MACoT: Synthesizing Chains of Thought for Small Models via Multi-Agent Collaboration

Guokai Tang, Feng Zhao*

Natural Language Processing and Knowledge Graph Lab, School of Computer Science and Technology,
Huazhong University of Science and Technology, Wuhan, China
{guokaitang,zhaof}@hust.edu.cn

Abstract

Small language models (SLMs) run quickly, consume little memory, and can be deployed on edge devices, making them especially appealing when compute or energy is limited. Because of these advantages, boosting SLMs' reasoning ability has become an important research goal. A common approach is to distill the long chains of thought (long-CoTs) produced by large reasoning models (LRMs) into SLMs, hoping to transfer the larger models' strong reasoning ability. However, SLMs do not always benefit from distillation of long-CoTs. The lengthy and complex semantic steps and large amount of self-reflection content in long-CoTs may exceed the limited learning capabilities of SLMs, and the impact of self-reflection density on the performance of SLMs is unclear. To resolve this capacity mismatch, we propose **MACoT**, a multi-agent framework that synthesizes chains of thought (CoTs) that are more suitable for small models rather than compressing or pruning existing ones. Through the interactive collaboration among six types of agents, **MACoT** synthesizes semantically explicit, logically clear CoTs that efficiently activate a small model's internal knowledge through a carefully designed output pattern. At the same time, the CoTs synthesized by our method can retain a small amount of self-reflection content, thereby matching the learning capability of the small model and maximizing its reasoning accuracy. We fine-tuned Qwen2.5-7B-Instruct using only 1879 synthetic CoTs, significantly improving its performance on mathematical reasoning tasks and generalizing well, outperforming models trained on 5x more data. Through experiments, we found that a modest level of self-reflection boosts small-model performance, whereas excessive reflection sharply degrades it, which shows that "teaching SLMs to think" hinges on aligning each CoT's cognitive load with the model's capacity.

Introduction

Large reasoning models (LRMs) such as OpenAI-o3 (OpenAI 2025) and DeepSeek-R1 (DeepSeek-AI et al. 2025) have demonstrated strong reasoning capabilities in mathematical, legal, and logical reasoning tasks. They can expand the reasoning process to a deeper long-CoT through iterative exploration and self-reflection (Zhang et al. 2024) to improve the accuracy of reasoning. Although long-CoTs can substantially improve the accuracy of reasoning, they also cause a rapid increase in computational and resource costs as model parameters expand (Zhang et al. 2025) and the number of "deep thinking" layers during reasoning increases. Small language models ($\leq$ 7B) provide a natural remedy to this cost explosion. Because their parameter counts are dramatically lower, their deployment and inference costs amount to only a fraction of those of large reasoning models.

Question: I have a deck of 54 cards, and I deal all of the cards to x players, with each player getting y cards. If x is at least 2 and y is at least 5, then how many possible values of x are there?

Solution:
We are told that a deck of 54 cards is dealt evenly among x players, where each player gets y cards. The problem states that x≥2 and y≥5, and we are to find how many possible values x can take.
Since the total number of cards is 54, we know: x·y=54. This means x must divide 54 evenly, and for each such x, the corresponding y=\frac{54}{x} must be an integer.

But let me double-check. Is there any constraint I missed?

Actually, I realize the equation x·y=54 is fine, but maybe I'm interpreting the constraints too strictly. It only says each player gets y cards and that y≥5, but it doesn't explicitly say that the cards must be divided evenly in whole numbers.
So maybe we don't need x to divide 54 exactly. Perhaps fractional cards are allowed as long as the count y=\frac{54}{x} is at least 5. That opens up more possibilities.
Let's see — if y=\frac{54}{x} ≥5, then x≤frac{54}{5} = 10.8, so any integer x from 2 to 10 should be acceptable. That gives: x=2,3,4,5,6,7,8,9,10. So there are 9 possible values of x that satisfy both x≥2 and y≥5.
Thus, the answer is 9.

Figure 1: Small language Models fine-tuned with long-CoTs will go down the wrong reasoning trajectory after self-reflective sentences. The green region represents the correct reasoning trajectory; The blue sentence is self-reflection sentence; The red region represents the wrong reasoning trajectory.

To reduce the cost of inference, recent research has turned to long-CoT distillation, which migrates the long-context reasoning capabilities of large reasoning models to smaller models, enabling them to handle complex tasks more efficiently (Li et al. 2023). However, studies have shown that

*Corresponding author

SLMs do not always benefit from long-CoT distillation, and sometimes even lead to decreased reasoning performance (Luo et al. 2025), and that excessive overthinking steps or self-reflective content in long-CoTs such as "Wait, let me check the steps above..."—will exceed SLM's limited learning capacity and may degrade its reasoning performance (Wang et al. 2025). In other words, chains of thought (CoTs) that are not aligned with the model's capacity will lead SLMs into decreased reasoning accuracy, as shown in Figure 1. Regarding the above issues, recent works mostly adopt the idea of "compression-pruning-mixing", CoT compression methods such as TokenSkip and TokenCleaning (Xia et al. 2025; Pang et al. 2025) clean up redundant and unimportant tokens in CoTs, allowing the model to maintain strong reasoning performance in downstream tasks. Wang et al. (2025) used binary cutting method to prune unnecessary steps in the long-CoTs, enhancing the reasoning ability of SLMs. Mix-Distillation (Li et al. 2025) mixes long and short CoTs in a certain proportion, which significantly improves the reasoning performance of SLMs after fine-tuning. However, existing CoT compression and pruning strategies may delete some reasoning steps that are critical to small models when removing redundant steps, and the retained sentences still retain the complex wording and output patterns unique to large models, which will cause performance loss to the fine-tuned model. Secondly, these methods do not systematically explore the extent to which overthinking steps or self-reflection content should be pruned to best suit the capabilities of small models. At the same time, the CoT mixing strategy essentially only focuses on output preferences and does not fundamentally enhance the capability of small models to learn from long-CoTs reasoning.

In order to explore and solve the above issues, we designed a multi-agent framework named **MACoT**. **MACoT** does not simply compress or prune the existing long-CoTs, but directly synthesizes a new CoT. While **retaining the semantically explicit and logically clear reasoning steps**, it maximizes the activation of the internal knowledge of the small model through a **carefully designed output pattern** to improve the accuracy of reasoning. At the same time, we propose a **collaborative paradigm that dynamically adjusts the count of self-reflection** in CoTs to explore how much self-reflection can match the learning capacity of small models. Our framework consists of six types of agents: Solution Decomposer, Verifier, Problem Analyst, Reflection Coordinator, Student, and Teacher. First, the Solution Decomposer agent will regenerate a stepwise solution, then the Verifier agent checks the correctness of this solution. Next, Problem Analyst agents will generate the known conditions of the problem, unknown conditions, the field and sub-field of the problem, and possible useful problem-solving skills in related fields through debating. Then, the Reflection Coordinator assembles student–teacher teams and assigns them tasks, and Teacher–Student teams co-write the remaining solution steps.

The above process generates 1879 CoTs, which constitute our MACoT dataset. We fine-tuned Qwen2.5-7B-Instruct on our dataset through LoRA, significantly improving its performance on mathematical reasoning tasks and generalizing well, outperforming models trained on 5× more data. Meanwhile, token consumption is only 15% of long-CoT distillation. In the quantitative analysis, we dynamically adjust the number of self-reflective statements in the CoT to verify their impact on the performance of SLMs. The experiment shows that CoTs with a small amount of self-reflective content perform best. Our results show that "teaching SLMs to think" depends on the cognitive and ability alignment of CoTs, rather than on long-CoTs that exceed SLMs' capacity.

Our main contributions are as follows:

- We propose **MACoT**, a multi-agent framework that synthesizes semantically explicit and logically clear CoTs tailored to SLMs.
- We propose a **collaborative paradigm that dynamically adjusts the amount of self-reflection content** in CoTs. Through experiments, we found that a small, carefully controlled amount of self-reflection yields higher accuracy than either zero reflection or verbose long-reflection traces.
- We show that MACoT achieves higher reasoning performance while **token consumption is only 15%** of long-CoT distillation, demonstrating its practical value for resource-constrained deployment.

Related Works

Chain-of-Thought Distillation and Compression

Early successes with CoT prompting on billion-parameter LRMs spurred efforts to distill these stepwise rationales into lighter students. Shridhar et al. (2023) pioneered the use of GPT-3-generated trajectories for supervised fine-tuning of a student model and reported significant improvements on GSM8K and SVAMP. LIMO enabled a 32B model to achieve 0.57 accuracy on the challenging AIME24 benchmark using only 817 carefully designed CoTs (Ye et al. 2025). Niklas et al. (2025) used 1000 CoTs to achieve SOTA performance on multiple reasoning benchmarks through test-time scaling on a 32B model. However, the above methods lack empirical research on models with smaller parameters. In terms of CoTs compression, TokenSkip (Xia et al. 2025) selectively removed low-efficiency tokens in CoTs, shortening the CoT length by about 0.4 with less than 0.01 accuracy loss. Yuyang et al. (2025) proposed a method to derive the optimal length of CoTs based on model capability and task difficulty, showing that the number of reasoning steps in CoTs should match the model capability. Shangziqi et al. (2025) enhanced the reasoning ability of models with smaller parameter sizes by pruning inefficient reasoning steps in long-CoTs. In general, the above works show that simply using the long-CoTs for supervised fine-tuning of a small model may be counterproductive, and it is necessary to synthesize specialized CoTs for small models.

Data Synthesis and Multi-Agent Reasoning

Data synthesis is increasingly being used in model training, especially in large model alignment and instruction fine-tuning stages. Magpie (Xu et al. 2025) utilized the autoregressive properties of Llama-3-Instruct to synthesize data at

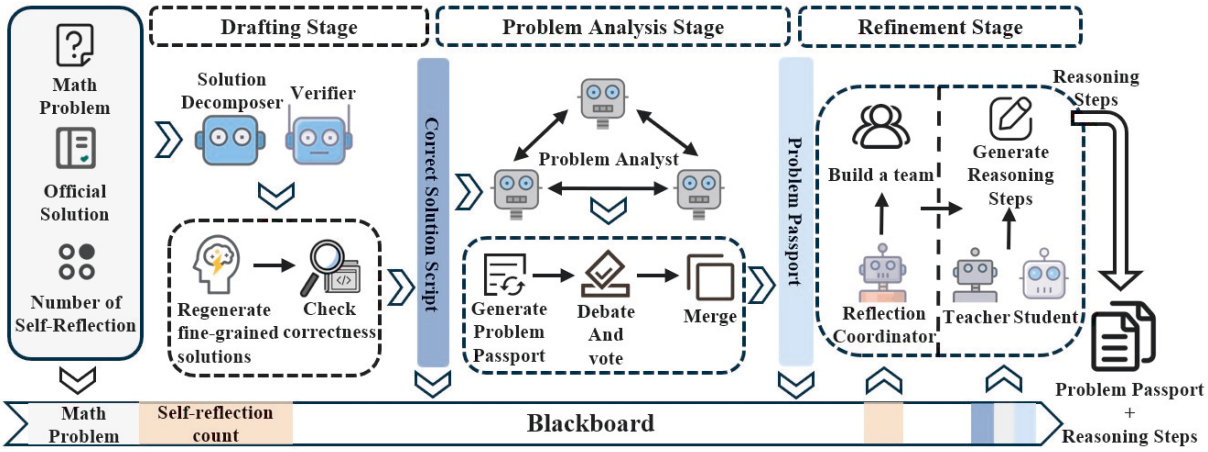

Figure 2: MACoT framework: The Solution Decomposer, Verifier, Problem-Analyst, Reflection Coordinator and Teacher agent are based on DeepSeek-v3, while the Student agent is based on Qwen2.5-7B-Instruct.

scale by prompting without relying on seed data. Genetic Instruct (Majumdar et al. 2025) synthesized coding instruction data by defining LLM as three different roles. GRA (Gao et al. 2025b) synthesized data with comparable performance to LLM distillation through collaboration among multiple small models. In terms of multi-agent reasoning, PlanGEN (Parmar et al. 2025) coordinated constraint, verification, and selection agents, dynamically choosing the best inference-time algorithm per instance and achieving new state-of-the-art results on OlympiadBench and DocFinQA. To rein in the high token cost of debate, GroupDebate (Liu et al. 2024) clustered discussants and shares intermediate conclusions, halving tokens while maintaining or even improving accuracy. Although recent data synthesis and multi-agent collaborative reasoning have achieved notable progress, there is still little work on multi-agent collaboration to synthesize chains of thought that align with the capabilities of small models.

Methodology

Multi-agent Workflow Framework Overview

To efficiently synthesize Chain-of-Thoughts (CoTs) tailored to small models, we present MACoT, a multi-agent LLM-based workflow illustrated in Figure 2. All agents communicate through a shared blackboard queue, where they post and consume intermediate states and task-specific context. Meanwhile, all agents are scheduled by a central orchestrator. The framework recasts CoT construction as a cooperative endeavor among six types of autonomous agents, each powered by its own LLM instance:

- **Solution Decomposer**: Rewrites each official solution into a fine-grained, multi-step script, delivering a higher-quality solution for subsequent agents.
- **Verifier**: Walks through each micro-step in the solution that Solution Decomposer rewrites, checking local validity and global coherence; writes a pass/fail flag to the blackboard.
- **Problem Analyst**: Extracts knowns, unknowns, domain tags, and key problem-solving skills from the math problem.
- **Reflection Coordinator**: According to the self-reflective count n in the user prompt, dynamically assembles Student–Teacher pairs for each step or step group, and assigns block sizes and specific tasks.
- **Student**: Generates one or more reasoning steps depending on the assigned task.
- **Teacher**: According to the assigned task, evaluates each Student's steps and generates self-reflective steps.

MACoT operates in three stages—Drafting, Problem Analysis, and Refinement. During drafting stage, the orchestrator posts the problem P and official solution S to the blackboard. The Solution Decomposer agent rewrites S into a fine-grained correct solution script and writes it back; the Verifier agent checks this script. If the Verifier flags errors in three consecutive drafts, the orchestrator drops the task; otherwise, the validated script is locked for the next stages. During Problem Analysis stage, three Problem Analyst agents read the math problem statement in parallel, each producing a JSON draft of givens, unknowns, domain tags, and problem-solving skills. After two debate rounds recorded on the blackboard, the orchestrator merges the votes into a single Problem Passport which is the first part of the final CoT, and posts it to the blackboard and broadcasts it downstream. During refinement stage, Reflection Coordinator defines tasks and forms student-teacher teams based on the number n of self-reflection defined in the user prompt. Each student and teacher agent performs the tasks assigned to them by Reflection Coordinator, students generate unverified reasoning steps, and teachers evaluate each Student's steps and generate self-reflective steps. The reasoning steps that student-teacher teams generate are the second part of the final CoT. When all tasks are completed, the orchestrator stitches together the problem passport and the output of the student-teacher in sequence to form a CoT that is suitable for small models.

MACoT delivers a two-layer CoT—Problem Passport plus refined reasoning—crafted through multi-agent collaboration. The problem passport primes the fine-tuned small model by activating the knowledge which is most relevant to the math problem inside the small model, while the Student–Teacher reflection loop supplies a semantically explicit, logically clear script that trims away the verbose self-reflection of long-CoTs. This synergy yields a logically complete Chain of Thought that matches the context budget and capability of small models. MACoT provides an output pattern for the small model, through which the small model can maximize its internal knowledge and enhance its reasoning performance in downstream tasks.

Collaborative Process

Drafting Stage. Solution Decomposer first reads the math problem P and the official solution S from the blackboard. Rather than copying S verbatim, the Solution Decomposer rewrites it into a fully ordered, fine-grained sequence of logical moves—each "micro-step" records the inference rule used, any intermediate result, and a clear statement of how it follows from the preceding step. This restructuring produces a self-contained script that is explicit enough to drive later verification and expansion.

Next, the Verifier walks through the script step-by-step. For every micro-step it checks (i) local correctness with respect to mathematical rules and (ii) global coherence with everything proven so far. If a flaw or discontinuity is found, the Solution Decomposer is asked to rewrite the solution; after three consecutive failures, the entire problem is abandoned. A script that passes all checks is locked and forwarded, unchanged, to the Problem-Analysis stage.

The purpose of the Drafting Stage is two-fold. First, it operates as a difficulty filter: by asking the Solution Decomposer agent to restate the official solution as a fully ordered sequence of micro-steps and requiring the Verifier's approval within three attempts, the framework automatically eliminates problems whose logical structure remains unstable after repeated rewriting—typically those that exceed the learning capacity of downstream small-scale models. Second, for problems that pass this gate, this stage functions as a foundation builder: it converts often terse or step-skipping official solutions into a rigorously validated, fine-grained script that records every inference rule and intermediate result. The resulting "locked" script not only guarantees correctness but also offers a clear, comprehensive scaffold for the subsequent Problem Analysis and Student-Teacher stages, where agents rely on explicit step references rather than opaque or incomplete textual explanations.

Problem Analysis Stage. When the correct solution script has been verified and locked, the orchestrator will launch three autonomous Problem Analyst agents under an "1 script-aware + 2 script-blind" setting:

- A_0: Script-Aware Analyst; Receives P and the correct solution script, and produces a high-confidence draft following the correct solution script.
- A_1, A_2: Script-Blind Analysts; Receive only P and, without any knowledge of the correct solution script, independently propose knowns, unknowns, domain labels, and skills—thereby introducing complementary perspectives.

Algorithm 1: Problem–Analysis Stage (1 Aware + 2 Blind)

Input: Problem statement P, correct solution script C
Output: Problem Passport $\mathcal{P}$

$A_0 \leftarrow$ Script-Aware Analyst(P, C)
$A_1, A_2 \leftarrow$ Script-Blind Analyst(P)
$\mathcal{P} \leftarrow \emptyset$
$D[A_0] \leftarrow \textsc{GenDraft}(A_0, P, C)$
for all $a \in \{A_1, A_2\}$ **in parallel do**
  $D[a] \leftarrow \textsc{GenDraft}(a, P)$
end for
for all $a \in \{A_0, A_1, A_2\}$ **do**
  $C[a, 1] \leftarrow \textsc{VoteAndScore}\big(a,\ D \setminus \{D[a]\}\big)$
end for
$\textsc{Disputed} \leftarrow \textsc{GetDisagreementItems}(C[\cdot, 1])$
for all $a \in \{A_0, A_1, A_2\}$ **do**
  for all entry $e \in \textsc{Disputed}$ **do**
    $C[a, 2][e] \leftarrow \textsc{ScoreItem}(a, e)$
  end for
end for
for all entry e appearing in any draft **do**
  $\textsc{ConfMean}[e] \leftarrow$ average of $C[a, 2][e]$ over agents that scored
  $\textsc{ConfVar}\ [e] \leftarrow$ variance of same scores
end for
for all $k \in \{\texttt{Knowns}, \texttt{Unknowns}, \texttt{DomainTags}, \texttt{Skills}\}$ **do**
  for all entry e under key k in any draft **do**
    if $\textsc{ConfVar}[e] < \delta$ **and** $\textsc{ConfMean}[e] > \tau$ **then**
      $\mathcal{P}[k] \leftarrow \mathcal{P}[k] \cup \{e\}$ {high-agreement, high-confidence}
    else if A_0 supports e with confidence $> \tau_{\text{gold}}$ **then**
      $\mathcal{P}[k] \leftarrow \mathcal{P}[k] \cup \{e\}$ {trust vote}
    else
      discard e
    end if
  end for
end for

return $\mathcal{P}$

As shown in Algorithm 1, each agent will generate a draft in JSON format, including the known conditions of the problem, unknown conditions, the field and sub-field of the problem, and possible useful problem-solving skills in related fields.

The three drafts next undergo a two-round micro-debate. In round 1, every agent scores the other two drafts on every item; the system collects those entries that still show substantial disagreement and, in round 2, has the agents rescore only this disputed set, producing a refined confidence matrix. For each candidate entry, the orchestrator then computes the round-two mean and variance: any item with low variance and a mean above the threshold τ—or one strongly

endorsed by the script-aware agent A_0 with a score exceeding τ_{gold}—is retained, whereas all remaining low-confidence items are discarded. The surviving information is merged into a compact, high-confidence Problem Passport $\mathcal{P}$, which is then posted to the blackboard to prime the downstream Student–Teacher reasoning loop.

The primary objective of the Problem-Analysis Stage is to construct an explicit, high-level "conceptual scaffold" that primes the downstream real reasoning steps. Through the two-round, tri-agent micro-debate, the system distills the math problem into a Problem Passport that enumerates (i) all givens and unknowns, (ii) precise domain and sub-domain tags, and (iii) a short list of candidate theorems or problem-solving skills likely to be useful. This structured metadata serves as a form of targeted retrieval: it narrows the solution search space and prompts the fine-tuned model to activate the internal knowledge most relevant to this math problem, thereby guiding the subsequent reasoning to remain focused and use tokens more efficiently.

Refinement Stage. After the Problem Passport is posted to the blackboard, we designed a collaborative paradigm, where the Reflection Coordinator (RC) instantiates a number of Student–Teacher teams equal to the user-specified self-reflection count n (with n capped at 4). Formally,

$$\mathcal{R}_\psi : n \longmapsto \left\{ (\mathcal{S}^{(k)},\ \mathcal{T}^{(k)},\ \tau_k) \right\}_{k=1}^{\min(n,4)} \tag{1}$$

Each triple contains a Student policy, a Teacher policy, and the target-step index τ_k that the pair must handle; Students see only the problem statement, whereas Teachers additionally know the correct solution script. For clarity, we denote,

$$\mathcal{S}_\theta : \big(P,\ H_{k-1}\big) \longmapsto s_k \tag{2}$$

$$\mathcal{T}_\phi : \big(s_k,\ C_k\big) \longmapsto \big(v_k,\ c_k\big) \tag{3}$$

where $\mathcal{S}_\theta$ drafts a new block s_k of reasoning steps from the problem P and the verified history H_{k-1}, and $\mathcal{T}_\phi$ instantly validates that block against the correct solution script C_k, returning a flag $v_k \in \{0, 1\}$ and an in-line correction c_k.

With $n = 2$ as an example, RC first instructs $\big(\mathcal{S}^{(1)}, \mathcal{T}^{(1)}\big)$ to let the Student produce the first 1-3 reasoning steps (the exact block size is chosen by RC). The Teacher then validates this block immediately: if it is correct, the Teacher writes a short verification paragraph and posts "`verified-1`" to the blackboard; if it is incorrect, the Teacher appends an in-line correction c_1 while *preserving* the student's original text. The second pair is given the output of the first; its Student generates all remaining reasoning steps, and its Teacher validates or corrects them in the same manner. When $n = 0$, RC spawns only a Teacher, which rewrites the correct solution into a logically rigorous, fine-grained script that contains no self-reflection.

The refinement stage aims to transform the script of the correct solution into a CoT that can be directly used for fine-tuning, and its granularity can be adjusted according to the experimental requirements for small model training. Both the Student and Teacher agents are instantiated with carefully engineered prompt templates that compel them to produce richly annotated micro-steps—each step includes the necessary formulae, local justification, and an explicit indication of how it follows from the previous step, ensuring a smooth and coherent reasoning flow. After every student block, the teacher either issues a terse "verified" paragraph or supplies an in-line correction, so that the emerging solution remains both error-free and pedagogically dense. Crucially, the number s of self-reflection is user-controlled, allowing the framework to dynamically vary the amount of self-reflection and explanatory detail, which enables systematic exploration of how much self-reflection content is best for small models.The reasoning steps generated at this stage have clear semantics and explicit logic, which enables the small model to better absorb the internalized knowledge and reasoning patterns in the reasoning chain. Because the Student agent itself is a Qwen2.5-7B-Instruct instance—the same model that will be fine-tuned on the generated data—its sentence patterns, vocabulary range, and token distribution are intrinsically aligned with the target model's expressive capacity, yielding maximally "on-distribution" supervision signals.

Experiment

Math Problem Selection

In terms of data selection, we prioritize the challenge and diversity of questions. Challenging questions allow the model to learn different problem-solving methods and patterns, stimulate the model's diverse thinking process, and fully mobilize the model's pre-trained knowledge. At the same time, diverse question types can prevent the model from falling into overfitting to a single path, significantly improving its generalization and transfer capabilities on unfamiliar question types. Therefore, we selected AIME history exam questions from 1983-2023, math problems from MATH (Hendrycks et al. 2021) dataset and a small number of high-quality questions collected from the Internet, which offer both high difficulty and broad topical coverage, thus meeting our dual criteria of challenge and diversity. After manual screening and processing with the MACoT framework, we assembled the MACoT dataset, which contains 1,879 math problems and their corresponding solutions generated by the framework.

Experimental Setup

We fine-tuned Qwen2.5-7B-Instruct using LoRA(with a LoRA rank of 8) on our MACoT dataset. The model was trained with a batch size of 4 for four epochs.

Dataset. We benchmark our method on 9 publicly available test suites: **AIME**: 60 short-answer problems from the 2024 and 2025 (Balunovic et al. 2025) American Invitational Mathematics Examination; **MATH-500** (Hendrycks et al. 2021); **AMC23**: Forty questions from the 2023 AMC competition; **GSM8K** (Cobbe et al. 2021); **Minerva-Math** (Lewkowycz et al. 2022); **BBH(causal judgement)**: 187 examples of short scenarios designed to evaluate a model's ability to make human-like causal inferences (Suzgun et al. 2023); **GPQA** (Rein et al. 2024); **Omni-Math**: A subset of 500 data from the Omni-Math dataset (Gao et al. 2025a);

Datasets	Qwen2.5-7B Instruct	Numina	Limo	R1-long	Bespoke	MACoT (ours)
In-domain						
AIME	6.7	3.3	10.0	10.0	8.3	**11.67**
MATH500	74.0	61.4	76.0	70.4	65.2	**77.4**
AMC23	47.5	32.5	**52.5**	37.5	40.0	**52.5**
Out-of-domain						
GSM8K	91.43	87.26	91.58	91.0	87.95	**92.2**
Minerva	48.16	41.54	50.0	44.1	34.56	**51.84**
BBH	55.61	57.22	58.82	56.15	62.57	**63.64**
GPQA	26.79	22.77	28.12	20.1	18.3	**29.46**
Gaokao	**41.31**	30.48	39.32	28.4	28.77	**41.31**
Omni-Math	29.8	21.2	**31.6**	22.0	21.8	30.4
Avg. Accuracy	46.8	39.7	48.7	42.2	40.8	**50.04**
Avg. Token	773	671	837	**4972**	3184	745

Table 1: Pass@1 accuracy (%) on math and reasoning benchmarks

Gaokao: the Gaokao MathQA subtask of AGIEval (Zhong et al. 2024);

Baseline Methods. We compare our proposed MACoT with five representative baselines: **Qwen2.5-7B-Instruct**: which serves as our base model for comparative analysis (Qwen 2024); **NuminaMath-CoT**: Approximately 860k math problems, where each solution is formatted in a CoT manner (LI et al. 2024). Based on the data sources, we randomly selected 6000 CoTs from *amc-aime* and 4000 CoTs from *MATH* to ensure that the data sources of this baseline are consistent with those of our method; **Limo**: 817 carefully selected mathematics problems, each accompanied by a CoT authored by human and LLM experts (Ye et al. 2025); **R1-long**: a control variant that distills 1879 long CoTs from DeepSeek-R1 on exactly the same set of math problems as our MACoT framework—was fine-tuned on Qwen2.5-7B-Instruct using identical hyper-parameter settings; **Bespoke**: We extracted 1500 reasoning trajectories generated by Deepseek-R1 from *bespokelabs-sky-t1-numina-amc-aime-subset* and *bespokelabs-sky-t1-numina-math-subset* respectively, forming a training set of 3000 data; **MACoT (ours)**: The multi-agent framework that synthesizes 1879 fine-grained, capacity-aligned CoTs, each embedding exactly one self-reflection paragraph (n = 1); Qwen2.5-7B-Instruct is then re-tuned on this dataset under the same hyper-parameter settings as the control variant.

Main Results

Table 1 reports Pass@1 accuracy on nine math-reasoning suites. Across in-domain tasks, MACoT secures the best or joint-best score on every set: 11.67% on AIME, 77.4% on MATH500, and a shared 52.5% on AMC23, exceeding the base Qwen2.5 (6.7%, 74.0%, 47.5%), the short-CoT baseline Numina (3.3%, 61.4%, 32.5%), and the curated LIMO dataset run (10.0%, 76.0%, 52.5%). On the six out-of-domain benchmarks, MACoT continues to generalize strongly: it tops GSM8K (92.2%), Minerva (51.84%), BBH (63.64%), and GPQA (29.46%); ties the base model on Gaokao-MathQA (41.31%); and finishes a close second to LIMO on the Olympiad-level Omni-Math subset (30.4% vs. 31.6%). Overall, MACoT achieves the highest average accuracy of 50.04%, improving on the strongest data-centric baseline LIMO (48.7%), the long-CoT baselines R1-long (42.2%) and Bespoke (40.8%), and the short-CoT baseline Numina (39.7%). These results show that capacity-aligned CoTs enable SLMs to outperform models trained on both much longer and much shorter reasoning traces.

MACoT's accuracy gains come with a modest cost. Its mean token budget is 745 tokens per problem, close to Qwen2.5 (773) and lower than LIMO (837). By contrast, the long-CoT baselines are far more expensive: R1-long consumes 4972 tokens, so MACoT uses only about 15% of that budget, while Bespoke averages 3184 tokens—yet both trail MACoT by more than 7 percentage points. Numina is token-efficient (671 tokens) but suffers from shallow reasoning and therefore lower accuracy. Thus, MACoT delivers the best accuracy–efficiency balance: it matches the compactness of short-form reasoning while decisively outperforming both verbose long-CoT distillation methods and other baselines.

Variant	AIME	Math500	amc23	gsm8k	Minerva	BBH
Base Model	6.7	74.0	47.5	91.43	48.16	55.61
w/o Passport	8.3	76.4	45.0	91.9	50.4	58.3
w/o Reflection	8.3	76.6	50.0	92.0	51.1	61.5
MACoT (n=1)	**11.67**	**77.4**	**52.5**	**92.2**	**51.84**	**63.64**

Table 2: Ablation results: Pass@1 accuracy (%)

Ablation Study

To isolate the contribution of each stage in the MACoT framework, we conduct two ablation experiments based on a fixed configuration.

- **MACoT w/o Passport**: We remove the problem analysis stage and only kept the student and teacher agents collaborating to generate the final reasoning steps, where the number of self-reflection paragraphs is set to 1.

- **MACoT w/o Reflection**: We remove the Student–Teacher refinement stage and only let the teacher agent directly generate reasoning steps without self-reflection content. This will test whether moderate self-reflection content is beneficial for small models.

Table 2 shows that deleting the "Problem Passport" causes the largest accuracy drop, indicating that the analysis stage contributes most to the overall gains; the passport guides the fine-tuned model to activate the subset of its internal knowledge most relevant to the question, thereby boosting reasoning accuracy. Removing the "Student–Teacher refinement" stage also lowers performance, though less sharply, suggesting that controlled self-reflection improves the model's error-correction capacity. Notably, both ablated variants still outperform the base model, confirming that each stage of our framework adds value to the resulting chains of thought.

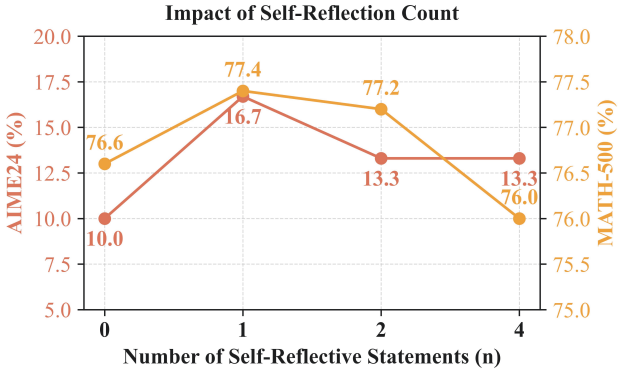

Figure 3: Accuracy Comparison: Impact of Self-Reflection Count on AIME24 and MATH-500.

Quantitative Analysis

In order to study the impact of self-reflective statements(e.g., "Wait, let me check again" "However") in the CoTs on the model performance, we regenerated four versions of the training set in which each CoT contains exactly 0, 1, 2, or 4 self-reflective paragraphs (denoted N0, N1, N2, N4). We fine-tuned Qwen2.5-7B-Instruct on each variant and evaluated on AIME24 and MATH-500. Results are in Figure 3.

The results revealed a clear trend: N1 produced the highest accuracies (16.7% on AIME24, 77.4% on MATH-500), marginally surpassing the baseline (N0). Once the number of self-reflective statements exceeded one, performance declined monotonically: N2 showed an average drop of 1.8 points per benchmark, while N4 declined by an average of 2.4 points. The evidence suggests that one lightweight self-reflection marker can help the model organize its reasoning, but additional markers quickly consume context budget and trigger counter-productive self-corrections. In short, self-reflective statements must be kept scarce to remain in harmony with the capacity of small models.

Control Experiment

A legitimate concern is that MACoT's gains could stem merely from different training data rather than from the multi-agent synthesis itself. To rule out this possibility, we designed a control experiment in which the exact same 817 math problems—those that constitute the public LIMO dataset—are used in both training conditions:

- **Limo**: Use LIMO's original dataset to fine-tune Qwen2.5-7b-instruct using LoRA.
- **MACoT**: Based on the math problems in the Limo dataset, MACoT framework is used to synthesize CoTs (where the self-reflection paragraph n is set to 1).

All training parameters are kept consistent, and the results are shown in Figure 4.

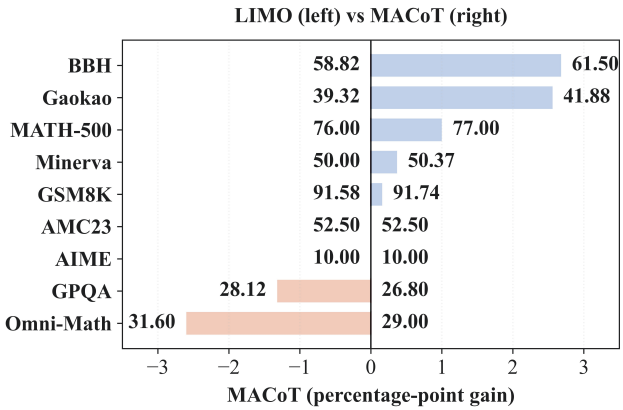

Figure 4: Accuracy Comparison Using the Same Set of Math Problems: MACoT-Synthesized CoTs (n=1) vs. Original LIMO CoTs.

The results are clear and consistent across nine diverse math and reasoning benchmarks. MACoT outperforms or matches LIMO on seven out of nine datasets, showing no degradation on any in-domain task. Specifically, MACoT delivers a notable 2.68 percentage-point gain on BBH and a 2.56-point gain on Gaokao, and improves by 1.00 point on the widely used MATH-500 benchmark. It also registers marginal improvements on Minerva and GSM8K. On the two competition-style datasets (AIME and AMC23), MACoT matches LIMO's performance. While slight regressions are observed on GPQA and Omni-Math, the overall trend favors MACoT. Since both models are trained on the same set of math problems, the observed accuracy gains can be directly attributed to the design of MACoT's ability-aligned CoTs—more concise and better tailored to small language models' reasoning capacity. These results suggest that while long-CoTs can help, excessive self-reflection may exceed the learning capacity of small language models and diminish fine-tuning efficacy. MACoT instead strikes a better balance between depth and efficiency.

Conclusion

In this work, we present MACoT, a multi-agent framework designed to synthesize CoTs for mathematical reasoning. MACoT synthesizes high-quality reasoning traces tailored to SLMs. Our results show that capability-aligned CoTs can improve both accuracy and generalization of SLMs.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant 2023YFF0905503, National Natural Science Foundation of China under Grants No.62472188.

References

Balunovic, M.; Dekoninck, J.; Petrov, I.; Jovanović, N.; and Vechev, M. 2025. MathArena: Evaluating LLMs on Uncontaminated Math Competitions. In *Proceedings of the Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. San Diego, California, USA.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. arXiv:2110.14168.

DeepSeek-AI; Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; Zhang, X.; Yu, X.; Wu, Y.; Wu, Z. F.; Gou, Z.; Shao, Z.; Li, Z.; Gao, Z.; Liu, A.; Xue, B.; Wang, B.; Wu, B.; Feng, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; Ruan, C.; Dai, D.; Chen, D.; Ji, D.; Li, E.; Lin, F.; Dai, F.; Luo, F.; Hao, G.; Chen, G.; Li, G.; Zhang, H.; Bao, H.; Xu, H.; Wang, H.; Ding, H.; Xin, H.; Gao, H.; Qu, H.; Li, H.; Guo, J.; Li, J.; Wang, J.; Chen, J.; Yuan, J.; Qiu, J.; Li, J.; Cai, J. L.; Ni, J.; Liang, J.; Chen, J.; Dong, K.; Hu, K.; Gao, K.; Guan, K.; Huang, K.; Yu, K.; Wang, L.; Zhang, L.; Zhao, L.; Wang, L.; Zhang, L.; Xu, L.; Xia, L.; Zhang, M.; Zhang, M.; Tang, M.; Li, M.; Wang, M.; Li, M.; Tian, N.; Huang, P.; Zhang, P.; Wang, Q.; Chen, Q.; Du, Q.; Ge, R.; Zhang, R.; Pan, R.; Wang, R.; Chen, R. J.; Jin, R. L.; Chen, R.; Lu, S.; Zhou, S.; Chen, S.; Ye, S.; Wang, S.; Yu, S.; Zhou, S.; Pan, S.; Li, S. S.; Zhou, S.; Wu, S.; Ye, S.; Yun, T.; Pei, T.; Sun, T.; Wang, T.; Zeng, W.; Zhao, W.; Liu, W.; Liang, W.; Gao, W.; Yu, W.; Zhang, W.; Xiao, W. L.; An, W.; Liu, X.; Wang, X.; Chen, X.; Nie, X.; Cheng, X.; Liu, X.; Xie, X.; Liu, X.; Yang, X.; Li, X.; Su, X.; Lin, X.; Li, X. Q.; Jin, X.; Shen, X.; Chen, X.; Sun, X.; Wang, X.; Song, X.; Zhou, X.; Wang, X.; Shan, X.; Li, Y. K.; Wang, Y. Q.; Wei, Y. X.; Zhang, Y.; Xu, Y.; Li, Y.; Zhao, Y.; Sun, Y.; Wang, Y.; Yu, Y.; Zhang, Y.; Shi, Y.; Xiong, Y.; He, Y.; Piao, Y.; Wang, Y.; Tan, Y.; Ma, Y.; Liu, Y.; Guo, Y.; Ou, Y.; Wang, Y.; Gong, Y.; Zou, Y.; He, Y.; Xiong, Y.; Luo, Y.; You, Y.; Liu, Y.; Zhou, Y.; Zhu, Y. X.; Xu, Y.; Huang, Y.; Li, Y.; Zheng, Y.; Zhu, Y.; Ma, Y.; Tang, Y.; Zha, Y.; Yan, Y.; Ren, Z. Z.; Ren, Z.; Sha, Z.; Fu, Z.; Xu, Z.; Xie, Z.; Zhang, Z.; Hao, Z.; Ma, Z.; Yan, Z.; Wu, Z.; Gu, Z.; Zhu, Z.; Liu, Z.; Li, Z.; Xie, Z.; Song, Z.; Pan, Z.; Huang, Z.; Xu, Z.; Zhang, Z.; and Zhang, Z. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948.

Gao, B.; Song, F.; Yang, Z.; Cai, Z.; Miao, Y.; Dong, Q.; Li, L.; Ma, C.; Chen, L.; Xu, R.; Tang, Z.; Wang, B.; Zan, D.; Quan, S.; Zhang, G.; Sha, L.; Zhang, Y.; Ren, X.; Liu, T.; and Chang, B. 2025a. Omni-MATH: A Universal Olympiad Level Mathematic Benchmark for Large Language Models. In *Proceedings of the Thirteenth International Conference on Learning Representations*. Singapore EXPO.

Gao, X.; Pei, Q.; Tang, Z.; Li, Y.; Lin, H.; Wu, J.; Wu, L.; and He, C. 2025b. A Strategic Coordination Framework of Small LMs Matches Large LMs in Data Synthesis. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 11552–11570. Vienna, Austria: Association for Computational Linguistics.

Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. Measuring Mathematical Problem Solving With the MATH Dataset. In *Proceedings of the Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. Virtual: Curran Associates, Inc.

Lewkowycz, A.; Andreassen, A.; Dohan, D.; Dyer, E.; Michalewski, H.; Ramasesh, V.; Slone, A.; Anil, C.; Schlag, I.; Gutman-Solo, T.; Wu, Y.; Neyshabur, B.; Gur-Ari, G.; and Misra, V. 2022. Solving Quantitative Reasoning Problems with Language Models. In *Proceedings of the Thirty-Sixth Conference on Neural Information Processing Systems*, volume 35, 3843–3857. New Orleans, USA: Curran Associates, Inc.

LI, J.; Beeching, E.; Tunstall, L.; Lipkin, B.; Soletskyi, R.; Huang, S. C.; Rasul, K.; Yu, L.; Jiang, A.; Shen, Z.; Qin, Z.; Dong, B.; Zhou, L.; Fleureau, Y.; Lample, G.; and Polu, S. 2024. NuminaMath. https://huggingface.co/datasets/AI-MO/NuminaMath-CoT. Accessed: 2025-05-20.

Li, L. H.; Hessel, J.; Yu, Y.; Ren, X.; Chang, K.-W.; and Choi, Y. 2023. Symbolic Chain-of-Thought Distillation: Small Models Can Also "Think" Step-by-Step. In *Proceedings of the Association for Computational Linguistics (Volume 1: Long Papers): ACL 2023*, 2665–2679. Toronto, Canada: Association for Computational Linguistics.

Li, Y.; Yue, X.; Xu, Z.; Jiang, F.; Niu, L.; Lin, B. Y.; Ramasubramanian, B.; and Poovendran, R. 2025. Small Models Struggle to Learn from Strong Reasoners. In *Findings of the Association for Computational Linguistics: ACL 2025*, 25366–25394. Vienna, Austria: Association for Computational Linguistics.

Liu, T.; Wang, X.; Huang, W.; Xu, W.; Zeng, Y.; Jiang, L.; Yang, H.; and Li, J. 2024. GroupDebate: Enhancing the Efficiency of Multi-Agent Debate Using Group Discussion. arXiv:2409.14051.

Luo, R.; Li, J.; Huang, C.; and Lu, W. 2025. Through the Valley: Path to Effective Long CoT Training for Small Language Models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 4972–4992. Suzhou, China: Association for Computational Linguistics.

Majumdar, S.; Noroozi, V.; Samadi, M.; Narenthiran, S.; Ficek, A.; Ahmad, W. U.; Huang, J.; Balam, J.; and Ginsburg, B. 2025. Genetic Instruct: Scaling up Synthetic Generation of Coding Instructions for Large Language Models. In *Proceedings of the Association for Computational Linguistics (Volume 6: Industry Track): ACL 2025*, 208–221. Vienna, Austria: Association for Computational Linguistics.

Muennighoff, N.; Yang, Z.; Shi, W.; Li, X. L.; Fei-Fei, L.; Hajishirzi, H.; Zettlemoyer, L.; Liang, P.; Candes, E.; and

Hashimoto, T. 2025. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 20286–20332. Suzhou, China: Association for Computational Linguistics.

OpenAI. 2025. Introducing OpenAI O3 and O4-Mini. https://openai.com/index/introducing-o3-and-o4-mini. Accessed: 2025-04-16.

Pang, J.; Di, N.; Zhu, Z.; Wei, J.; Cheng, H.; Qian, C.; and Liu, Y. 2025. Token Cleaning: Fine-Grained Data Selection for LLM Supervised Fine-Tuning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267. Vancouver, Canada: PMLR.

Parmar, M.; Liu, X.; Goyal, P.; Chen, Y.; Le, L.; Mishra, S.; Mobahi, H.; Gu, J.; Wang, Z.; Nakhost, H.; Baral, C.; Lee, C.-Y.; Pfister, T.; and Palangi, H. 2025. PlanGEN: A Multi-Agent Framework for Generating Planning and Reasoning Trajectories for Complex Problem Solving. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 20651–20677. Suzhou, China: Association for Computational Linguistics.

Qwen. 2024. Qwen2.5: A Party of Foundation Models. https://qwenlm.github.io/blog/qwen2.5. Accessed: 2025-05-21.

Rein, D.; Hou, B. L.; Stickland, A. C.; Petty, J.; Pang, R. Y.; Dirani, J.; Michael, J.; and Bowman, S. R. 2024. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. In *Proceedings of the First Conference on Language Modeling*. Philadelphia, USA.

Shridhar, K.; Stolfo, A.; and Sachan, M. 2023. Distilling Reasoning Capabilities into Smaller Language Models. In *Findings of the Association for Computational Linguistics: ACL 2023*, 7059–7073. Toronto, Canada: Association for Computational Linguistics.

Suzgun, M.; Scales, N.; Schärli, N.; Gehrmann, S.; Tay, Y.; Chung, H. W.; Chowdhery, A.; Le, Q.; Chi, E.; Zhou, D.; and Wei, J. 2023. Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. In *Findings of the Association for Computational Linguistics: ACL 2023*, 13003–13051. Toronto, Canada: Association for Computational Linguistics.

Wang, Z.; Jiang, J.; Qiu, T.; Liu, H.; Tang, X.; and Yao, H. 2025. Efficient Long CoT Reasoning in Small Language Models. In *NeurIPS 2025 Workshop on Efficient Reasoning*. San Diego, California, USA.

Wu, Y.; Wang, Y.; Du, T.; Jegelka, S.; and Wang, Y. 2025. When More is Less: Understanding Chain-of-Thought Length in LLMs. In *ICLR 2025 Workshop on Reasoning and Planning for Large Language Models*. Singapore EXPO.

Xia, H.; Leong, C. T.; Wang, W.; Li, Y.; and Li, W. 2025. TokenSkip: Controllable Chain-of-Thought Compression in LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 3351–3363. Suzhou, China: Association for Computational Linguistics.

Xu, Z.; Jiang, F.; Niu, L.; Deng, Y.; Poovendran, R.; Choi, Y.; and Lin, B. Y. 2025. Magpie: Alignment Data Synthesis from Scratch by Prompting Aligned LLMs with Nothing. In *Proceedings of the Thirteenth International Conference on Learning Representations*. Singapore EXPO.

Ye, Y.; Huang, Z.; Xiao, Y.; Chern, E.; Xia, S.; and Liu, P. 2025. LIMO: Less is More for Reasoning. In *Proceedings of the Second Conference on Language Modeling*. Montreal, Canada.

Zhang, J.; Zhu, Y.; Sun, M.; Luo, Y.; Qiao, S.; Du, L.; Zheng, D.; Chen, H.; and Zhang, N. 2025. LightThinker: Thinking Step-by-Step Compression. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 13318–13339. Suzhou, China: Association for Computational Linguistics.

Zhang, Z.; Ge, T.; Liang, Z.; Yu, W.; Yu, D.; Jia, M.; Yu, D.; and Jiang, M. 2024. Learn Beyond The Answer: Training Language Models with Reflection for Mathematical Reasoning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 14720–14738. Miami, Florida, USA: Association for Computational Linguistics.

Zhao, S.; Yuan, J.; Yang, G.; and Naseem, U. 2025. Can Pruning Improve Reasoning? Revisiting Long-CoT Compression with Capability in Mind for Better Reasoning. arXiv:2505.14582.

Zhong, W.; Cui, R.; Guo, Y.; Liang, Y.; Lu, S.; Wang, Y.; Saied, A.; Chen, W.; and Duan, N. 2024. AGIEval: A Human-Centric Benchmark for Evaluating Foundation Models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, 2299–2314. Mexico City, Mexico: Association for Computational Linguistics.