K-STaR: Knowledge-Aware Self-Taught Reasoner

Guozheng Li[*†‡], **Xinyu Zhang**[*]

Southeast University
{gzli, zxyseu}@seu.edu.cn

Abstract

Self-training large language models (LLMs) with generated reasoning paths has emerged as a promising approach to improve performance on complex reasoning tasks. However, most existing methods rely on correctness-based supervision, treating samples that reach the correct answer as high-quality despite potentially flawed intermediate steps, leading to noisy training signals. In this work, we propose **K-STaR** (**K**nowledge-aware **S**elf-**Ta**ught **R**easoner), a self-training framework that *verifies reasoning paths through knowledge elicitation and integration as a proxy, without requiring any external reward models or dense step-by-step annotations*. K-STaR models reasoning as a structured composition of knowledge units and automatically assigns process rewards to intermediate steps via consistency and frequency analysis, ensuring that only knowledge-grounded reasoning paths are retained. Experiments on mathematical and commonsense reasoning tasks show that K-STaR consistently discovers higher-quality reasoning paths and achieves superior self-training performance compared to prior methods. Our results highlight the importance of *moving beyond correctness-centric supervision toward knowledge-grounded self-improvement*.

Introduction

Recent studies show that post-training large language models (LLMs) with explicit intermediate reasoning trajectories can significantly boost their performance on complex reasoning tasks like mathematical reasoning and commonsense reasoning. However, obtaining high-quality reasoning paths typically requires detailed human supervision. As most high-quality web text has already been used for training, research is now shifting toward leveraging LLM-generated content for self-training (Bai et al. 2022; Chen et al. 2024; Hosseini et al. 2024; Huang et al. 2023; Jiao et al. 2024; Lightman et al. 2024; Peng et al. 2025; Singh et al. 2023; Yuan et al. 2023; Zelikman et al. 2022; Zhang et al. 2024b). Similar to most reinforcement learning problems, LLM self-training requires a reward signal. Most existing reinforced self-improvement approaches (e.g., STaR (Zelikman et al. 2022), RFT (Yuan et al. 2023), ReST^{EM} (Singh et al. 2023), V-STaR (Hosseini et al. 2024)) assume access to a ground-truth reward model, either in the form of supervised dataset labels or a pre-trained reward model. These methods use an LLM to generate multiple chain-of-thought (CoT) (Wei et al. 2022) reasoning paths for each question and treat the sample leading to the correct solution as high-quality, which is then used for further training. Such procedures show effective in improving LLM performance across a range of reasoning tasks.

However, a key limitation of this procedure is that even if a reasoning path leads to the correct solution, it does not necessarily mean the entire path is accurate. LLMs often produce incorrect or irrelevant intermediate steps while still arriving at the correct answer by chance (Lanham et al. 2023). As a result, self-training datasets can contain many false positives — paths where the final output is correct but the intermediate reasoning is flawed — which ultimately limits the effectiveness of LLM fine-tuning for complex reasoning tasks (Xia et al. 2024; Zhou et al. 2023). One way to addressing this issue is to use a value function or reward model to verify the correctness of reasoning paths (Jiao et al. 2024; Lightman et al. 2024; Luo et al. 2025; Wang et al. 2024). However, training a reliable reward model typically requires dense human annotations at each reasoning step (Lightman et al. 2024), or costly evaluations like Monte Carlo Tree Search (MCTS) to estimate the overall trajectory value, which does not scale well. Our research aims to address this gap by developing a novel approach that automates the acquisition of reliable reasoning traces while effectively utilizing reward signals for verification purposes. The central question is: *How to automatically acquire high-quality reasoning paths and effectively process reward signals for verification and self-training*?

In this work, we propose **K-STaR** (**K**nowledge-aware **S**elf-**Ta**ught **R**easoner), a self-improvement framework that assesses the reliability of reasoning paths by incorporating **knowledge elicitation and integration** into the reasoning path, **without relying on any reward model for step-by-step verification**, as shown in Figure 1. Our motivation stems from the fundamental understanding that knowledge provides both the foundation and constraints for reasoning, while reasoning relies on knowledge to generate new conclusions or refine existing knowledge. In the reasoning process, we treat knowledge-related operations as a *proxy* for evaluating the quality of reasoning paths, thus shifting self-improvement

[*]These authors contributed equally.

[†]Corresponding author.

[‡]This work was supported by the SEU Innovation Capability Enhancement Plan for Doctoral Students (No. CXJH_SEU 25031).

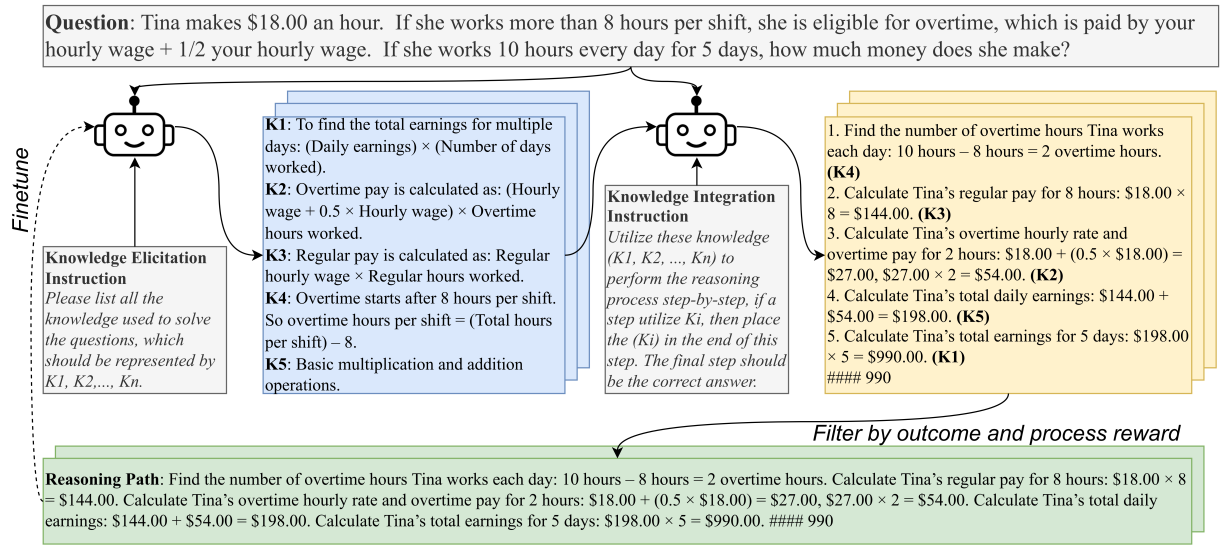

Figure 1: K-STaR is a self-improvement method designed to enhance the reasoning capabilities of LLMs by leveraging structured knowledge exploration and path refinement. We first create a set of related knowledge unit sets based on the given question. Then each knowledge unit set is used to generate multiple potential reasoning paths and predicted answers. We filter out incorrect reasoning paths using either outcome reward (ground-truth) or process reward (knowledge operations). Finally, the correct reasoning paths, along with the instructions and answers, are used to fine-tune the LLM, enhancing its reasoning capability.

methods from correctness-centric to knowledge-rigorous optimization. Instead of treating rationales as monolithic texts, K-STaR explicitly models reasoning as a composition of knowledge units. For each question, we first sample diverse knowledge units through few-shot prompting (Brown et al. 2020). Then we statistically validate knowledge and generate grounded rationales via consistency and frequency analysis, retaining only reasoning chains grounded in high-consistency knowledge (Wang et al. 2023; Li, Wang, and Ke 2023). This process acts as a knowledge filter, ensuring each sample inherently *encodes valid knowledge patterns*, without requiring additional human intervention or model training. We validate experimentally that K-STaR outperforms prior work in discovering good reasoning paths on the GSM8K (Cobbe et al. 2021), NumGLUE (Mishra et al. 2022), CommonsenseQA (Talmor et al. 2019) and StrategyQA (Geva et al. 2021) benchmarks, which consequently leads to improved self-training. To summarize, our contributions are:

- We propose K-STaR, a self-training method that generates reliable reasoning paths through knowledge elicitation and integration. The key idea is the automatic annotation of process rewards for intermediate steps using consistency and frequency analysis, enabling the estimation of process and outcome rewards for complete reasoning paths.
- K-STaR provides process supervision at the knowledge level without the need to train an additional reward model, making it both simple and effective. We validate multiple reasoning benchmarks and find that K-STaR outperforms prior self-improvement approaches. We perform a comprehensive analysis of our framework, evaluating the effects of various models, design choices, filtering methods across different language models.

Related Work

Language Model Reasoning. Recent advanced LLM reasoning algorithms develop effective prompting techniques (Kojima et al. 2022; Wang et al. 2023; Wei et al. 2022; Yao et al. 2023) to elicit the reasoning capability of LLMs. For example, chain-of-thought (CoT) (Wei et al. 2022) imitates the thought process like humans to provide step-by-step solutions given a question, making it easier to solve complex, multi-step reasoning tasks. Self-consistency (SC) (Wang et al. 2023) improves the reliability and self-consistency of answers by sampling multiple interpretations from LLMs and then selecting the final answer that appears most frequently. Tree-of-thoughts (ToT) (Yuan et al. 2023) further generalizes the CoT methodology by considering multiple different reasoning paths in the tree and exploring coherent units of thought to execute thoughtful decision-making. These prompting methods can generate more diverse and general reasoning paths.

Language Model Training. Recent studies focus on improving the reasoning capability of LLMs, including collecting high-quality or larger domain-specific data (Gunasekar et al. 2023; Taylor et al. 2022; Touvron et al. 2023a,b; Zhang et al. 2024a), or training with supervised learning (Yu et al. 2024; Yue et al. 2024; Zhang et al. 2024a), or reinforcement learning (Dong et al. 2023; Gulcehre et al. 2023; Rafailov et al. 2023; Yuan et al. 2024). Empirical results show that employing explicit reasoning paths with extensive supervision from either humans or superior models, can lead to better reasoning capabilities after post-training LLMs (Achiam et al. 2023; Lightman et al. 2024; Sun et al. 2023). For example, Orca (Mukherjee et al. 2023) and Orca2 (Mitra et al. 2023) illustrate how smaller models can benefit from detailed explanations and step-by-step reasoning processes provided

Method	Value Label	Outcome	Process	w/o Train
STaR (Zelikman et al. 2022)	**Outcome reward** annotated by ground-truth answer	✓	✗	✓
RFT (Yuan et al. 2023)	**Outcome reward** annotated by ground-truth answer	✓	✗	✓
ReST^{EM} (Zhang et al. 2024b)	**Outcome reward** annotated by ground-truth answer	✓	✗	✓
RPO (Pang et al. 2024)	**Outcome reward** annotated by ground-truth answer	✓	✗	✓
LMSI (Huang et al. 2023)	**Outcome reward** annotated by self-consistent answers	✓	✗	✓
Self-Rewarding (Yuan et al. 2024)	**Outcome reward** generated and judged by LLMs	✓	✗	✓
V-STaR (Hosseini et al. 2024)	**Outcome reward** generated by multi-iteration LLMs	✓	✗	✗
Verify Step-by-Step (Lightman et al. 2024)	**Process reward** annotated by human	✓	✓	✗
MATH-SHEPHERD (Wang et al. 2024)	**Process reward** inferred from random rollout	✓	✓	✗
pDPO (Jiao et al. 2024)	**Process reward** inferred from random rollout	✓	✓	✗
TS-LLM (Feng et al. 2023)	**Process reward** inferred from TD-λ (Sutton 1988)	✓	✓	✗
ReST-MCTS* (Zhang et al. 2024b)	**Process reward** inferred from tree search	✓	✓	✗
K-STaR (Ours)	**Process reward** inferred from knowledge as a proxy signal	✓	✓	✓

Table 1: Key differences between existing methods, w/o Train indicates whether to explicitly train a process reward model.

by GPT-4 (Achiam et al. 2023). Moreover, process supervision methods (Lightman et al. 2024; Luo et al. 2025; Wang et al. 2024) uses extensive finer-grained human annotations on step-by-step reasoning paths as post-training data. These approaches all rely on more advanced teacher models or human experts for extensive supervision to obtain high-quality, explicit intermediate reasoning paths, but we are curious to see how LLMs can leverage itself on generating reasoning paths to enhance its own reasoning capabilities.

Language Model Self-Training. Given LLMs' ability to generate strong rationales and solutions, many approaches use their correct outputs to further boost reasoning, a process known as *self-training* or *self-improvement*. Self-training involves sampling reasoning traces from the base LLM and then deriving a learning signal to fine-tune it. This process can be repeated, with the main challenge being the design of an effective learning signal. Two common approaches are Outcome Reward Models (ORMs) (Cobbe et al. 2021), trained on final answer correctness, and Process Reward Models (PRMs) (Lightman et al. 2024), trained on the correctness of each reasoning step. Ideally, a dense signal at every intermediate step, as provided by PRMs, is preferred; otherwise, sparse signals suffer from credit assignment issues akin to those in reinforcement learning. However, reliable PRM typically relies on dense human supervision for each reasoning step or costly trajectory evaluations via methods like MCTS, making it difficult to scale, as shown in Table 1. Compared to existing self-improvement methods, K-STaR fundamentally differs in that it leverages the strong correlation between reasoning and knowledge. It utilizes knowledge states and operations along the reasoning path to provide both process rewards in addition to traditional outcome rewards, without the need to train an explicit PRM.

Knowledge-Aware Self-Taught Reasoner

We adopt the standard setup for LLM-based reasoning. Starting from a policy π instantiated by a base LLM, given a question q, the model generates a reasoning trace $\tau = (s_1, s_2, ..., s_T) \sim \pi(\cdot|q)$ by autoregressively predicting reasoning steps, where each step s_i is a sentence. We assume the final step s_T produces the answer. Given a new training dataset $\mathcal{D} = \{(q_i, a_i)\}_{i=1}^{|\mathcal{D}|}$, self-training methods use generator π to generate reasoning steps τ_i and final answer a_i per question q_i. K-STaR adopts a strategy to generate reasoning paths by progressing from knowledge elicitation and knowledge integration, achieving reliable reasoning from correctness-centric to knowledge-rigorous self-improvement. Building on this idea, K-STaR can automatically generate and integrate reasoning paths into post-training datasets that originally contain only instructions and ground-truth answers. Figure 1 shows how we create such reasoning paths for a task. All system prompts we used during this process are shown in Appendix A. The goal of K-STaR is to use π to generate reasoning paths as post-training data to fine-tune π, to further improve π's performance on such reasoning tasks.

Generating Reasoning Paths

K-STaR generates reasoning paths through knowledge elicitation and integration. It first decomposes reasoning into discrete knowledge units, then reorganizes and applies these units to complete the task. To ensure high-quality reasoning paths, it is crucial that they both contain correct knowledge units and apply them appropriately. We first construct a small *prompt* set $\mathcal{P} = \{(q_i^*, \kappa_i^*, \tau_i^*, \zeta_i^*, a_i^*)\}_{i=1}^{|\mathcal{P}|}$, where each example consists of a question q_i^*, its associated knowledge unit set κ_i^*, a reasoning path τ_i^*, a knowledge trace ζ_i^*, and the final answer a_i^*. Here, $\kappa_i^* = \{k_j\}_{j=1}^{|\kappa_i^*|}$ denotes the set of knowledge units needed to solve q_i^*, with each k_j representing an individual unit. The reasoning path $\tau_i^* = (s_1, s_2, ..., s_T)$ is annotated with the sequence $\zeta_i^* = (t_1, t_2, ..., t_T)$, where each reasoning step s is associated with a knowledge step t containing one (or more) knowledge units k_j from κ_i^*. When generating new reasoning paths, K-STaR explicitly considers the knowledge utilized at each step, ensuring that each path τ is accompanied by a corresponding knowledge trace ζ. **This allows the validity of the knowledge unit set κ and the reliability of the knowledge trace ζ to serve as proxies for evaluating the value of a reasoning path τ.**

Knowledge Unit Generation. Adopting few-shot prompting, we concatenate $\mathcal{P}$ to each example in $\mathcal{D}$, i.e. $q_i = (q_1^*, \kappa_1^*, ..., q_{|\mathcal{P}|}^*, \kappa_{|\mathcal{P}|}^*, q_i)$, which encourages π to produce a knowledge unit set κ_i for q_i. The instruction for this process is "*Please list all the knowledge used to solve the questions, which should be represented by K1, K2, ..., Kn.*". In this step, for each question q_i, we sample and generate n knowledge unit sets $\{\kappa_i^1, ..., \kappa_i^n\}$, where each κ_i is composed of multiple knowledge units, denoted as $\kappa_i = \{k_1, ..., k_{|\mathcal{K}_i|}\}$. For example, a knowledge unit k could be "*Addition is used to combine regular pay and overtime pay*" for mathematical reasoning or "*Wearing hats is not a universal requirement for work and depends on the job type*" for commonsense reasoning.

Reasoning Path Generation. Each knowledge unit set κ_i is used to generate potential reasoning paths, knowledge traces and answers $(\tau_i, \zeta_i, \hat{a}_i)$. The reasoning path addresses q_i step-by-step based on κ_i. This process also involves few-shot prompting, and we also concatenate $\mathcal{P}$ to each example in $\mathcal{D}$, i.e. $q_i = (q_1^*, \kappa_1^*, \tau_1^*, \zeta_1^*, a_1^*, ..., q_{|\mathcal{P}|}^*, \kappa_{|\mathcal{P}|}^*, \tau_{|\mathcal{P}|}^*, \zeta_{|\mathcal{P}|}^*, a_{|\mathcal{P}|}^*, q_i, \kappa_i)$, which encourages π to produce a reasoning path τ_i accompanied with ζ_i for q_i utilizing κ_i, followed by an answer $\hat{a}_i$. The instruction for this process is "*Utilize these knowledge (K1, K2, ..., Kn) to perform the reasoning process step-by-step, if a step utilize Ki, then place the (Ki) in the end of this step. The final step should be the correct answer.*". In this step, for each question q_i and κ_i, we sample and generate m reasoning paths $\{\tau_i^1, ..., \tau_i^m\}$, highlighting the knowledge trace ζ_i used at each step.

Filtering Reasoning Paths

After obtaining n knowledge unit sets and $n \times m$ reasoning paths per question. Similar to STaR (Zelikman et al. 2022), we filter out incorrect solutions, assuming that only reasoning paths leading to the correct answer (i.e., $\hat{a}_i = a_i$) are valid. However, this approach treats the reasoning paths as opaque and lacks of process supervision, optimizing only for final correctness while ignoring possible spurious correlations or domain rule violations. To address this, we introduce two simple evaluation ideas that leverage knowledge units and knowledge traces for process reward estimation.

Filtering Knowledge Units with Consistency. To select the highest-quality and most useful knowledge unit set κ_i for a given question q_i from n sets of knowledge, a straightforward approach is to evaluate the proportion of reasoning paths among the m paths corresponding to each knowledge set that lead to the correct final answer a_i. If a particular knowledge set consistently enables π to reach the correct answer, then that knowledge set should be selected, which is similar in concept to self-consistency (Wang et al. 2023):

$$\kappa_i^{\text{optimal}} = \underset{\kappa_i, \kappa_i \in \{\kappa_i^1, ..., \kappa_i^n\}}{\arg\max} \sum_{j=1}^{m} \mathbb{1}_{\hat{a}_i^j = a_i} \tag{1}$$

where $\hat{a}_i^j$ represents the answer of j-th reasoning path τ_i^j generated by π through sampling via κ_i. We define $\kappa_i^{\text{optimal}}$ as the optimal set of knowledge units for subsequent filtering of reasoning paths.

Filtering Reasoning Paths with Frequency. Since K-STaR explicitly tracks the knowledge utilized at each reasoning step, each reasoning path $\tau_i = (s_1, s_2, ..., s_T)$ is accompanied by a corresponding knowledge trace $\zeta_i = (t_1, t_2, ..., t_T)$, where each t_j denotes the set of knowledge units used in step s_j, i.e., $t_j = \{k_1, k_2, ..., k_c \mid k_j \in \kappa_i, c \geq 0\}$. In most cases, $c = 1$, meaning that each reasoning step typically involves a single knowledge unit. Thus, ζ_i forms a knowledge chain such as $k_2 \rightarrow k_5 \rightarrow k_1 \rightarrow k_3$. To provide process-level supervision, we define a reward function over knowledge units and their transitions, rather than solely relying on the final answer correctness. We also use ζ_i to represent the knowledge chain corresponding to the reasoning path τ_i, consisting of z knowledge units and $z-1$ ordered knowledge transitions. For all reasoning paths that reach the correct final answer using κ_i, we compute two types of rewards based on their associated knowledge traces. We define the unit reward as the frequency with which a knowledge unit k_x appears across all correct reasoning paths, and the transition reward as the frequency with which a knowledge transition $k_x \rightarrow k_y$ appears across all correct paths:

$$\begin{aligned} \tilde{r}(k_x) &= \frac{\text{count}(k_x)}{\max_a \text{count}(k_a)}, \\ \tilde{r}(k_x \rightarrow k_y) &= \frac{\text{count}(k_x \rightarrow k_y)}{\max_{a \rightarrow b} \text{count}(k_a \rightarrow k_b)} \end{aligned} \tag{2}$$

After normalization (by dividing by the maximum), all rewards are constrained within the range [0, 1], thereby mitigating the impact of extreme values. These process-level rewards serve as fine-grained learning signals, encouraging the model not only to reach the correct answer but also to traverse reliable and interpretable reasoning paths. The average normalized reward of the whole knowledge trace ζ_i (also the reward of reasoning path τ_i) is defined as:

$$r(\zeta_i) = \frac{1}{2z-1} \cdot [\sum_{k_x \in \zeta_i} \tilde{r}(k_x) + \sum_{k_x \rightarrow k_y \in \zeta_i} \tilde{r}(k_x \rightarrow k_y)] \tag{3}$$

We consider both the importance of each knowledge unit and the coherence of their transitions for reward estimation. Finally, we take the average rather than a simple sum, allowing for meaningful comparisons across paths of varying lengths. Then we can rank the value of different reasoning paths τ_i^j obtained from the optimal knowledge $\kappa_i^{\text{optimal}}$.

Supervised Fine-Tuning

For each dataset $\mathcal{D}$, we create a new dataset incorporating reasoning paths, denoted as $\mathcal{D}' = \{q_i, \tau_i^j, a_i\}_{i=1,j=1}^{|\mathcal{D}|,<m}$, where τ_i^j represents the correct reasoning path after filtering. We select up to p correct reasoning paths based on value ranking for each question q_i. K-STaR selects the most reliable reasoning path τ_i for a question q_i via a two-step sampling and evaluation process, leveraging both outcome and process supervision. The practical way to accomplish reasoning tasks on $\mathcal{D}'$ is supervised fine-tuning (SFT) that trains a model by minimizing the negative log-likelihood loss on $\mathcal{D}'$:

$$\mathcal{L}_{\text{SFT}}(\pi) = -\mathbb{E}_{(q_i, \tau_i) \sim \mathcal{D}'} \sum_{t=1}^{T} \log \pi(s_t | s_{<t}, q_i) \tag{4}$$

Models	Methods	GSM8K	NumGLUE	CommonsenseQA	StrategyQA	Average
Mistral-7B-Instruct	w/o SFT	42.3%	38.6%	64.2%	76.4%	55.4%
	w/ SFT	11.8%	53.0%	67.5%	84.6%	54.2%
	STaR	44.5%	46.3%	72.0%	81.1%	61.0%
	RFT	45.2%	47.1%	68.8%	79.3%	60.1%
	RFT w/ PRM	51.7%	51.8%	70.5%	80.5%	63.6%
	ReST^{EM}	52.3%	52.6%	69.2%	80.8%	63.7%
	ReST^{EM} w/ PRM	57.1%	56.7%	73.1%	81.4%	67.1%
	K-STaR	**59.6%**	**58.9%**	**77.5%**	**85.7%**	**70.4%**
Llama-3-8B-Instruct	w/o SFT	44.3%	40.1%	69.0%	78.0%	57.9%
	w/ SFT	15.5%	56.0%	72.0%	85.8%	57.3%
	STaR	50.2%	50.1%	73.0%	83.5%	64.2%
	RFT	51.3%	51.6%	71.5%	81.2%	63.9%
	RFT w/ PRM	57.2%	56.3%	73.8%	82.6%	67.5%
	ReST^{EM}	58.6%	57.1%	72.6%	83.1%	67.9%
	ReST^{EM} w/ PRM	60.6%	59.1%	75.6%	83.9%	69.8%
	K-STaR	**62.7%**	**61.3%**	**78.6%**	**86.7%**	**72.3%**

Table 2: Comparison between Mistral-7B-Instruct-v0.3 and Meta-Llama-3-8B-Instruct models using different self-improvement methods. All fine-tuned models are trained on a single training set from one dataset and evaluated on the corresponding test set across 4 mathematical and commonsense reasoning datasets.

Experiments

Experimental Setup

Datasets. We evaluate our approach on four widely used benchmarks covering both mathematical and commonsense reasoning tasks. (1) Mathematical Reasoning: We use GSM8K (Cobbe et al. 2021) and NumGLUE (Mishra et al. 2022). For GSM8K, we retain only the numerical answers as ground-truth labels and discard the annotated rationales. This setup allows us to isolate and measure the impact of generated reasoning paths. (2) Commonsense Reasoning: We adopt CommonsenseQA (Talmor et al. 2019) and StrategyQA (Geva et al. 2021). For all datasets, the training splits are used to generate synthetic reasoning paths and fine-tune the language models, while the test sets are reserved for performance evaluation.

Baselines. Our evaluation compares several baseline methods: (1) w/o SFT: We just report the performance of the base LLM using CoT prompting without further fine-tuning. (2) w/ SFT: We fine-tune the base LLM using the original instructions and ground-truth answers in the training sets, without incorporating any self-synthesized reasoning paths. (3) Self-improvement methods including STaR (Zelikman et al. 2022), RFT (Yuan et al. 2023) and ReST^{EM} (Singh et al. 2023). Additionally, we implement RFT and ReST^{EM} with two types of reward function: "without PRM" (w/o PRM) and "with PRM" (w/ PRM). We train a PRM using the idea in MATH-SHEPHERD (Wang et al. 2024) and pDPO (Jiao et al. 2024). See more details in Appendix B.

Training and Testing. Mistral-7B-Instruct-v0.3 (Jiang et al. 2023) and Meta-Llama-3-8B-Instruct (Grattafiori et al. 2024) are used. The process begins with reasoning path generation as previously described. Using a temperature of 0.8, we first generate $n = 5$ diverse knowledge unit sets for each instruction in the training set, followed by sampling $m = 10$ reasoning paths for each knowledge unit set. We then apply an exact match filter, discarding paths where the generated answer does not match the ground-truth. The remaining reasoning paths are ranked based on process reward estimation, and we select up to $p = 3$ high-quality paths per question. This results in a final training sample size of up to $3 \times |\mathcal{D}|$. Finally, the LLMs are fine-tuned with the selected reasoning paths using parameter-efficient LoRA (Hu et al. 2022). At inference, for a fair and accurate comparison, we use exact match accuracy on all experiments. We prompt the LLMs with: "*Solve the question step-by-step. Question:* " + {*question*} + "*Answer:*", using a temperature of 0.7. Further implementation details about our baseline methods and K-STaR are provided in Appendix C.

Main Results

We evaluate the enhancement of LLMs' reasoning capabilities using K-STaR, comparing it with existing self-improvement methods. As presented in Table 2, K-STaR significantly surpasses all baselines, achieving an average performance enhancement of 14.75% over the original model without fine-tuning. Baseline approaches, such as STaR, RFT and ReST^{EM}, which incorporate self-synthesized reasoning paths into their fine-tuning processes, also demonstrated improvements, albeit less significant than those achieved by K-STaR. This suggests that the quality or reliability of the reasoning paths generated by these models is inferior to those created by K-STaR. Specifically, Mistral and Llama achieve average scores of 70.4% and 72.3% with K-STaR, outperforming the best baseline ReST^{EM} w/ PRM by 3.3% and 2.5%, respectively.

Notably, the improvements are more substantial in commonsense reasoning tasks, where accurate and reliable knowledge are required. For instance, K-STaR achieves 77.5% on CommonsenseQA and 85.7% on StrategyQA with Mistral, setting a new benchmark in this category. The gains in math-

Dataset	w/o SFT	STaR	RFT w/ PRM	ReST^{EM} w/ PRM	K-STaR
GSM8K → CommonsenseQA	69.0% -	65.1% ↓	66.5% ↓	67.2% ↓	70.2% ↑
GSM8K → StrategyQA	78.0% -	74.8% ↓	76.0% ↓	76.8% ↓	78.3% ↑
NumGLUE → CommonsenseQA	69.0% -	65.5% ↓	67.0% ↓	67.8% ↓	69.6% ↑
NumGLUE → StrategyQA	78.0% -	75.5% ↓	76.3% ↓	77.0% ↓	77.7% ↓
CommonsenseQA → GSM8K	44.3% -	41.5% ↓	42.4% ↓	43.1% ↓	44.8% ↑
CommonsenseQA → NumGLUE	40.1% -	37.0% ↓	38.2% ↓	38.8% ↓	41.8% ↑
StrategyQA → GSM8K	44.3% -	41.9% ↓	42.8% ↓	43.5% ↓	44.9% ↑
StrategyQA → NumGLUE	40.1% -	37.5% ↓	38.3% ↓	38.9% ↓	40.5% ↑

Table 3: Comparison between out-of-domain performances using Meta-Llama-3-8B-Instruct model with different methods. All fine-tuned models are trained on a source training set from one dataset and evaluated on the target test set.

Method	GSM8K	NumGLUE	CommonsenseQA	StrategyQA	Average
w/o SFT	44.3%	40.1%	69.0%	78.0%	57.9%
K-STaR w/o PR	57.1%	55.8%	72.3%	81.4%	66.7%
K-STaR w/o OR	52.6%	49.5%	76.8%	80.2%	64.8%
K-STaR	62.7%	61.3%	78.6%	86.7%	72.3%

Table 4: Ablation results using Meta-Llama-3-8B-Instruct model with our K-STaR method on four datasets.

ematical reasoning tasks including GSM8K and NumGLUE are similarly significant, with K-STaR achieving 59.6% and 58.9% on Mistral, highlighting its capacity to effectively generate and utilize high-quality reasoning paths.

We also observe that models without SFT (w/o SFT) consistently lag behind all self-improvement methods, particularly in mathematical reasoning. This supports the intuition that LLMs require structured reasoning paths to perform well in tasks demanding step-by-step logical deductions. Furthermore, traditional SFT (w/ SFT) surprisingly underperforms on GSM8K, reinforcing the hypothesis that simply fine-tuning with ground-truth labels, without explicit reasoning steps, is insufficient for capturing the complexity of multi-step arithmetic reasoning. These results underscore the effectiveness of K-STaR in enhancing reasoning capabilities across both mathematical and commonsense tasks, establishing it as a state-of-the-art approach for self-improvement.

Out-of-Domain Results

To validate the robustness of K-STaR on out-of-domain (OOD) tasks, we conduct cross-domain evaluations between mathematical and commonsense reasoning tasks, as shown in Table 3. Experimental results indicate that baseline methods such as STaR and RFT w/ PRM generally experience a performance drop of 2∼4% points when transferred across domains. In contrast, K-STaR, leveraging its frequency and consistency-based knowledge unit selection, achieves an average improvement of 0.6% across six cross-domain settings, with notable gains of 1.7% in certain challenging transfers, such as from CommonsenseQA to NumGLUE.

This improvement underscores K-STaR's ability to capture domain-agnostic reasoning patterns through structured knowledge chains. Unlike baseline methods that rely primarily on answer correctness or domain-specific process rewards, K-STaR's reasoning paths are guided by knowledge-driven consistency, enabling it to generalize more effectively across divergent reasoning tasks. These findings highlight K-STaR's potential as a robust self-improvement framework for LLMs, particularly in settings where domain shifts are prevalent.

Ablation Studies

In this ablation study, we validate that each component of K-STaR. We compare two variants: K-STaR w/o PR and K-STaR w/o OR. K-STaR w/o PR only filters reasoning paths that lead to the correct answer (similar to STaR). K-STaR w/o OR filters reasoning paths solely based on the frequency of knowledge units, allowing for incorrect final answers. The ablation results are illustrated in Table 4.

Limitations of Outcome-Only Reward. Outcome-only reward achieves 57.1% on GSM8K and 55.8% on NumGLUE, significantly outperforming baseline models (+12.8% and +15.7%, respectively). However, due to its retention of paths that are "correct in answer but flawed in logic", its performance still lags behind the full K-STaR model (-5.6% and -5.5%). In CommonsenseQA, outcome-only reward reaches 72.3% (+3.3%), yet the absence of knowledge unit verification results in a noticeable gap compared to K-STaR (-6.3%).

Advantages of Process-Only Reward. Process-only reward achieves 76.8% on CommonsenseQA (+7.8%), approaching K-STaR performance. This demonstrates that knowledge unit filtering significantly enhances logical consistency in commonsense reasoning. In GSM8K and NumGLUE, process-only reward improves to 52.6% and 49.5% (+8.3% and +9.4%, respectively). However, since it allows for incorrect answers, its performance remains lower than K-STaR (-10.1% and -11.8%).

Outcome-based filtering provides critical signals for mathematical reasoning tasks by preventing entirely incorrect paths. However, when used in isolation, it cannot guarantee logical coherence throughout the reasoning process. On the other hand, frequency and consistency-based verification of

Metric (%)	R0	R1	R2	R3	R4	R5
Unique K	92	82↓	75↓	68↓	65↓	62↓
Path Repetition	0	18↑	25↑	32↑	38↑	42↑
Accuracy	69.0	73.2↑	76.1↑	78.1↑	78.4↑	78.6↑

Table 5: Results of multi-round iterations using K-STaR.

knowledge units in process-only reward significantly boosts reliability in commonsense reasoning while mitigating logical contradictions in mathematical tasks. K-STaR synergistically combines these two mechanisms, achieving optimal performance across both mathematical and commonsense reasoning tasks. This validates the collaborative advantage of knowledge-aware self-improvement, where knowledge consistency and outcome correctness are jointly optimized.

Iterative Improvement

We illustrate the multi-round iterations to show the long-term effects of K-STaR on Llama, using the CommonsenseQA dataset. Starting with the base model (w/o SFT), we repeat our process for five iterations, measuring test accuracy, unique knowledge unit ratio (proportion of distinct knowledge units among total units), and path redundancy rate (semantic similarity > 0.7 based on Sentence-BERT (Reimers and Gurevych 2019) embeddings), as shown in Table 5.

Convergence of Accuracy and Knowledge Unit Solidification. The accuracy improves rapidly during the first three iterations (+9.1%), with only a marginal increase of 0.5% in the fourth and fifth iterations, indicating that the model increasingly relies on previously validated high-quality knowledge units. The ratio of unique knowledge units drops from 91% initially to 62% by the fifth iteration, reflecting a reduction in new knowledge exploration during later stages. This suggests the model gravitates towards high-frequency knowledge combinations.

Risks of Rising Path Redundancy. The path redundancy rate increases from 18% in the first iteration to 42% in the fifth, indicating a growing tendency for the model to reuse validated paths. This repetitive behavior could lead to overfitting, as evidenced by the stagnation in accuracy during the fifth iteration. In later iterations, the model's reliance on high-frequency knowledge combinations may limit its generalization ability to new problems.

Sampling Hyperparameters

We investigate how sensitive K-STaR's performance is to the two key sampling hyperparameters: the number of knowledge-unit sets n and the number of candidate reasoning paths per set m. We utilize Llama to evaluate on the four benchmarks. For each (n, m) pair, generate n knowledge sets and m paths each, then apply the usual outcome + process filtering (select top-3 paths). After fine-tuning, we report the average accuracy across the four datasets.

Diminishing Returns. As shown in Table 6, moving from $n = 1 \rightarrow 3$ and $m = 5 \rightarrow 10$ yields substantial gains (+3.9% and +0.9%), showing the value of diversity. Increasing further to $n = 7$ or $m = 15$ yields only marginal improvements ($\leq$0.2%), indicating that $n = 5$, $m = 10$ is a sweet spot—balancing performance and cost.

$n \downarrow \setminus m \rightarrow$	$m = 5$	$m = 10$	$m = 15$
$n = 1$	68.2%	69.1%	69.3%
$n = 3$	71.4%	72.1%	72.2%
$n = 5$	72.0%	72.3%	72.4%
$n = 7$	72.1%	72.2%	72.2%

Table 6: Sampling hyperparameters average results using Meta-Llama-3-8B-Instruct model across four datasets.

Method	GSM8K	NumGLUE	CSQA	StrategyQA	Average
RFT	49.1%	47.0%	72.3%	79.5%	62.0%
ReST$^{\text{EM}}$	50.2%	48.4%	73.1%	80.2%	63.0%
K-STaR	52.1%	50.5%	74.2%	81.0%	64.5%

Table 7: Few-shot results of filtering reasoning paths.

Robustness to Hyperparameter Choice. Even at the lowest setting ($n = 1$, $m = 5$), K-STaR still outperforms most baselines ($\sim$68.2% vs. 67.9% for ReST$^{\text{EM}}$), indicating the effectiveness of incorporating knowledge into reasoning processes. Performance remains within a tight $\pm$0.4% window around the peak for all combinations with $n \geq 3$ and $m \geq 10$, demonstrating stability.

Quality of Selected Reasoning Paths

To further demonstrate the superiority of K-STaR's knowledge-guided process supervision over external PRM, we perform a 5-shot prompting experiment on Llama in Table 7. From each dataset, we held out 50 examples and, for each method (RFT w/ PRM, ReST$^{\text{EM}}$ w/ PRM, K-STaR), generated and filtered reasoning chains to pick the top five exemplars per dataset. These were concatenated and used to answer 200 unseen questions in each domain with temperature 0.7 and greedy decoding. K-STaR's exemplars yielded the highest gains over the zero-shot baselines—improving GSM8K by +7.8 points, NumGLUE by +10.4 points, CommonsenseQA by +5.2 points, and StrategyQA by +3.0 points—while PRM-based methods achieved smaller improvements. This demonstrates that K-STaR's knowledge-grounded process reward more effectively filters for broadly instructive, human-like reasoning paths, resulting in stronger generalization when used as few-shot exemplars.

Conclusion

In this work, we propose K-STaR, a self-training framework that models reasoning as structured knowledge compositions and assigns process rewards via consistency and frequency, without relying on reward models or human step-level labels. By combining process and outcome supervision, K-STaR effectively filters spurious reasoning and generates high-quality exemplars, demonstrating the power of shifting self-improvement toward a knowledge-rigorous paradigm.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Bai, Y.; Kadavath, S.; Kundu, S.; Askell, A.; Kernion, J.; Jones, A.; Chen, A.; Goldie, A.; Mirhoseini, A.; McKinnon, C.; et al. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. In *Proceedings of NeurIPS*.

Chen, Z.; Deng, Y.; Yuan, H.; Ji, K.; and Gu, Q. 2024. Self-play fine-tuning converts weak language models to strong language models. In *Proceedings of ICML*.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Dong, H.; Xiong, W.; Goyal, D.; Zhang, Y.; Chow, W.; Pan, R.; Diao, S.; Zhang, J.; Shum, K.; and Zhang, T. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*.

Feng, X.; Wan, Z.; Wen, M.; McAleer, S. M.; Wen, Y.; Zhang, W.; and Wang, J. 2023. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*.

Geva, M.; Khashabi, D.; Segal, E.; Khot, T.; Roth, D.; and Berant, J. 2021. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. In *TACL*.

Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Vaughan, A.; et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Gulcehre, C.; Paine, T. L.; Srinivasan, S.; Konyushkova, K.; Weerts, L.; Sharma, A.; Siddhant, A.; Ahern, A.; Wang, M.; Gu, C.; et al. 2023. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*.

Gunasekar, S.; Zhang, Y.; Aneja, J.; Mendes, C. C. T.; Del Giorno, A.; Gopi, S.; Javaheripi, M.; Kauffmann, P.; de Rosa, G.; Saarikivi, O.; et al. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*.

Hosseini, A.; Yuan, X.; Malkin, N.; Courville, A.; Sordoni, A.; and Agarwal, R. 2024. V-star: Training verifiers for self-taught reasoners. In *Proceedings of COLM*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. 2022. Lora: Low-rank adaptation of large language models. In *Proceedings of ICLR*.

Huang, J.; Gu, S. S.; Hou, L.; Wu, Y.; Wang, X.; Yu, H.; and Han, J. 2023. Large language models can self-improve. In *Proceedings of EMNLP*.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; Casas, D. d. l.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; et al. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Jiao, F.; Qin, C.; Liu, Z.; Chen, N. F.; and Joty, S. 2024. Learning planning-based reasoning by trajectories collection and process reward synthesizing. In *Proceedings of EMNLP*.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022. Large language models are zero-shot reasoners. In *Proceedings of NeurIPS*.

Lanham, T.; Chen, A.; Radhakrishnan, A.; Steiner, B.; Denison, C.; Hernandez, D.; Li, D.; Durmus, E.; Hubinger, E.; Kernion, J.; et al. 2023. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*.

Li, G.; Wang, P.; and Ke, W. 2023. Revisiting large language models as zero-shot relation extractors. In *Findings of EMNLP*.

Lightman, H.; Kosaraju, V.; Burda, Y.; Edwards, H.; Baker, B.; Lee, T.; Leike, J.; Schulman, J.; Sutskever, I.; and Cobbe, K. 2024. Let's verify step by step. In *Proceedings of ICLR*.

Luo, H.; Sun, Q.; Xu, C.; Zhao, P.; Lou, J.; Tao, C.; Geng, X.; Lin, Q.; Chen, S.; and Zhang, D. 2025. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. In *Proceedings of ICLR*.

Mishra, S.; Mitra, A.; Varshney, N.; Sachdeva, B.; Clark, P.; Baral, C.; and Kalyan, A. 2022. NumGLUE: A suite of fundamental yet challenging mathematical reasoning tasks. In *Proceedings of ACL*.

Mitra, A.; Del Corro, L.; Mahajan, S.; Codas, A.; Simoes, C.; Agarwal, S.; Chen, X.; Razdaibiedina, A.; Jones, E.; Aggarwal, K.; et al. 2023. Orca 2: Teaching small language models how to reason. *arXiv preprint arXiv:2311.11045*.

Mukherjee, S.; Mitra, A.; Jawahar, G.; Agarwal, S.; Palangi, H.; and Awadallah, A. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

Pang, R. Y.; Yuan, W.; He, H.; Cho, K.; Sukhbaatar, S.; and Weston, J. 2024. Iterative reasoning preference optimization. In *Proceedings of NeurIPS*.

Peng, X.; Xia, C.; Yang, X.; Xiong, C.; Wu, C.-S.; and Xing, C. 2025. ReGenesis: LLMs can Grow into Reasoning Generalists via Self-Improvement. In *Proceedings of ICLR*.

Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C. D.; Ermon, S.; and Finn, C. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Proceedings of NeurIPS*.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of EMNLP*.

Singh, A.; Co-Reyes, J. D.; Agarwal, R.; Anand, A.; Patil, P.; Garcia, X.; Liu, P. J.; Harrison, J.; Lee, J.; Xu, K.; et al. 2023. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*.

Sun, Z.; Shen, Y.; Zhou, Q.; Zhang, H.; Chen, Z.; Cox, D.; Yang, Y.; and Gan, C. 2023. Principle-driven self-alignment of language models from scratch with minimal human supervision. In *Proceedings of NeurIPS*.

Sutton, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine learning*, 3: 9–44.

Talmor, A.; Herzig, J.; Lourie, N.; and Berant, J. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of NAACL*.

Taylor, R.; Kardas, M.; Cucurull, G.; Scialom, T.; Hartshorn, A.; Saravia, E.; Poulton, A.; Kerkez, V.; and Stojnic, R. 2022. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Wang, P.; Li, L.; Shao, Z.; Xu, R.; Dai, D.; Li, Y.; Chen, D.; Wu, Y.; and Sui, Z. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of ACL*.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2023. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of ICLR*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of NeurIPS*.

Xia, M.; Malladi, S.; Gururangan, S.; Arora, S.; and Chen, D. 2024. Less: Selecting influential data for targeted instruction tuning. In *Proceedings of ICML*.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Proceedings of NeurIPS*.

Yu, L.; Jiang, W.; Shi, H.; Yu, J.; Liu, Z.; Zhang, Y.; Kwok, J. T.; Li, Z.; Weller, A.; and Liu, W. 2024. Metamath: Bootstrap your own mathematical questions for large language models. In *Proceedings of ICLR*.

Yuan, W.; Pang, R. Y.; Cho, K.; Li, X.; Sukhbaatar, S.; Xu, J.; and Weston, J. 2024. Self-Rewarding Language Models. In *Proceedings of ICML*.

Yuan, Z.; Yuan, H.; Li, C.; Dong, G.; Lu, K.; Tan, C.; Zhou, C.; and Zhou, J. 2023. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*.

Yue, X.; Qu, X.; Zhang, G.; Fu, Y.; Huang, W.; Sun, H.; Su, Y.; and Chen, W. 2024. Mammoth: Building math generalist models through hybrid instruction tuning. In *Proceedings of ICLR*.

Zelikman, E.; Wu, Y.; Mu, J.; and Goodman, N. 2022. Star: Bootstrapping reasoning with reasoning. In *Proceedings of NeurIPS*.

Zhang, D.; Hu, Z.; Zhoubian, S.; Du, Z.; Yang, K.; Wang, Z.; Yue, Y.; Dong, Y.; and Tang, J. 2024a. Sciglm: Training scientific language models with self-reflective instruction annotation and tuning. *arXiv preprint arXiv:2401.07950*.

Zhang, D.; Zhoubian, S.; Hu, Z.; Yue, Y.; Dong, Y.; and Tang, J. 2024b. Rest-mcts*: Llm self-training via process reward guided tree search. In *Proceedings of NeurIPS*.

Zhou, C.; Liu, P.; Xu, P.; Iyer, S.; Sun, J.; Mao, Y.; Ma, X.; Efrat, A.; Yu, P.; Yu, L.; et al. 2023. Lima: Less is more for alignment. In *Proceedings of NeurIPS*.