost.

†DAGGER addresses this by forcing explicit dependency commitments through the DAG structure, preventing distractors from entering the solution path, while distractor annotations allow acknowledging irrelevant quantities without incorporating them. This robustness emerges without distractor-augmented training (0.21 distractor nodes in training vs. 2.7 at evaluation), suggesting the graph structure itself induces filtering. Output length tracks computational complexity ($R^2$=0.600 vs. $R^2$=0.015 for free-form reasoning), yielding 89% fewer tokens at comparable accuracy.

Related Entity Distractors remain hardest across all models (Figure 3), suggesting entity-level disambiguation may require richer representations beyond node-level annotations.

## 8 Conclusion

We introduce DistractMath-BN, revealing that standard CoT prompting fails catastrophically under distractors while reasoning-specialized models achieve robustness at prohibitive computational cost. †DAGGER demonstrates that structured computational graph generation enables comparable accuracy with substantially improved token efficiency. Critically, this robustness emerges without explicit distractor training. Our findings suggest structured representations provide a practical pathway for robust mathematical reasoning in noisy low-resource settings, though entity-level semantic disambiguation remains an open challenge.

## Limitations

Our systematic distractor generation employs JSON-structured outputs to ensure semantic consistency, defining distractors operationally as numerical values not reached during backward graph traversal. However, our benchmark is limited to arithmetic word problems in Bangla with only three distractor categories (Related Entity, Orthogonal Attribute, and Null-Effect Event) and does not address non-numeric distractors such as irrelevant textual descriptions. The quality assurance process relied on review by a single native Bangla-speaking PhD researcher, which may introduce individual judgment biases.

Our computational graph framework assumes problems can be decomposed into directed acyclic graphs with atomic operations, which may not extend to geometric reasoning, algebraic manipulation, or mathematical proof. It is designed for deterministic solutions and cannot represent probabilistic reasoning or problems with multiple valid solution paths. The complexity of generating well-formed graphs creates practical capacity requirements, suggesting an empirical lower bound on model size for effective performance, though smaller models still show improvements over baselines.

Our approach demonstrates benefits even without supervised fine-tuning, though combining SFT with GRPO yields optimal performance. The SFT phase creates a dependency on constructing high-quality initial training examples, which may limit scalability to resource-intensive annotation scenarios. While we demonstrate improvements for frontier models like GPT-4.1 in few-shot settings, our fine-tuning experiments are limited to the Gemma model family and may require adaptation for other architectures. The cross-lingual evaluation on four languages used limited test sets without distractor augmentation, providing preliminary evidence for transferability. Our evaluation relies on exact numeric match, which may not capture reasoning quality or penalize minor arithmetic errors proportionately. Finally, this work focuses on elementary arithmetic word problems and should not be generalized to advanced mathematical reasoning including algebra, geometry, or calculus.

## Ethics Statement

This work addresses the gap in NLP capabilities for Bangla, a widely spoken but underrepresented language in language technology. We have made DistractMath-BN, code, and prompts publicly available. Dataset construction relies in part on a proprietary large language model, which may introduce biases or stylistic artifacts specific to that model and should be considered when interpreting results or extending the benchmark. Our method is designed to improve mathematical reasoning in educational technology and low-resource language support tools. While the structured representation improves interpretability, responsible deployment requires human oversight, particularly in educational settings.

## References


Kawsar Ahmed, Md Osama, Omar Sharif, Eftekhar Hossain, and Mohammed Moshiul Hoque. 2025. Bennumeval: A benchmark to assess llms' numerical reasoning capabilities in bengali. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 17782–17799.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. 38(16):17682–17690.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

Xiang Cheng, Chengyan Pan, Minjun Zhao, Deyang Li, Fangchao Liu, Xinyu Zhang, Xiao Zhang, and Yong Liu. 2025. Revisiting chain-of-thought prompting: Zero-shot can be stronger than few-shot. *arXiv preprint arXiv:2506.14641*.

Shubhashis Roy Dipta, Daniel Bis, Kun Zhou, Lichao Wang, Benjamin Z. Yao, Chenlei Guo, and Ruhi Sarikaya. 2026a. Pa3: Policy-aware agent alignment through chain-of-thought. *Preprint*, arXiv:2603.14602.

Shubhashis Roy Dipta, Khairul Mahbub, and Nadia Najjar. 2026b. Ganitllm: Difficulty-aware bengali mathematical reasoning through curriculum-grpo. *Preprint*, arXiv:2601.06767.

Shubhashis Roy Dipta, Ankur Padia, and Francis Ferraro. 2026c. Decomposerl: Learning to ask useful, informative, and diverse questions for semi-supervised, traceable claim verification. *Preprint*, arXiv:2605.27858.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Renato Lui Geh and Denis Deratani Mauá. 2021. Learning probabilistic sentential decision diagrams under logic constraints by sampling and averaging. In *Uncertainty in Artificial Intelligence*, pages 2039–2049. PMLR.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024. Numinamath. https://huggingface.co/datasets/AI-MO/NuminaMath-CoT.

Zekun Li, Baolin Peng, Pengcheng He, and Xifeng Yan. 2023. Do you really follow me? adversarial instructions for evaluating the robustness of large language models. *arXiv preprint arXiv:2308.10819*.

Minh-Thang Luong, Dawsen Hwang, Hoang H Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, and 1 others. 2025. Towards robust mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 35406–35430.

Gouki Minegishi, Hiroki Furuta, Takeshi Kojima, Yusuke Iwasawa, and Yutaka Matsuo. 2025. Topology of reasoning: Understanding large reasoning models through reasoning graph properties. *arXiv preprint arXiv:2506.05744*.

Zabir Al Nazi, Shubhashis Roy Dipta, and Md Rizwan Parvez. 2026. Omni-modal dissonance benchmark: Systematically breaking modality consensus to probe robustness and calibrated abstention. *Preprint*, arXiv:2603.27187.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.

Bidyarthi Paul, Jalisha Jashim Era, Mirazur Rahman Zim, Tahmid Sattar Aothoi, and Faisal Muhammad Shah. 2025. Leveraging large language models for bengali math word problem solving with chain of thought reasoning. *arXiv preprint arXiv:2505.21354*.

Tabia Tanzin Prama, Christopher M Danforth, and Peter Dodds. 2025. Banglamath: A bangla benchmark dataset for testing llm mathematical reasoning at grades 6, 7, and 8. In *Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025)*, pages 134–149.

Ashrafur Rahman. 2024. NuminaMath-CoT-bn: Numinamath-cot fully translated to bengali. https://www.kaggle.com/datasets/risenfromashes/numinamath-cot-bn. Kaggle dataset. Machine-translated using Gemini-1.5-Flash. License: CC BY-NC-SA 4.0.

Shubhashis Roy Dipta and Francis Ferraro. 2025. If we may de-presuppose: Robustly verifying claims through presupposition-free question decomposition. In *Proceedings of the 14th Joint Conference on Lexical and Computational Semantics (*SEM 2025)*, pages 253–266, Suzhou, China. Association for Computational Linguistics.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. *arXiv preprint arXiv:2302.00093*.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, and 1 others. 2022. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*.

Mahbub E Sobhani, Md Faiyaz Abdullah Sayeedi, Tasnim Mohiuddin, Md Mofijul Islam, and Swakkhar Shatabda. 2025. Mathmist: A parallel multilingual benchmark dataset for mathematical problem solving and reasoning. *arXiv preprint arXiv:2510.14305*.

Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2024. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. *arXiv preprint arXiv:2409.12183*.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

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

Ningning Xu, Yuxuan Jiang, Shubhashis Roy Dipta, and Hengyuan Zhang. 2025. Learning how to use tools, not just when: Pattern-aware tool-integrated reasoning.

Tianyi Xu, Kosei Uemura, Alfred Malengo Kondoro, Tadesse Destaw Belay, Catherine Nana Nyaah Essuman, Ifeoma Okoh, Ganiyat Afolabi, Ayodele Awokoya, and David Ifeoluwa Adelani. 2026. Mgsm-pro: A simple strategy for robust multilingual mathematical reasoning evaluation. *arXiv preprint arXiv:2601.21225*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Kaiqi Yang, Hang Li, Yucheng Chu, Zitao Liu, Mi Tian, and Hui Liu. 2025b. Iterative llm-based generation and refinement of distracting conditions in math word problems. *arXiv preprint arXiv:2510.08615*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Fangxu Yu, Lai Jiang, Haoqiang Kang, Shibo Hao, and Lianhui Qin. 2024. Flow of reasoning: Training llms for divergent problem solving with minimal examples. *arXiv preprint arXiv:2406.05673*.

Abdullah Khan Zehady, Shubhashis Roy Dipta, Naymul Islam, Safi Al Mamun, and Santu Karmaker. 2026. Banglallama: Llama for bangla language. In *Proceedings of the Second Workshop on Language Models for Low-Resource Languages (LoResLM 2026)*, pages 73–89, Rabat, Morocco. Association for Computational Linguistics.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and 1 others. 2023. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

# A Appendix

## Table of Contents


A.1 Impact of Distractor Augmentation on Different Models
A.2 Ablation: Effect of GRPO Initialization (Base vs. SFT)
A.3 CoT Prompt Sensitivity
A.4 Reward Structure Analysis
A.5 Inference Configuration
A.6 Training Configuration
A.6.1 Supervised Fine-Tuning
A.6.2 Group Relative Policy Optimization
A.7 Distractor Distribution Analysis
A.8 Graph Execution Algorithm
A.9 Qualitative Analysis: †DAGGER vs. Chain-of-Thought Reasoning
A.9.1 Example 1
A.9.2 Example 2
A.9.3 Example 3
A.10 Prompts
A.10.1 Generating Related Entity Distractor
A.10.2 Generating Orthogonal Attribute Distractor
A.10.3 Generating Null-Effect Event Distractor
A.10.4 Verifying Modified Problems
A.10.5 Generating Computational Graph
A.10.6 Prompt for Reasoning Models
A.10.7 Prompt for Chain of Thought (5-Shot)
A.11 Examples of Augmented Questions from DistractMath-BN


### A.1 Impact of Distractor Augmentation on Different Models

Table 2 reports the full tabular results corresponding to the trends summarized in Figure 2. Beyond the aggregate view in Figure 2, the table provides per-dataset performance under both original and distractor-augmented settings, together with accuracy drops and output-token statistics.

We compute a weighted average accuracy over four evaluation splits using dataset-size weighting:

$$\mathrm{Acc_w} = \frac{n_{\mathrm{MG}} A_{\mathrm{MG}} + n_{\mathrm{MS}} A_{\mathrm{MS}} + n_{\mathrm{MG}_{+D}} A_{\mathrm{MG}_{+D}} + n_{\mathrm{MS}_{+D}} A_{\mathrm{MS}_{+D}}}{n_{\mathrm{MG}} + n_{\mathrm{MS}} + n_{\mathrm{MG}_{+D}} + n_{\mathrm{MS}_{+D}}}.$$

Here, MG and MS denote the original MGSM and MSVAMP splits, while $\mathrm{MG}_{+D}$ and $\mathrm{MS}_{+D}$ denote their distractor-augmented counterparts. $A_{\cdot}$ represents accuracy on each split, and $n_{\cdot}$ denotes the corresponding number of problems, with $n_{\mathrm{MG}}$=250, $n_{\mathrm{MS}}$=1000, $n_{\mathrm{MG}_{+D}}$=738, and $n_{\mathrm{MS}_{+D}}$=2947.

### A.2 Ablation: Effect of GRPO Initialization (Base vs. SFT)

**Motivation.** While Group Relative Policy Optimization (GRPO) enforces correctness through executable and verifiable rewards, it remains unclear whether GRPO alone is sufficient to induce structured computational graph generation, or whether an intermediate Supervised Fine-Tuning (SFT) phase is necessary to bootstrap this behavior. This ablation isolates the contribution of *initialization* by comparing GRPO trained directly from the original pre-trained checkpoint (*Base→GRPO*) against GRPO initialized from an SFT checkpoint (*SFT→GRPO*).

| Model | MGSM | MSVAMP | MGSM (+D) | MSVAMP (+D) |
|---|---|---|---|---|
| **Gemma-3-4B** | | | | |
| Base→GRPO | 29.2 | 57.1 | 13.1 | 29.3 |
| SFT→GRPO | 54.8 | 70.3 | 31.4 | 42.9 |
| **Gemma-3-12B** | | | | |
| Base→GRPO | 67.6 | 75.0 | 48.4 | 59.6 |
| SFT→GRPO | 78.4 | 78.8 | 64.0 | 66.8 |

Table 6: **Ablation on GRPO initialization.** Accuracy (%) on Bangla MGSM and MSVAMP, with and without distractor augmentation (+D). SFT initialization consistently improves GRPO performance, especially under distractors.

**Setup.** We train two GRPO variants for Gemma-3-4B and Gemma-3-12B using identical GRPO hyperparameters, reward definition, and training data. The only difference is the initialization: (i) **Base→GRPO** starts from the original pre-trained Gemma-3 checkpoint, while (ii) **SFT→GRPO** starts from the SFT checkpoint trained on verified computational graph demonstrations. We evaluate both variants on MGSM and MSVAMP (Bangla), as well as their distractor-augmented counterparts from DistractMath-BN.

**Results.** Table 6 summarizes the results. Across both model sizes and datasets, GRPO initialized from SFT consistently outperforms GRPO trained directly from the base model, with the gap being substantially larger under distractor augmentation.

For Gemma-3-12B, Base→GRPO already achieves reasonable performance on clean data (67.6% on MGSM, 75.0% on MSVAMP), but degrades sharply under distractors (48.4% and 59.6%). Initializing GRPO from SFT yields consistent gains across all settings, with particularly large improvements on augmented MGSM (+15.6 points) and augmented MSVAMP (+7.2 points), indicating improved robustness to irrelevant context.

For Gemma-3-4B, the effect of initialization is even more pronounced. Base→GRPO struggles to learn reliable graph generation, achieving only 29.2% on MGSM and collapsing to 13.1%

on augmented MGSM. In contrast, SFT→GRPO nearly doubles accuracy on MGSM (54.8%) and more than doubles performance under distractors (31.4% on augmented MGSM). This suggests that smaller models require SFT to learn basic structural and execution constraints before GRPO can effectively refine reasoning strategies.

Overall, these results show that while GRPO can improve performance even from a base checkpoint, *SFT provides a critical scaffold* that stabilizes learning and enables GRPO to focus on semantic parsing and distractor filtering rather than merely discovering valid executable structures. This explains why SFT→GRPO yields the strongest robustness and accuracy throughout our experiments.

### A.3 CoT Prompt Sensitivity

To assess sensitivity to in-context examples and instruction format, we evaluate GPT-4.1 on English MGSM using four CoT prompting configurations - three different ICL settings and one without ICL (Xu et al., 2026). Table 7 shows substantial variation across prompts.

| Configuration | Accuracy (%) |
|---|---|
| 5-shot CoT, original MGSM ICL set | 78.4 |
| 5-shot CoT, alternate MGSM ICL set | 80.0 |
| 3-shot CoT, alternate MGSM ICL set | 82.4 |
| MGSM-Pro instruction prompt (no ICL) | **89.6** |

Table 7: Prompt sensitivity of GPT-4.1 CoT on English MGSM.

### A.4 Reward Structure Analysis

We analyze graph complexity and reward component distributions across training phases to assess whether GRPO training produces structurally meaningful improvements.

**Graph Complexity Across Training Phases** Table 8 reports the average number of nodes per generated graph at each stage of training, evaluated on MSVAMP using Gemma-3 12B. Average node count increases from 7.33 (without training) to 7.74 (SFT) to 7.84 (GRPO), indicating that GRPO does not reduce graph complexity relative to SFT.

| Phase | Avg. Nodes |
|---|---|
| Without training | 7.33 |
| After SFT | 7.74 |
| After GRPO | 7.84 |

Table 8: Average number of nodes per generated graph across training phases (Gemma-3 12B, MSVAMP).

**Reward Component Breakdown** After SFT, format compliance ($\mathbb{I}_{\text{fmt}}$) and execution success ($\mathbb{I}_{\text{exec}}$) are already high, leaving limited room for improvement on these components. The observed gains from GRPO (+2.7 weighted points for 12B, +3.0 for 4B) are concentrated in $\mathbb{I}_{\text{acc}}$: the model produces graphs that execute to the correct answer more frequently, while format and execution scores remain largely unchanged.

**Reward Hacking Prevention** The compositional reward design imposes several structural constraints that limit reward exploitation. Format compliance requires valid JSON with at least 3 nodes, ruling out degenerate single-operation graphs. Execution success requires all node dependencies to resolve during topological traversal - referencing a nonexistent dependency causes execution failure and zero execution reward. The accuracy reward evaluates the numerical output of graph execution against ground truth, not the graph structure itself: a graph that is well-formatted and executes successfully but computes the wrong result receives zero accuracy reward. Combined with the empirical evidence above - stable graph complexity and gains concentrated in accuracy - these constraints prevent reward hacking and improve genuine reasoning and distractor identification capabilities.

### A.5 Inference Configuration

For all chain-of-thought (CoT) baselines, we use the default inference configuration provided by the underlying model implementation. Models are loaded in full precision without quantization, using BF16 arithmetic and a maximum sequence length of 4096 tokens. During inference, we allow up to 2048 output tokens to accommodate multi-step reasoning traces. Unless otherwise specified, all remaining decoding and runtime hyperparameters are kept at their default values.

For reasoning-specialized models, we employ a larger context window to support longer reasoning traces. Specifically, models are loaded with a maximum sequence length of 8192 tokens in BF16 precision. To prevent premature truncation of extended reasoning chains, we allow up to 7168 output tokens during generation. All other inference-time settings follow the default configuration of the respective models.

For DAGGER-style inference without training, we use the same default inference setup as above

but restrict the maximum number of generated output tokens to 1024.

## A.6 Training Configuration

### A.6.1 Supervised Fine-Tuning

We fine-tune the base Gemma-3 models (4B and 12B) using Rank-Stabilized LoRA (RSLoRA) with rank 64. The training uses a global batch size of 256 across 4 epochs with AdamW optimizer. We employ cosine learning rate scheduling starting from $1 \times 10^{-5}$ with minimum rate of $1 \times 10^{-6}$. Gradient checkpointing and sequence packing are enabled to optimize memory usage. Training is performed in BF16 precision with maximum gradient norm clipping at 1.0.

| Parameter | Value |
|---|---|
| Base Model | Gemma-3 (4B/12B) |
| LoRA Rank / Alpha | 64 / 128 |
| Max Seq Length | 4096 |
| Global Batch Size | 256 |
| Epochs | 4 |
| Learning Rate | $10^{-5} \rightarrow 10^{-6}$ |
| Optimizer | AdamW |
| Weight Decay | 0.001 |
| Precision | BF16 |

Table 9: SFT hyperparameters.

| Parameter | Value |
|---|---|
| Base Model | SFT checkpoint |
| LoRA Rank / Alpha | 64 / 128 |
| Max Prompt / Completion | 2048 / 2048 |
| Global Batch Size | 32 |
| Generations per Prompt | 8 |
| Epochs | 4 |
| Learning Rate | $10^{-5} \rightarrow 10^{-6}$ |
| Loss Type | BNPO |
| $\beta$ / $\epsilon$ / $\epsilon_{\text{high}}$ | 0.0 / 0.2 / 0.28 |
| Sampling (top-p / top-k) | 0.95 / 64 |
| Temperature | 1.0 |
| Precision | BF16 |

Table 10: GRPO hyperparameters.

### A.6.2 Group Relative Policy Optimization

We train the SFT checkpoints using GRPO with a structured reward function that assigns +0.5 for valid JSON graph format, +0.5 for successful execution, and +1.0 for correct answers (maximum 2.0 per generation). We employ GRPO with BNPO loss as it is specifically designed for tasks with verifiable outputs and discrete reward signals, making it well-suited for computational graph generation where correctness can be deterministically evaluated. The group-based policy optimization allows efficient comparison across multiple generated graphs per prompt, enabling the model to learn from relative quality differences. BNPO's binary normalization is particularly effective for our structured three-tier reward system (format validation, execution success, and answer correctness). For each prompt, we generate 8 completions using VLLM with top-p sampling (0.95) and top-k (64). The training uses BNPO loss with $\beta = 0.0$, epsilon clipping at 0.2-0.28, and temperature 1.0. We employ LoRA rank 64 with global batch size 32 over 4 epochs. The learning rate follows cosine scheduling from $1 \times 10^{-5}$ to $1 \times 10^{-6}$ with 20 warmup steps.

## A.7 Distractor Distribution Analysis

To better understand the robustness gap between training and evaluation, we analyze the distribution of distractor nodes across the training and augmented evaluation datasets. Table 11 summarizes the average, minimum, and maximum number of distractors per problem.

| Dataset | Avg. Distractors | Min | Max |
|---|---|---|---|
| MGSM (+Distractor) | 2.76 | 2 | 5 |
| MSVAMP (+Distractor) | 2.68 | 2 | 4 |
| SFT Training Data | 0.21 | 0 | 10 |

Table 11: Distribution of distractor nodes across training and augmented evaluation datasets.

As shown in Table 11, the augmented MGSM and MSVAMP evaluation sets contain substantially more distractors per problem than the data used for supervised fine-tuning. This imbalance arises because no explicit distractor augmentation is applied during SFT; instead, training samples are randomly drawn from SOMADHAN and Numina Math CoT datasets and no distractor augmentation is performed, hence naturally most examples contain few or no distractor nodes. As a result, the average number of distractors in training remains very low despite a small number of outliers.

Importantly, although the model is not trained on augmented examples, it still demonstrates improved robustness under distractor-heavy evaluation. This suggests that the structured design of distractor nodes in the computational graph formulation itself provides a strong inductive bias for robustness. We expect that explicitly applying distractor augmentation during training would further improve performance; however, these results confirm that meaningful robustness gains can already be achieved through graph-level distractor design alone, even without direct exposure to augmented data.

## A.8 Graph Execution Algorithm

We present the high-level pseudo-code for executing the computational graph. The algorithm recursively computes node values while handling distractors through topological traversal.

**Algorithm: Computational Graph Execution**

**Input:** Graph JSON: $G = \{V, E\}$ where $V$ are nodes
**Output:** Final numeric result $r \in \mathbb{R}$

```
FUNCTION ExecuteGraph(G):
    N ← PARSE(G)                                   [Parse & validate]
    cache ← ∅,   visited ← ∅                       [Initialize]
    v_final ← FINDFINALNODE(N)
    return COMPUTE(v_final, N, cache, visited)

FUNCTION Compute(v, N, cache, visited):
    if v ∈ cache:
        return cache[v]                            [Memoized]
    if v ∈ visited:
        raise CIRCULARDEPENDENCY                   [Cycle detect]
    visited ← visited ∪ {v}
    node ← N[v]
    op ← node.op
    // Execute operation
    if op = const:
        r ← node.val
    else if op = add:
        r ← Σ_{a∈node.args} RESOLVE(a, ...)
    else if op = sub:
        r ← RESOLVE(node.args[0], ...) − RESOLVE(node.args[1], ...)
    else if op = mul:
        r ← Π_{a∈node.args} RESOLVE(a, ...)
    else if op = div:
        r ← RESOLVE(node.args[0], ...) / RESOLVE(node.args[1], ...)
    else if op ∈ {floor, ceil, round, ...}:
        r ← APPLYOP(op, node.args, ...)
    cache[v] ← r                                   [Cache result]
    visited ← visited \ {v}
    return r

FUNCTION Resolve(a, N, cache, visited):
    if a ∈ ℝ:
        return a                                   [Numeric literal]
    else if a ∈ N:
        return COMPUTE(a, N, cache, visited)       [Node ref]
    else:
        raise INVALIDARGUMENT
```

**Complexity:** $\mathcal{O}(|V| + |E|)$ where $|V|$ is the number of nodes and $|E|$ is the number of edges.

## A.9 Qualitative Analysis: †DAGGER vs. Chain-of-Thought Reasoning

We present qualitative examples demonstrating how †DAGGER correctly identifies and filters distractor information mitigating the limitations of chain-of-thought based approaches for Bangla MWP.

### A.9.1 Example 1

**Dataset:** MGSM-BN
**Augmentation Type:** Related Entity Distractor (RED)

**Question:**
জিনের কাছে 30টি ললিপপ আছে। দোকানে আরও ৫০টি ললিপপ বিক্রি হছিল। তার ছোটবোনের কাছে ১৮টি ললিপপ আছে। জিন 2টি ললিপপ খেয়েছে। জিনের বন্ধু মিমি প্রতিদিন ৩টি ললিপপ খায়। অবশিষ্ট ললিপপগুলি দিয়ে, জিন একটি ব্যাগের মধ্যে 2টি করে ললিপপ ভরতে চায়। তাহলে জিন কতগুলি ব্যাগ ভর্তি করতে পারবে?

*Jean has 30 lollipops. 50 more lollipops were being sold at the store. Her younger sister has 18 lollipops. Jean ate 2 lollipops. Jean's friend Mimi eats 3 lollipops per day. With the remaining lollipops, Jean wants to fill bags with 2 lollipops each. How many bags can Jean fill?*

**Ground Truth Answer: 14**

**Gemma 3 12B (CoT)** Predicted Answer: 48

প্রথমে, জিন 30টি ললিপপ কিনেছে এবং তার ছোটবোনের কাছে 18টি ললিপপ ছিল। জিন 2টি ললিপপ খেয়েছে। তার মানে জিন 30 + 18 - 2 = 46টি ললিপপ নিয়ে শুরু করেছে। এরপর জিন 50টি ললিপপ কিনেছে। তার মানে জিন 46 + 50 = 96টি ললিপপ নিয়ে শুরু করেছে। মিমি প্রতিদিন 3টি করে ললিপপ খায়। এই তথ্যটি এখানে অপ্রাসঙ্গিক। জিন 2টি করে ললিপপ ভরতে চায়। তার মানে জিন 96 / 2 = 48টি ব্যাগ ভর্তি করতে পারবে। উত্তর হল 48।

**Error Analysis:** Incorrectly includes sister's lollipops (18) and store lollipops (50) in Jean's count, leading to wrong calculation: 30 + 18 + 50 - 2 = 96, then 96 / 2 = 48.

**Qwen 3 8B (Reasoning)** Predicted Answer: 23

<think> Okay, let's try to figure out how many bags Jean can fill with the lollipops. Hmm, first, I need to find out the total number of lollipops Jean has. Let me start by adding up all the lollipops from different sources.
So, Jean has 30 lollipops. Then, there are 50 more lollipops being sold at the store. Wait, does that mean those 50 are added to Jean's collection? Or are they separate? The problem says "দোকানে আরও ৫০টি ললিপপ বিক্রি হচ্ছিল" which translates to "there are 50 more lollipops being sold at the store." ... The problem might be that Jean has 30, and then there are 50 more at the store, and her sister has 18. But the question is about Jean's lollipops. So maybe the 50 at the store are not hers. ...
... So maybe the total lollipops Jean has are 30 (hers) + 18 (sister's) + 50 (store's)? But that seems odd. Unless Jean is combining all of them. But the problem doesn't specify that. Alternatively, maybe the 50 at the store are not hers, and the sister's 18 are hers. So Jean has 30 + 18 = 48? Then she ate 2, so 48 - 2 = 46. Then Mimi eats 3 per day, but how many days? Wait, the problem says "জিনের বন্ধু মিমি প্রতিদিন ৩টি ললিপপ খায়" (Jean's friend Mimi eats 3 lollipops per day). But how many days? The problem doesn't mention the number of days. ...
... Maybe the problem is that the 50 at the store are being sold, so Jean might have bought them? But the problem doesn't say she bought them. Alternatively, maybe the total lollipops Jean has are 30 (hers) + 18 (sister's) = 48. Then she ate 2, so 46. Then Mimi eats 3 per day, but again, no days. ...

... Given the ambiguity, I think the most logical assumption is that the 50 at the store are not part of Jean's lollipops, and the sister's 18 are hers. So total lollipops: 30 + 18 = 48. Jean ate 2, so 46. Then Mimi eats 3 per day, but since the problem doesn't specify the number of days, maybe it's a mistake and we should ignore Mimi's consumption. So 46 divided by 2 is 23 bags. But this is also a guess. ... So the answer is 23.

</think>

**Error Analysis:** Confused by distractor information; incorrectly adds sister's lollipops (18) to Jean's count, resulting in (30 + 18 - 2) / 2 = 23.

**†DAGGER (Ours)** Predicted Answer: 14


```
{
 "nodes": [
{"id": "jin_start", "op": "const", "val": 30,
 "distractor": false,
 "label": "জিনের প্রাথমিক ললিপপ"},
{"id": "store_lollipop", "op": "const", "val": 50,
 "distractor": true,
 "label": "দোকানে বিক্রি হওয়া ললিপপ"},
{"id": "sister_lollipop", "op": "const", "val": 18,
 "distractor": true,
 "label": "ছোট বোনের ললিপপ"},
{"id": "jin_eaten", "op": "const", "val": 2,
 "distractor": false,
 "label": "জিন খেয়েছে"},
{"id": "jin_left", "op": "sub", "args": ["jin_start",
 "jin_eaten"],
 "distractor": false, "label": "জিনের অবশিষ্ট"},
{"id": "mimmi_eat", "op": "const", "val": 3,
 "distractor": true,
 "label": "মিমি প্রতিদিন খায়"},
{"id": "bag_capacity", "op": "const", "val": 2,
 "distractor": false},
{"id": "bags_full", "op": "div", "args": ["jin_left",
 "bag_capacity"],
 "distractor": false},
{"id": "full_bags", "op": "floor", "args":
 ["bags_full"],
 "distractor": false},
{"id": "final_result", "op": "identity", "args":
 ["full_bags"],
 "distractor": false, "label": "চূড়ান্ত উত্তর"}
 ]
}
```


**Success:** Correctly identifies distractors (store=50, sister=18, Mimi=3) and computes only from Jean's lollipops: (30 - 2) / 2 = 14.

### A.9.2 Example 2

**Dataset:** MGSM-BN
**Augmentation Type:** Orthogonal Attribute Distractors (OAD)

**Question:**
পাঞ্চো একদিনে 20 মাইল হাঁটেন। প্রতিদিন হাঁটার জন্য তিনি ১.৫ ঘণ্টা সময় নেন। সপ্তাহান্তে ব্যতিরেকে তিনি 10 মাইল হাঁটেন। সপ্তাহান্তে হাঁটার সময় ৪৫ মিনিট লাগে। এক সপ্তাহে তিনি কত মাইল হাঁটেন?

*Pancho walks 20 miles a day. He takes 1.5 hours per day for walking. Except for weekends, he walks 10 miles. Walking on weekends takes 45 minutes. How many miles does he walk in a week?*

**Ground Truth Answer: 120**

**Gemma 3 12B (CoT)** Predicted Answer: 110

সপ্তাহান্তে ব্যতিরেকে তিনি 10 মাইল হাঁটেন। তিনি 5 দিন ধরে 20 মাইল করে হাঁটেন। সুতরাং, তিনি 5 * 20 = 100 মাইল হাঁটেন। সপ্তাহান্তে তিনি 10 মাইল হাঁটেন। সুতরাং, তিনি 100 + 10 = 110 মাইল হাঁটেন। উত্তর হল 110।

**Error Analysis:** Correctly calculates weekday miles (5 × 20 = 100) but fails to multiply weekend miles by 2 days. Uses 10 instead of 20 (10 × 2), resulting in 110 instead of 120.

**Qwen 3 8B (Reasoning)** Predicted Answer: N/A

<think> Okay, let's try to figure out how many miles Pancho walks in a week. ... The first sentence says "পাঞ্চো একদিনে 20 মাইল হাঁটেন।" So that translates to "Pancho walks 20 miles a day." But then it says "প্রতিদিন হাঁটার জন্য তিনি ১.৫ ঘণ্টা সময় নেন।" Which means "He takes 1.5 hours per day for walking." ...
... Then it says: "সপ্তাহান্তে ব্যতিরেকে তিনি 10 মাইল হাঁটেন।" Which translates to "Except for weekends, he walks 10 miles." Wait, that's confusing. ... The phrase "সপ্তাহান্তে ব্যতিরেকে" means "except for weekends" or "excluding weekends." ... So that would mean that on weekdays, he walks 10 miles, and on weekends, he walks something else? But the first sentence said he walks 20 miles a day. So there's a contradiction here. ...
... Wait, maybe the problem is structured as follows: On weekdays, he walks 20 miles per day, taking 1.5 hours each day. On weekends, he walks 10 miles, taking 45 minutes. But then the third sentence says "সপ্তাহান্তে ব্যতিরেকে তিনি 10 মাইল হাঁটেন।" Which is "Except for weekends, he walks 10 miles." So that would mean that on weekdays, he walks 10 miles, and on weekends, he walks something else. But the first sentence says he walks 20 miles a day. So this is conflicting. ...
... I think the confusion comes from the translation of the third sentence. Let me try to parse it again: "সপ্তাহান্তে ব্যতিরেকে তিনি 10 মাইল হাঁটেন।" ... So the sentence is saying "Except for weekends, he walks 10 miles." So that would mean that on weekdays, he walks 10 miles, and on weekends, he walks something else. But the first sentence says he walks 20 miles a day. So this is conflicting. ...
... I think I need to proceed with the assumption that the first sentence is saying that on weekdays, he walks 20 miles each day, and on weekends, he walks 10 miles each day. Then, the third sentence says "except for weekends, he walks 10 miles," which would mean that on weekdays, he walks 10 miles, and on ... </think>

**Error Analysis:** Gets extremely confused by the problem structure and distractor information (time), unable to parse the question correctly and loops in contradictory reasoning without reaching an answer even after generating 7168 tokens.

**†DAGGER (Ours)** Predicted Answer: 120


```
{
"nodes": [
{"id": "weekday_miles", "op": "const", "val": 20,
"distractor": false,
"label": "প্রতিদিন কর্মদিবসে হাঁটা (মাইল)"},
{"id": "weekday_time", "op": "const", "val": 1.5,
"distractor": true,
"label": "প্রতিদিন কর্মদিবসে হাঁটার সময় (ঘণ্টা)"},
{"id": "weekend_miles", "op": "const", "val": 10,
"distractor": false,
"label": "প্রতিদিন সপ্তাহান্তে হাঁটা (মাইল)"},
{"id": "weekend_time", "op": "const", "val": 0.75,
"distractor": true,
"label": "সপ্তাহান্তে হাঁটার সময় (ঘণ্টা, ৪৫ মিনিট)"},
{"id": "weekdays", "op": "const", "val": 5,
"distractor": false,
"label": "কর্মদিবস সংখ্যা"},
{"id": "weekend_days", "op": "const", "val": 2,
"distractor": false,
"label": "সপ্তাহান্তের দিন সংখ্যা"},
{"id": "total_weekday_miles", "op": "mul",
"args": ["weekday_miles", "weekdays"], "distractor":
false,
"label": "কর্মদিবসে মোট হাঁটা (মাইল)"},
{"id": "total_weekend_miles", "op": "mul",
"args": ["weekend_miles", "weekend_days"],
"distractor": false,
"label": "সপ্তাহান্তে মোট হাঁটা (মাইল)"},
{"id": "total_miles", "op": "add",
"args": ["total_weekday_miles",
"total_weekend_miles"],
"distractor": false, "label": "এক সপ্তাহে মোট হাঁটা (মাইল)"},
{"id": "final_result", "op": "identity", "args":
["total_miles"],
"distractor": false, "label": "চূড়ান্ত উত্তর"}
```

```
]
}
```

**Success:** Correctly identifies time-related information as distractors (1.5 hours, 45 minutes) and computes: (20 × 5) + (10 × 2) = 100 + 20 = 120.

### A.9.3 Example 3

**Dataset:** MGSM-BN

**Augmentation Type:** Null-Effect Event Distractors (NEED)

**Question:**
লয়েডের একটি এগ ফার্ম আছে। একদিন লয়েড ভেবেছিলেন ডিমের দাম বাড়াবেন, কিন্তু পরে আর বাড়ালেন না। প্রতিদিন তাঁর মুরগিরা 252টি করে ডিম দেয় ও তিনি সেগুলি প্রতি ডজন 2 ডলার দরে বিক্রয় করেন। এক সপ্তাহে তাঁর এক বন্ধু ৩ ডজন ডিম কিনতে চেয়েছিলেন, কিন্তু শেষে আর নেননি। তিনি প্রতি সপ্তাহে ডিম থেকে কত উপার্জন করেন? যদি লয়েড সপ্তাহে অতিরিক্ত ৫০টি ডিম পেতেন, তাহলে উপার্জন আরও বেশি হতো, কিন্তু তা হয়নি।

*Lloyd has an egg farm. One day Lloyd thought about raising egg prices, but later didn't. Every day his hens lay 252 eggs and he sells them at $2 per dozen. In a week, his friend wanted to buy 3 dozen eggs but didn't buy them in the end. How much does he earn per week from eggs? If Lloyd had received an extra 50 eggs per week, his earnings would have been higher, but that didn't happen.*

**Ground Truth Answer: 294**

**Gemma 3 12B (CoT)** Predicted Answer: 288

প্রথমে, আমাদের প্রতিদিনের ডিম বিক্রির সংখ্যা বের করতে হবে। যেহেতু তিনি প্রতিদিন 252টি ডিম পাড়েন এবং প্রতি ডজন ডিমের দাম 2, তাই প্রথমে ডিমের সংখ্যাকে ডজনে পরিবর্তন করতে হবে। 252টি ডিম / 12 ডিম/ডজন = 21 ডজন ডিম। এরপর, প্রতিদিনের আয় হবে 21 ডজন * 2/ডজন = 42। সুতরাং, প্রতিদিনের আয় 42। এরপর, প্রতি সপ্তাহে আয় হবে 42/দিন * 7 দিন/সপ্তাহ = 294। তার বন্ধু ৩ ডজন ডিম কিনতে চেয়েছিল, কিন্তু কেনেনি। তাই ৩ ডজন ডিমের দাম বাদ দিতে হবে। ৩ ডজন ডিম * 2/ডজন = 6। সুতরাং, প্রতি সপ্তাহে লয়েডের আয় হবে 294 - 6 = 288। উত্তর হল 288।

**Error Analysis:** Correctly calculates weekly earnings as $294 but then incorrectly subtracts $6 for the friend's eggs that were never purchased, resulting in $288.

**Qwen 3 8B (Reasoning)** Predicted Answer: 294

<think> Okay, let's try to figure out how much money Lloyd earns from selling eggs each week. ... The problem says he has a farm with hens. Each day, his hens lay 252 eggs. He sells them at $2 per dozen. ...
... So in a week, which is 7 days, the total number of eggs would be 252 multiplied by 7. ... 252 times 7 is 1764 eggs per week. ... Now, since he sells them by the dozen, I need to convert the total number of eggs into dozens. One dozen is 12 eggs. So 1764 divided by 12. ... That's 147 dozens. ...
... Then, he sells each dozen for $2. So 147 dozens times $2 per dozen. ... So his weekly earnings would be $294. ... The friend didn't end up buying them, it doesn't affect the total earnings because he still sells all the eggs he has. ... So the answer should be $294. </think>

প্রশ্ন বিশ্লেষণ:
লয়েডের আয়ের হিসাব করার জন্য আমাদের মোট ডিমের সংখ্যা বের করে সেটিকে ডজনে রূপান্তর করতে হবে এবং তারপর মোট মূল্য নির্ধারণ করতে হবে। প্রশ্নে কিছু বিভ্রান্তিকর তথ্য আছে, যেমন: বন্ধুর ডিম না কেনা বা কাল্পনিক অতিরিক্ত ৫০টি ডিম, যা মূল হিসাবে প্রভাব ফেলবে না।
ধাপ দ্বারা ধাপ সমাধান:
১. মোট ডিমের সংখ্যা নির্ণয়: লয়েডের মুরগিরা প্রতিদিন ২৫২টি ডিম দেয়। এক সপ্তাহে (৭ দিন) মোট ডিমের সংখ্যা: $252 \times 7 = 1764$ টি ডিম।
২. ডজনে রূপান্তর: আমরা জানি, ১ ডজন = ১২টি ডিম। সুতরাং, ১৭৬৪টি ডিমের ডজন সংখ্যা: $\frac{1764}{12} = 147$ ডজন।
৩. মোট উপার্জন নির্ণয়: প্রতি ডজন ডিমের বিক্রয়মূল্য 2। সুতরাং, এক সপ্তাহে মোট উপার্জন: $147 \times 2 = 294$ ডলার।

উত্তর: 294।

**Success:** Correctly identifies that the friend's unpurchased eggs and hypothetical extra eggs are distractors that don't affect the actual earnings calculation.

## †DAGGER (Ours) Predicted Answer: 294


```
{
 "nodes": [
{"id": "eggs_per_day", "op": "const", "val": 252,
 "distractor": false,
 "label": "প্রতিদিন ডিম উৎপাদন"},
{"id": "days_in_week", "op": "const", "val": 7,
 "distractor": false,
 "label": "এক সপ্তাহে দিন"},
{"id": "total_eggs_week", "op": "mul",
 "args": ["eggs_per_day", "days_in_week"],
 "distractor": false,
 "label": "সপ্তাহে মোট ডিম"},
{"id": "dozen", "op": "const", "val": 12,
 "distractor": false,
 "label": "এক ডজন ডিম"},
{"id": "price_per_dozen", "op": "const", "val": 2,
 "distractor": false,
 "label": "প্রতি ডজন ডিমের দাম"},
{"id": "dozens_sold", "op": "div", "args":
 ["total_eggs_week", "dozen"],
 "distractor": false, "label": "মোট ডজন ডিম"},
{"id": "total_earning", "op": "mul",
 "args": ["dozens_sold", "price_per_dozen"],
 "distractor": false,
 "label": "সপ্তাহে মোট উপার্জন"},
{"id": "friend_eggs", "op": "const", "val": 3,
 "distractor": true,
 "label": "বন্ধুর ডিমের চাহিদা (ডজন)"},
{"id": "extra_eggs", "op": "const", "val": 50,
 "distractor": true,
 "label": "অতিরিক্ত ডিম (কিন্তু পায়নি)"},
{"id": "final_result", "op": "identity", "args":
 ["total_earning"],
 "distractor": false, "label": "চূড়ান্ত উত্তর"}
 ]
 }
```


**Success:** Correctly identifies negation-based distractors (friend's unpurchased 3 dozen eggs and hypothetical extra 50 eggs) and computes: $(252 \times 7)/12 \times 2 = 147 \times 2 = 294$.

## A.10 Prompts

### A.10.1 Prompt for Generating Related Entity Distractor

**Related Entity Distractor Generation**

**Your task:** Add 1-3 new **distractor sentences** to a Bengali math problem. These sentences add information about the item whose quantity is being asked for or related entities or intermediate facts, but DO NOT change the final answer.

**RULES:**

1. You may mention the same type of item being asked about, but only when referring to other people, places, or unrelated situations - you must NOT change or add facts about the main subject's quantity.
2. If you include numbers, they must only represent intermediate or unrelated quantities and must not affect the final calculation.
3. NEVER add a number that equals the final answer
4. Each new sentence must be standalone and insertable
5. Specify exactly WHERE to insert (after which sentence number, counting from 1)

**OUTPUT FORMAT:**
Return a JSON object with a "sentences" key containing a list:

```
{
"sentences": [
{"sentence": "Bengali sentence here",
"insert_after": 1},
{"sentence": "Another Bengali
sentence", "insert_after": 2}
]
}
```

**EXAMPLE 1:**
*Input:*
**Original:** রহিমের ৫টি কলম আছে এবং ৩ টি কলম উপহার পেল। সে আরও ৮টি কলম কিনল। রহিমের মোট কত কলম?
**Answer:** 16
*Output:*

```
{
"sentences": [
{"sentence": "তার বন্ধু করিমের ১৩টি কলম
আছে।", "insert_after": 1},
{"sentence": "করিমের স্কুলে ৮টি কলম রাখা ছিল।",
"insert_after": 2},
{"sentence": "দোকানে আরও ২০টি কলম ছিল।",
"insert_after": 3}
]
}
```

**EXAMPLE 2:**
*Input:*
**Original:** মিনার ৪টি আপেল ছিল। সে বাজারে গেল এবং ৭টি আপেল কিনল। তারপর ৩টি আপেল খেল। মিনার কাছে কতটি আপেল বাকি?
**Answer:** 8
*Output:*

```
{
"sentences": [
{"sentence": "বাজারে মোট ১১টি আপেল বিক্রি
হয়েছিল সেদিন।", "insert_after": 2},
{"sentence": "মিনার ভাই রাজুর কাছে ৫৬৭ টি
আপেল ছিল।", "insert_after": 3},
{"sentence": "রাজু বাজার থেকে আরো ১০ টি
আপেল কিনলো।", "insert_after": 4}
]
}
```

**EXAMPLE 3:**
*Input:*
**Original:** একটি গাছে ১৫টি পাখি বসেছিল। ৮টি পাখি উড়ে গেল। কতটি পাখি বাকি রইল?
**Answer:** 7
*Output:*

```
{
"sentences": [
{"sentence": "পাশের গাছে ২৩টি পাখি ছিল।",
"insert_after": 1},
{"sentence": "আকাশে উড়ন্ত পাখির মধ্যে ৮টি পাখি
উত্তর দিকে গেল।", "insert_after": 2}
]
}
```

**NOW GENERATE:**
**Original:** {question}
**Answer:** {answer}
**Output (JSON only):**

### A.10.2 Prompt for Generating Orthogonal Attribute Distractor

**Orthogonal Attribute Distractor Generation**

**Your task:** Add 1-3 **distractor sentences** that give additional properties (price, weight, time, distance, etc.) to objects already in the problem. These properties are in DIFFERENT units than what the question asks and DO NOT change the final answer.

**RULES:**

1. Add properties like: price (টাকা), weight (কেজি/গ্রাম), time (ঘণ্টা/মিনিট), distance (কিমি/মিটার), etc.
2. The question must ask about COUNT or a DIFFERENT property than what you add
3. Properties must be about the SAME objects mentioned in the problem
4. You can use phrases like: "যার দাম ক", "যার ওজন খ", "যা গ সময় লাগে"

**OUTPUT FORMAT:**
Return a JSON object with a "sentences" key containing a list:

```
{
"sentences": [
{"sentence": "Bengali sentence here",
"insert_after": 1},
{"sentence": "Another Bengali
sentence", "insert_after": 2}
]
}
```

**EXAMPLE 1:**
*Input:*
**Original:** রহিমের ৬টি কলম আছে। সে আরও ৪টি কলম কিনল। রহিমের মোট কত কলম?
**Answer:** 10
*Output:*

```
{
"sentences": [
{"sentence": "প্রতিটি কলমের দাম ১৫ টাকা।",
"insert_after": 1},
{"sentence": "৬ টি কলম দিয়ে ১২০০ পৃষ্ঠা লিখা
যায়।", "insert_after": 2}
]
}
```

**EXAMPLE 2:**
*Input:*
**Original:** সেলিমের ৮টি বই ছিল। সে আরও ৫টি বই কিনল। সেলিমের মোট কতটি বই?
**Answer:** 13
*Output:*

```
{
"sentences": [
{"sentence": "প্রথম ৮টি বইয়ের মোট ওজন ৩
কেজি।", "insert_after": 1},
{"sentence": "পরের ৫টি বইয়ের ওজন ১০ কেজি।",
"insert_after": 2}
]
}
```

**EXAMPLE 3:**
*Input:*
**Original:** প্রথম দিন রিয়া ১০ কিলোমিটার দৌড়াল। দ্বিতীয় দিন ১৫ কিলোমিটার দৌড়াল। সে মোট কত কিলোমিটার দৌড়ালো?
**Answer:** 25
*Output:*

```
{
"sentences": [
{"sentence": "প্রথম দিনের দৌড়ে ২ ঘণ্টা সময়
লেগেছিল।", "insert_after": 2},
{"sentence": "দ্বিতীয় দিনের দৌড়ে ৫ ঘণ্টা সময়
লেগেছিল।", "insert_after": 3}
]
}
```

**NOW GENERATE:**
**Original:** {question}
**Answer:** {answer}
**Output (JSON only):**

### A.10.3 Prompt for Generating Null-Effect Event Distractor

**Null-Effect Event Distractor Generation**

**Your task:** Add 1–3 **distractor sentences** about actions that were planned but not completed, started but later canceled (net zero effect), or hypothetical situations that are clearly marked as NOT REAL. These sentences MUST NOT change the final answer.

**RULES:**

1. Actions must involve the MAIN person/entity in the problem and may involve another related person/entity.
2. At least one added sentence must use negation words: "কিন্তু না", "কিন্তু দিলনা", "রাজি হলো কিন্তু", "ফিরিয়ে দিল"
   OR hypothetical markers: "যদি...হতো", "তাহলে...হতো", "কিন্তু ...".
3. The actions must have NET ZERO effect on the final quantity (cancelled, returned, reversed, or never actually happened).
4. Make it clear that these actions do not change the final answer.

**OUTPUT FORMAT:**
Return a JSON object with a "sentences" key containing a list:

```
{
"sentences": [
{"sentence": "Bengali sentence here",
"insert_after": 1},
{"sentence": "Another Bengali
sentence", "insert_after": 2}
]
}
```

**EXAMPLE 1:**
*Input:*
**Original:** মিনার ১০০টি কলম আছে। প্রতিটি ৫ টাকা দামের। মিনা যদি সব কলম বিক্রি করে, কত টাকা পাবে?
**Answer:** 500
*Output:*

```
{
"sentences": [
{"sentence": "মিনার ভাই রাজুর কাছে ২০টি কলম
আছে।", "insert_after": 1},
{"sentence": "মিনা রাজুর কাছে ১৫টি কলম
চাইল।", "insert_after": 2},
{"sentence": "রাজু ১০টি দিতে রাজি হলো, কিন্তু
পরে আর দিলনা।", "insert_after": 3}
]
}
```

**EXAMPLE 2:**
*Input:*
**Original:** নাফিসার ২০টি চকলেট ছিল। সে ৮টি চকলেট খেল। কতটি চকলেট বাকি?
**Answer:** 12
*Output:*

```
{
"sentences": [
{"sentence": "সে বন্ধুকে ৫টি চকলেট দিয়েছিল।",
"insert_after": 1},
{"sentence": "কিন্তু তার বন্ধু পরে সবকটি চকলেট
তাকে ফিরিয়ে দিল।", "insert_after": 2}
]
}
```

**EXAMPLE 3:**
*Input:*
**Original:** করিমের ১২টি আম আছে। সে ৫টি আম কিনল। করিমের মোট কতটি আম?
**Answer:** 17
*Output:*

```
{
"sentences": [
{"sentence": "যদি বড় বাজার খোলা থাকত, তাহলে
করিম ২০টি আম কিনতে পারত।", "insert_after":
1},
{"sentence": "কিন্তু বাজার বন্ধ ছিল, তাই সে ২০টি
আম কিনতে পারেনি।", "insert_after": 2}
]
}
```

**NOW GENERATE:**
**Original:** {question}
**Answer:** {answer}
**Output (JSON only):**

### A.10.4 Prompt for Verifying Modified Problems

**MWP Distractor Verification**

You are a math problem verifier. Your task is to check whether the modified Bengali math problem preserves the same final answer as the original.

**INSTRUCTIONS:**

1. Read the original problem.
2. Read the modified problem.
3. Check whether the modified problem changes the numbers or operations affecting the final answer.
4. Set "answers_match" to true only if the modified problem should have the same numerical answer as the original (and therefore match the ground truth).
5. Provide a brief justification.

**ORIGINAL PROBLEM:**
{original}

**MODIFIED PROBLEM:**
{modified}

**GROUND TRUTH ANSWER:**
{ground_truth}

**OUTPUT FORMAT (JSON only):**

```
{
"justification": "brief explanation",
"answers_match": true/false
}
```

**Output (JSON only):**

### A.10.5 Prompt for Generating Computational Graph

**Computational Graph Generation**

You are an expert 'target language' Math Reasoner. Your task is to solve mathematical problems by constructing a "Computational Graph".

**Graph Rules:**

- `id`: Unique identifier (e.g., "n1", "n2").
- `val`: The raw number extracted from text (for input nodes).
- `op`: The operation (`add`, `sub`, `mul`, `div`, `round`, `sqrt`, `floor`, `sum`, `mean`). Use `const` for input numbers.
- `args`: List of input node IDs.
- `distractor`: Boolean (`true` / `false`). Set to `true` if the node is NOT used in the final calculation path.
- `label`: Label for the node.

**Available Operations:**

- **Input:** `const` (Use this for all numbers found in text or constants).
- **Arithmetic:** `add`, `sub`, `mul`, `div`, `abs` (absolute difference).
- **Logic/Stats:** `sum`, `mean`, `min` (minimum), `max` (maximum).
- **Rounding:** `round` (nearest int), `floor` (round down), `ceil` (round up).
- **Advanced:** `sqrt`, `pow`, `mod` (remainder), `gcd`, `lcm`.
- **Output:** `identity` ("final_result" points to the answer node)

Only output a JSON graph representing the solution, nothing else. Nodes must be topologically sorted, and there must be exactly one "final_result" node that represents the final answer. One example is provided below.

**Example:**
*Question:*
মিনার কাছে ১২২১৯৫ টা কলম আছে। রাজুর কাছে ২৫০৮৪ টা কলম আছে। মিনা রাজুর কাছে ১১২৬ টি কলম চাইল। রাজু ১০০০ টি কলম দিতে রাজি হল, কিন্তু পরে আর দিলেনা। প্রতিটি কলমের

দাম ৪৫.৬ টাকা। মিনা যদি কলমগুলো বিক্রি করতে চায়, সে কত টাকা পাবে?
*Output:*

```
{
 "nodes": [
 {"id": "n1", "op": "const", "val": 122195,
 "distractor": false,
 "label": "মিনার কলম"},
 {"id": "n2", "op": "const", "val": 25084,
 "distractor": true,
 "label": "রাজুর কলম"},
 {"id": "n3", "op": "const", "val": 1126,
 "distractor": true,
 "label": "মিনা রাজুর কাছে চাইল"},
 {"id": "n4", "op": "const", "val": 1000,
 "distractor": true,
 "label": "রাজু দিতে রাজি হল"},
 {"id": "n5", "op": "const", "val": 45.6,
 "distractor": false,
 "label": "প্রতিটি কলমের দাম"},
 {"id": "total_money", "op": "mul", "args": ["n1",
 "n5"],
 "distractor": false, "label": "মিনার মোট টাকা"},
 {"id": "final_result", "op": "identity", "args":
 ["total_money"],
 "distractor": false, "label": "চূড়ান্ত উত্তর"}
 ]
 }
```

**Your Task:**
*Question:*
{question}
*Output:*

### A.10.6 Prompt for Reasoning Models

**Reasoning Model Prompt**

```
Please reason step by step in Bangla,
and put your final numeric answer
within \boxed{}.

প্রশ্ন: {question}
```

### A.10.7 Prompt for Chain of Thought (5-Shot)

**Chain of Thought Prompt**

প্রশ্ন: রজারের 5টি টেনিস বল আছে। সে আরও 2 ক্যান টেনিস বল কিনেছে। প্রতিটি ক্যানে 3টি করে টেনিস বল আছে। তার কাছে এখন কতগুলি টেনিস বল আছে?
ধাপে ধাপে উত্তর: রজারের প্রথমে 5টি বল ছিল। 2টি ক্যানের প্রতিটিতে 3টে টেনিস বল মানে 6টি টেনিস বল। 5 + 6 = 11। উত্তর হল 11।

প্রশ্ন: সার্ভার কক্ষে নয়টি কম্পিউটার ছিল। সোমবার থেকে বৃহস্পতিবার প্রতিদিন আরও পাঁচটি করে কম্পিউটার স্থাপন করা হয়েছিল। সার্ভার কক্ষে এখন কতগুলি কম্পিউটার আছে?
ধাপে ধাপে উত্তর: সোমবার থেকে বৃহস্পতিবার 4দিন হয়। প্রতিদিন 5টি করে কম্পিউটার যোগ করা হয়েছে। যার অর্থ মোট 4 * 5 = 20টি কম্পিউটার যোগ করা হয়েছে। শুরুতে 9টি কম্পিউটার ছিল, তাই এখন 9 + 20 = 29টি কম্পিউটার রয়েছে। উত্তর হল 29।

প্রশ্ন: লিয়ার 32টি চকোলেট ছিল এবং তার বোনের ছিল 42টি। যদি তারা 35টি খেয়ে থাকে, তাহলে তাদের কাছে মোট কতগুলি অবিশিষ্ট আছে?
ধাপে ধাপে উত্তর: লিয়ার 32টি চকোলেট ছিল এবং লিয়ার বোনের ছিল 42টি। যার অর্থ শুরুতে 32 + 42 = 74টি চকোলেট ছিল। 35টি খাওয়া হয়ে গেছে। তাই তাদের কাছে মোট 74 - 35 = 39টি চকোলট আছে। উত্তর হল 39।

প্রশ্ন: শনের পাঁচটি খেলনা আছে। ক্রিসমাস উপলক্ষে সে তার মাতা ও পিতা উভয়ের থেকে দুটি করে খেলনা পেয়েছে। তার কাছে এখন কতগুলি খেলনা আছে?
ধাপে ধাপে উত্তর: তার কাছে 5টি খেলনা আছে। সে তার মাতার থেকে 2টি খেলনা পেয়েছিল অতএব, এরপর তার 5 + 2 = 7টি খেলনা হয়েছে। তারপর সে তার পিতার থেকে 2টি খেলনা পেয়েছিল, তাই তার মোট 7 + 2 = 9 টি খেলনা হয়েছে। উত্তর হল 9।

প্রশ্ন: মাইকেলের 58টি গলফ বল ছিল। মঙ্গলবার, সে 23টি গলফ বল হারিয়েছিল। বুধবার, সে আরও 2টি বল হারিয়েছিল। বুধবারের শেষে তার কাছে কয়টি গলফ বল ছিল?
ধাপে ধাপে উত্তর: শুরুতে মাইকেলের কাছে 58টি গলফ বল ছিল এবং সে 23টি বল হারিয়েছিল, তাই তার 58 - 23 = 35টি বল আছে। আরও 2টি বল হারানোর পর, তার এখন 35 - 2 = 33টি বল আছে। উত্তর হল 33।

প্রশ্ন: {question}
ধাপে ধাপে উত্তর:

## A.11 Examples of Augmented Questions from DistractMath-BN

| Dataset & Type | Original | Augmented |
|---|---|---|
| **MSVAMP Dataset** | | |
| RED | জুলিয়া সোমবার ১৮ জন ছেলেমেয়ের সঙ্গে ট্যাগ খেলেন। মঙ্গলবার তিনি ১০ জন ছেলেমেয়ের সঙ্গে ট্যাগ খেলেন। সোমবার তিনি কতজন ছেলেমেয়ের সঙ্গে আরও খেলেন তার সংখ্যা কত?<br>*(Julia played tag with 18 children on Monday. On Tuesday she played with 10 children. How many more children did she play with on Monday?)* | জুলিয়া সোমবার ১৮ জন ছেলেমেয়ের সঙ্গে ট্যাগ খেলেন। মঙ্গলবার তিনি ১০ জন ছেলেমেয়ের সঙ্গে ট্যাগ খেলেন। মঙ্গলবার মাঠে আরও ১৫ জন ছেলেমেয়ে খেলতে এসেছিল। তার ছোট বোন বুধবার ১২ জন ছেলেমেয়ের সঙ্গে লুকোচুরি খেলেছিল। সোমবার তিনি কতজন ছেলেমেয়ের সঙ্গে আরও খেলেন তার সংখ্যা কত?<br>*(Julia played tag with 18 children on Monday. On Tuesday she played with 10 children. 15 more children came to play on Tuesday. Her younger sister played hide-and-seek with 12 children on Wednesday. How many more children did she play with on Monday?)* |
| OAD | জুলিয়া সোমবার ১৮ জন ছেলেমেয়ের সঙ্গে ট্যাগ খেলেন। মঙ্গলবার তিনি ১০ জন ছেলেমেয়ের সঙ্গে ট্যাগ খেলেন। সোমবার তিনি কতজন ছেলেমেয়ের সঙ্গে আরও খেলেন তার সংখ্যা কত?<br>*(Julia played tag with 18 children on Monday. On Tuesday she played with 10 children. How many more children did she play with on Monday?)* | জুলিয়া সোমবার ১৮ জন ছেলেমেয়ের সঙ্গে ট্যাগ খেলেন। সোমবার খেলতে ১ ঘণ্টা সময় লেগেছিল। মঙ্গলবার তিনি ১০ জন ছেলেমেয়ের সঙ্গে ট্যাগ খেলেন। মঙ্গলবার খেলতে ৪৫ মিনিট সময় লেগেছিল। সোমবার তিনি কতজন ছেলেমেয়ের সঙ্গে আরও খেলেন তার সংখ্যা কত?<br>*(Julia played tag with 18 children on Monday. It took 1 hour to play on Monday. On Tuesday she played with 10 children. It took 45 minutes to play on Tuesday. How many more children did she play with on Monday?)* |
| NEED | জুলিয়া সোমবার ১৮ জন ছেলেমেয়ের সঙ্গে ট্যাগ খেলেন। মঙ্গলবার তিনি ১০ জন ছেলেমেয়ের সঙ্গে ট্যাগ খেলেন। সোমবার তিনি কতজন ছেলেমেয়ের সঙ্গে আরও খেলেন তার সংখ্যা কত?<br>*(Julia played tag with 18 children on Monday. On Tuesday she played with 10 children. How many more children did she play with on Monday?)* | জুলিয়া সোমবার ১৮ জন ছেলেমেয়ের সঙ্গে ট্যাগ খেলেন। জুলিয়া ভেবেছিলেন আরও ৫ জন ছেলেমেয়েকে সোমবার খেলায় ডাকবেন, কিন্তু পরে আর ডাকলেন না। মঙ্গলবার তিনি ১০ জন ছেলেমেয়ের সঙ্গে ট্যাগ খেলেন। যদি সোমবার বৃষ্টি না হতো, তাহলে আরও ৩ জন যোগ দিত, কিন্তু শেষ পর্যন্ত তারা আসেনি। সোমবার তিনি কতজন ছেলেমেয়ের সঙ্গে আরও খেলেন তার সংখ্যা কত?<br>*(Julia played tag with 18 children on Monday. Julia thought about inviting 5 more children to the game on Monday, but she didn't call them later. On Tuesday she played with 10 children. If it hadn't rained on Monday, 3 more would have joined, but they didn't come in the end. How many more children did she play with on Monday?)* |
| **MGSM Dataset** | | |
| RED | সোমবারে স্যাডি 8 ঘণ্টা ঘুমিয়েছিলেন। পরবর্তী দুইদিনের প্রত্যেকদিনে, তিনি 2 ঘণ্টা কম ঘুমিয়েছিলেন কারণ তাঁকে কিছু অ্যাসাইনমেন্ট শেষ করতে হত। যদি তিনি ঐ দুইদিনের তুলনায় বাকি সপ্তাহে 1 ঘণ্টা বেশি ঘুমিয়ে থাকেন, তবে সারা সপ্তাহ তিনি মোট কত ঘণ্টা ঘুমিয়েছিলেন?<br>*(Sadie slept 8 hours on Monday. On each of the next two days, she slept 2 hours less because she had to finish some assignments. If she slept 1 hour more than those two days for the rest of the week, how many hours did she sleep in total for the whole week?)* | সোমবারে স্যাডি 8 ঘণ্টা ঘুমিয়েছিলেন। স্যাডির ছোট বোন মঙ্গলবারে মাত্র ৬ ঘণ্টা ঘুমিয়েছিল। পরবর্তী দুইদিনের প্রত্যেকদিনে, তিনি 2 ঘণ্টা কম ঘুমিয়েছিলেন কারণ তাঁকে কিছু অ্যাসাইনমেন্ট শেষ করতে হত। বুধবার রাতে স্যাডি ঘুমানোর আগে একটি বই পড়েছিলেন। যদি তিনি ঐ দুইদিনের তুলনায় বাকি সপ্তাহে 1 ঘণ্টা বেশি ঘুমিয়ে থাকেন, তবে সারা সপ্তাহ তিনি মোট কত ঘণ্টা ঘুমিয়েছিলেন?<br>*(Sadie slept 8 hours on Monday. Sadie's younger sister slept only 6 hours on Tuesday. On each of the next two days, she slept 2 hours less because she had to finish some assignments. On Wednesday night Sadie read a book before sleeping. If she slept 1 hour more than those two days for the rest of the week, how many hours did she sleep in total for the whole week?)* |
| OAD | সোমবারে স্যাডি 8 ঘণ্টা ঘুমিয়েছিলেন। পরবর্তী দুইদিনের প্রত্যেকদিনে, তিনি 2 ঘণ্টা কম ঘুমিয়েছিলেন কারণ তাঁকে কিছু অ্যাসাইনমেন্ট শেষ করতে হত। যদি তিনি ঐ দুইদিনের তুলনায় বাকি সপ্তাহে 1 ঘণ্টা বেশি ঘুমিয়ে থাকেন, তবে সারা সপ্তাহ তিনি মোট কত ঘণ্টা ঘুমিয়েছিলেন?<br>*(Sadie slept 8 hours on Monday. On each of the next two days, she slept 2 hours less because she had to finish some assignments. If she slept 1 hour more than those two days for the rest of the week, how many hours did she sleep in total for the whole week?)* | সোমবারে স্যাডি 8 ঘণ্টা ঘুমিয়েছিলেন। সোমবারে স্যাডি ঘুমানোর পর ৫০০ ক্যালরি বার্ন করেছিলেন। পরবর্তী দুইদিনের প্রত্যেকদিনে, তিনি 2 ঘণ্টা কম ঘুমিয়েছিলেন কারণ তাঁকে কিছু অ্যাসাইনমেন্ট শেষ করতে হত। পরবর্তী দুইদিনে প্রতি রাতে ঘুমানোর জন্য তাঁর বিছানার চাদর প্রতিবার ২০ মিনিটে পরিষ্কার করা হয়। যদি তিনি ঐ দুইদিনের তুলনায় বাকি সপ্তাহে 1 ঘণ্টা বেশি ঘুমিয়ে থাকেন, তবে সারা সপ্তাহ তিনি মোট কত ঘণ্টা ঘুমিয়েছিলেন?<br>*(Sadie slept 8 hours on Monday. Sadie burned 500 calories after sleeping on Monday. On each of the next two days, she slept 2 hours less because she had to finish some assignments. On the next two days, her bed sheet is cleaned in 20 minutes each night for sleeping. If she slept 1 hour more than those two days for the rest of the week, how many hours did she sleep in total for the whole week?)* |
| NEED | সোমবারে স্যাডি 8 ঘণ্টা ঘুমিয়েছিলেন। পরবর্তী দুইদিনের প্রত্যেকদিনে, তিনি 2 ঘণ্টা কম ঘুমিয়েছিলেন কারণ তাঁকে কিছু অ্যাসাইনমেন্ট শেষ করতে হত। যদি তিনি ঐ দুইদিনের তুলনায় বাকি সপ্তাহে 1 ঘণ্টা বেশি ঘুমিয়ে থাকেন, তবে সারা সপ্তাহ তিনি মোট কত ঘণ্টা ঘুমিয়েছিলেন?<br>*(Sadie slept 8 hours on Monday. On each of the next two days, she slept 2 hours less because she had to finish some assignments. If she slept 1 hour more than those two days for the rest of the week, how many hours did she sleep in total for the whole week?)* | সোমবারে স্যাডি 8 ঘণ্টা ঘুমিয়েছিলেন। স্যাডি প্রথমে ঠিক করেছিলেন যে মঙ্গলবার ৮ ঘণ্টা ঘুমাবেন, কিন্তু না, পরে সিদ্ধান্ত বদলান এবং ৬ ঘণ্টা ঘুমান। পরবর্তী দুইদিনের প্রত্যেকদিনে, তিনি 2 ঘণ্টা কম ঘুমিয়েছিলেন কারণ তাঁকে কিছু অ্যাসাইনমেন্ট শেষ করতে হত। বুধবার তিনি ভাবছিলেন আরও কম ঘুমাবেন, কিন্তু শেষ পর্যন্ত আগের দিনের মতোই ৬ ঘণ্টা ঘুমান। যদি তিনি ঐ দুইদিনের তুলনায় বাকি সপ্তাহে 1 ঘণ্টা বেশি ঘুমিয়ে থাকেন, তবে সারা সপ্তাহ তিনি মোট কত ঘণ্টা ঘুমিয়েছিলেন?<br>*(Sadie slept 8 hours on Monday. Sadie first decided to sleep 8 hours on Tuesday, but no, she changed her mind and slept 6 hours. On each of the next two days, she slept 2 hours less because she had to finish some assignments. On Wednesday she thought she would sleep even less, but eventually slept 6 hours like the previous day. If she slept 1 hour more than those two days for the rest of the week, how many hours did she sleep in total for the whole week?)* |

Table 12: Examples of original and augmented questions with different distractor types across MSVAMP and MGSM datasets. Distractor sentences are marked in red.