Multi-Agent Evolve: LLM Self-Improve through Co-evolution

Yixing Chen[1*] **Yiding Wang**[1*] **Siqi Zhu**[1] **Weixian Xu**[1] **Haofei Yu**[1] **Tao Feng**[1]
Muhan Zhang[2] **Mostofa Patwary**[3] **Jiaxuan You**[1,3†]
[1]University of Illinois at Urbana-Champaign [2]Peking University [3]NVIDIA

Abstract

Reinforcement Learning (RL) has demonstrated significant potential in enhancing the capabilities of large language models (LLMs). However, the success of RL for LLMs heavily relies on human-curated datasets, which pose a burden on scalability and sustainability. Inspired by the success of the paradigm in games and Go, self-play methods are deployed to enhance LLM capabilities without human-annotated data. However, these methods primarily depend on a grounded environment for verifiable feedback (*e.g.*, a Python interpreter or a game engine); extending them to general domains remains challenging. To address these challenges, we propose **Multi-Agent Evolve (MAE)**, a framework in which LLM can self-evolve across diverse tasks from mathematics to general knowledge. MAE implements asymmetric self-play by leveraging a triplet of interacting roles (*Proposer*, *Solver*, *Judge*) that are instantiated from a single LLM. *Proposer* generates challenging questions, *Solver* optimizes solutions, and *Judge* provides reward signals. These roles are jointly optimized via reinforcement learning, creating a closed-loop system where the model continuously challenges and improves itself. Experiments on Qwen2.5-3B-Instruct demonstrate that MAE achieves an average improvement of 4.86% across multiple benchmarks, significantly outperforming baselines. These results highlight MAE as a scalable, data-efficient method for enhancing LLMs' general reasoning abilities with minimal reliance on human supervision.

1 Introduction

Reinforcement Learning (RL) (Kaelbling et al., 1996; Silver et al., 2014) has demonstrated substantial potential in training Large Language Models (LLMs), leading to notable improvements in tasks such as coding and reasoning (Guo et al., 2025a). However, these successes rely heavily on human-curated datasets that can provide verifiable rewards (Shao et al., 2024). Human-curated datasets are costly and limited in quantity, raising concerns about their scalability. Moreover, since LLMs are expected to advance beyond human-level intelligence (Hunt, 2010), they will likely require training signals that surpass the capacity of human curation. The above issues lead to the focus of this paper: can we build an effective **closed-loop** RL framework for LLMs to **self-improve** in general ability?

Self-play (DiGiovanni and Zell, 2021) has long served as a paradigm for self-improvement in reinforcement learning, enabling models to discover progressively stronger strategies in symmetric zero-sum environments like Go, Chess, and Dota 2 (OpenAI et al., 2019; Silver et al., 2017). While recent studies have adapted this mechanism to Large Language Models in simple games (Liu et al., 2026) or through asymmetric Proposer-Solver adversarial interaction (Zhao et al., 2026), they predominantly rely on verifiable environments (*e.g.*, code compilers or game engines) to provide ground-truth feedback. These external verifiers serve as an "anchor", ensuring the model's exploration remains grounded in objective correctness. However, the vast majority of human intelligence tasks (Hunt, 2010), ranging from creative writing (Gómez-Rodríguez and Williams, 2023) to complex commonsense (Guo et al., 2025b), lack such programmatic or deterministic gold standards. In other words, deterministic reward functions are unattainable for most real-world tasks, creating a bottleneck that prevents previous self-play paradigms from scaling toward general artificial intelligence (Hendrycks et al., 2025). As a replacement for hard-coded verifiers, we aim to use model-based judgments (Gu et al., 2026) while maintaining the stability of the

[*] Equal Contribution.
[†] Corresponding Author.

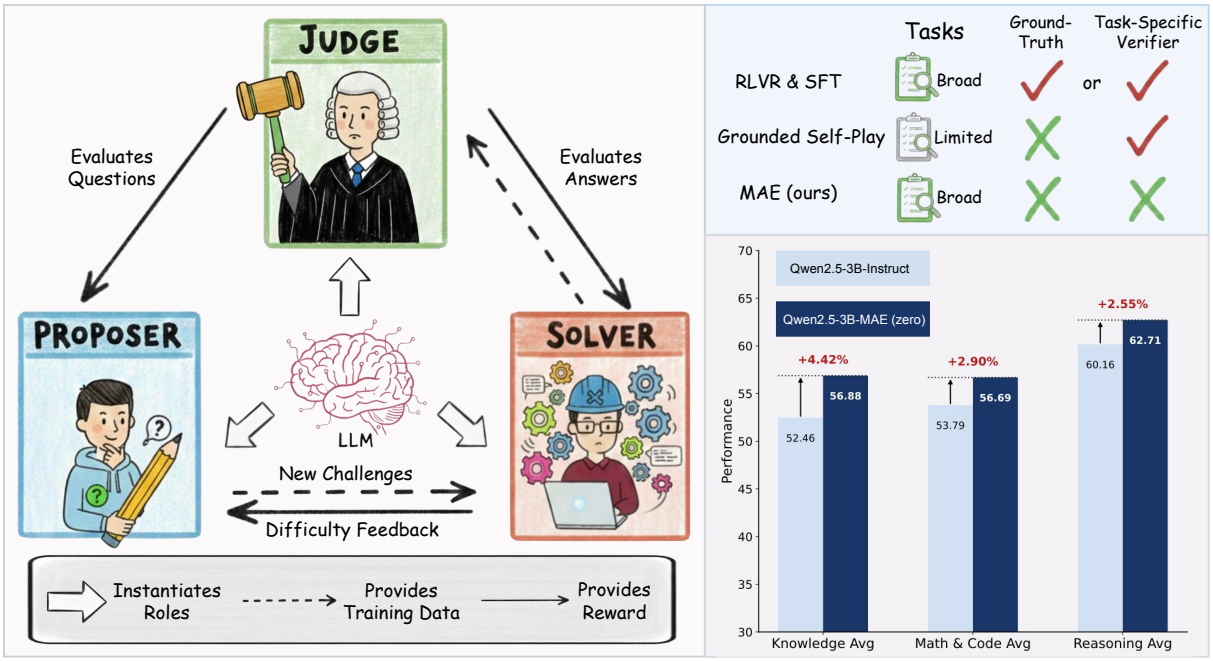


Figure 1: **Overview of Multi-Agent Evolve (MAE). (Top Right)** Comparison with existing training paradigms: unlike Reinforcement Learning with Verifiable Rewards (RLVR) and SFT which rely on expensive human data or task-specific verifiers, and grounded self-play RL which is limited to tasks with ground-truth simulators, MAE scales to broad tasks without external supervision. **(Left)** The MAE framework instantiates three interactive roles (*Proposer*, *Solver*, and *Judge*) from a single LLM to form a closed self-improving loop. The Proposer generates new challenges, the Solver attempts to answer them, and the Judge evaluates the output. This creates an adversarial co-evolution process where the Proposer is incentivized by *difficulty feedback* to create harder questions as the Solver improves. **(Bottom Right)** MAE (zero) enhances the performance of Qwen2.5-3B-Instruct across Knowledge QA, Math & Code, and Reasoning benchmarks with no human supervision.

self-improvement loop.

To this end, we propose **Multi-Agent Evolve (MAE)**, a self-contained, multi-role framework that extends the previous self-play paradigm to general domains. Recognizing the lack of grounded constraint, Multi-Agent Evolve instantiates three interactive roles (*Proposer*, *Solver*, and *Judge*) from a single base LLM to form a self-evolving loop (Figure 1). In this loop, the Proposer and Solver engage in a constrained adversarial interaction: the Solver is rewarded for correctness, while the Proposer is optimized with a dual objective consisting of a *difficulty reward* based on the Solver's performance and a *quality reward* enforced by the Judge. This dual signal serves as the necessary regularizer for asymmetric roles, ensuring that the Proposer drives evolution by creating tasks that are challenging yet valid. To maintain stability and scalability, MAE incorporates rule-based *format rewards* and *question quality filtering* based on the Judge's evaluations, preventing data corruption and ensuring robust self-evolution.

Experimental results demonstrate the effectiveness of Multi-Agent Evolve. As shown in Figure 1, even in a Zero (Zhao et al., 2026) setting without human-curated data, our method achieves substantial performance gains across diverse benchmarks, including General QA, Math & Coding, and complex Reasoning. Notably, MAE also outperforms strong grounded self-play baseline Absolute Zero Reasoner (AZR) (Zhao et al., 2026). Performance is further enhanced when the model is seeded with a small set of unlabeled reference questions, achieving an average improvement of 4.86% over the base model. Crucially, MAE outperforms standard Supervised Fine-Tuning (SFT) trained on the same seed dataset with ground-truth labels, demonstrating that our multi-role interaction can more effectively unlock the model's capabilities than static SFT training.

In summary, our main contributions are as follows:

- We introduce **Multi-Agent Evolve (MAE)**, a **closed-loop, self-evolve** framework that instantiates three interactive roles (*Proposer*, *Solver*, and *Judge*) from a single base LLM and jointly trains them via reinforcement learning. The framework can operate on simple, unlabeled data or with no human-curated

data.

- We empirically demonstrate the effectiveness and scalability of Multi-Agent Evolve on **Qwen2.5-3B-Instruct** (Qwen et al., 2025) across mathematics, coding, reasoning, and general knowledge benchmarks, achieving improvements over both base and supervised fine-tuned models.

- We provide key insights into LLM self-improvement via thorough analysis of the training process. We confirm the contributions of each designed component and observe intriguing dynamics, such as entropy decay leading to Proposer convergence and the desirable difficulty effect.

2 Related Work

LLM-as-a-judge The paradigm of "LLM-as-a-judge" (Gu et al., 2026) leverages the advanced capabilities of Large Language Models (LLMs) to perform evaluation tasks such as scoring, ranking, or selection, which have traditionally relied on costly human annotation (Li et al., 2025). This approach has been widely adopted across the LLM life cycle for applications such as evaluation, alignment, retrieval, and reasoning. Although it has some limitations, it has become a promising approach to provide self-training signals. Early prompting-based methods (Raman et al., 2024; Weng et al., 2023) utilize LLMs themselves to provide generative feedback and refine their answers. Pang et al. (2024) improve the accuracy of reasoning task answers through teacher-student self-play. Yuan et al. (2025) use Self-Rewarding Language Models to iteratively train their instruction-following capability given the self-rated scores. In our paper, the LLM-as-a-judge paradigm is integrated into the system and used to train the model.

Self-Play for LLM Self-Play (DiGiovanni and Zell, 2021) is a data-free approach that requires minimal human supervision. It primarily relies on the system's capabilities and its self-interaction to enable higher intelligence to emerge. Self-play settings often involve only two players, and their interactions are somewhat limited to a zero-sum format. Recent works have deployed self-play on different LLM settings to improve corresponding capabilities (Ye et al., 2025; Fang et al., 2026; Huang et al., 2026; Zhou et al., 2026; Lin et al., 2025). SelfRedTeam (Liu et al., 2025) conducts attacker and defender interactions in a game of LLM safety to produce a safer model. Absolute Zero (Zhao et al., 2026) uses self-play in RLVR to enhance performance in coding and mathematics, which relies on a verifiable environment. SPIRAL (Liu et al., 2026) conducts self-play under zero-sum game settings to enable a broader range of tasks. Multi-Agent Evolve distinguishes itself by utilizing the self-play paradigm while introducing a Judge for evaluation to make zero-sum no longer compulsory, thereby making even more general domain tasks applicable.

Multi-Agent for LLM LLMs have exhibited intelligence that is capable of all kinds of tasks. Therefore, they have been specialized as agents to cope with different situations (Wang et al., 2025), and their interactions show great potential (Liang et al., 2024; Talebirad and Nadiri, 2023; Chen et al., 2024; Bettini et al., 2024). However, previous works often directly deploy multi-agent systems and apply no training to these agents, which limits the adaptive evolution of these agents. Despite the promising effects of Multi-Agent for LLM, it presents various challenges in implementation, cost, and stability (Cemri et al., 2026). Motwani et al. (2025) deploy Multi-Agent LLM training with a voting mechanism for reasoning enhancement. Zeng et al. (2024) employ Multi-Agent LLM on a simpler setting of attack and defense of harmful responses. Our work presents a framework for general tasks that involves the interaction of three roles instantiated from a single base model using Task-Relative REINFORCE++ (Zhao et al., 2026), which applies RL training to each role.

3 Preliminaries

3.1 Problem Formulation: Asymmetric Self-Play

We formulate the self-evolution of LLMs as an **asymmetric self-play** process within a general-sum game setting. Unlike classic self-play in symmetric zero-sum games (e.g., Go) where agents π_1, π_2 optimize opposing objectives to reach a Nash Equilibrium (Silver et al., 2017):

$$\max_{\pi_1} \min_{\pi_2} \mathbb{E}_{s\sim\mathcal{E}}[R(s, \pi_1, \pi_2)] \tag{1}$$

Open-ended reasoning tasks require agents to adhere to linguistic and logical validity, rendering

strict zero-sum objectives prone to degenerate behaviors (e.g., generating adversarial noise) (Liu et al., 2026).

To address this, we define the interaction between a *Proposer* (π_P) and a *Solver* (π_S) as a **cooperative-competitive** game. While the agents compete on task difficulty, they must cooperate to maintain task quality. Formally, the Proposer optimizes a composite objective $J(\pi_P)$:

$$J(\pi_P) = \mathbb{E}_{q\sim\pi_P, a\sim\pi_S}\left[\underbrace{\lambda_D R_{\text{D}}(q,a)}_{\text{Adversarial}} + \underbrace{\lambda_Q R_{\text{Q}}(q)}_{\text{Cooperative}}\right] \quad (2)$$

where R_{D} incentivizes exploring the frontier of the Solver's capabilities, and R_{Q} imposes a validity constraint parameterized by an evaluator. Our **MAE** framework (Section 4) instantiates this formulation by introducing a **Judge** agent to explicitly approximate R_{Q}.

Task-Relative REINFORCE++ Multi-agent training requires applying reinforcement learning to each role. We follow Absolute Zero Reasoner (Zhao et al., 2026) and use Task-Relative REINFORCE++, which computes separate baselines for each agent. This is an interpolation between per-question algorithms such as Group Relative Policy Optimization (GRPO) (Shao et al., 2024) and single-baseline algorithms such as REINFORCE++ (Hu et al., 2025), enabling variance reduction for each agent's task type. We denote μ_{role} and σ_{role} as the average and standard deviation of the reward for each role. The normalized advantage A^{norm} for each role is computed as:

$$A^{\text{norm}}_{\text{role}} = \frac{r - \mu_{\text{role}}}{\sigma_{\text{role}}} \quad (3)$$

Subsequently, following Proximal Policy Optimization (PPO) (Schulman et al., 2017), the policy π_θ is updated by maximizing the following objective:

$$\mathcal{J}(\theta) = \mathbb{E}\Big[\min\left(\rho_t A^{\text{norm}}_{\text{role}}, \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon) A^{\text{norm}}_{\text{role}}\right)\Big] \quad (4)$$

where $\rho_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)}$ denotes the importance sampling ratio, ϵ controls the clipping ratio. Appendix A.3 provides a local surrogate-optimization view of the synchronized shared-parameter update and discusses its limitations.

4 Multi-Agent Evolve

In this section, we introduce our Multi-Agent Evolve (MAE) framework, in which a Large Language Model is instantiated into three functional roles (Proposer, Solver, and Judge) that share parameters θ but have role-specific system instructions. We detail the roles and their reward configurations below, and defer implementation-level prompts and reward details to Appendix B. We additionally provide a focused discussion of why we use a shared-parameter backbone and how this design affects training stability in Appendix A.1, with supporting gradient-alignment analysis in Appendix A.2.

4.1 Proposer: Curriculum Design via Generative Exploration

For the model to continuously improve, it requires a stream of training tasks that lie on the frontier of its current capabilities. The Proposer (π_P) assumes this responsibility, conducting adaptive curriculum design by generating questions and engaging in adversarial interaction with the Solver.

However, generalizing the typical Proposer-Solver interaction to a general domain exposes a vulnerability inherent in the asymmetry between the two roles: a Proposer that optimizes purely for difficulty may degenerate into generating invalid or nonsensical questions to trivially defeat the Solver. To regulate this asymmetry, we define the Proposer's objective as dual: (1) **Quality**: The generated questions must be well-formed and logical; (2) **Difficulty**: The questions must present a challenge to the current Solver.

At each step, the Proposer generates a new question q. This is done either originally or by modifying a reference question q_{ref} (without its ground truth) to alter its complexity:

$$q \sim \pi_P(\cdot|(q_{ref}); \theta) \quad (5)$$

The Proposer's optimization is driven by a composite reward function R_P, consisting of a quality reward R_Q, a difficulty reward R_D, and a format reward R_F. The Judge assigns the **quality reward** R_Q based on the question's clarity and solvability. To quantify the difficulty of the proposed question, we leverage the Solver's empirical performance as a proxy. Specifically, for a generated question q, we sample N responses from the Solver; the **difficulty reward** R_D is calculated as the complement of the normalized average Solver score.

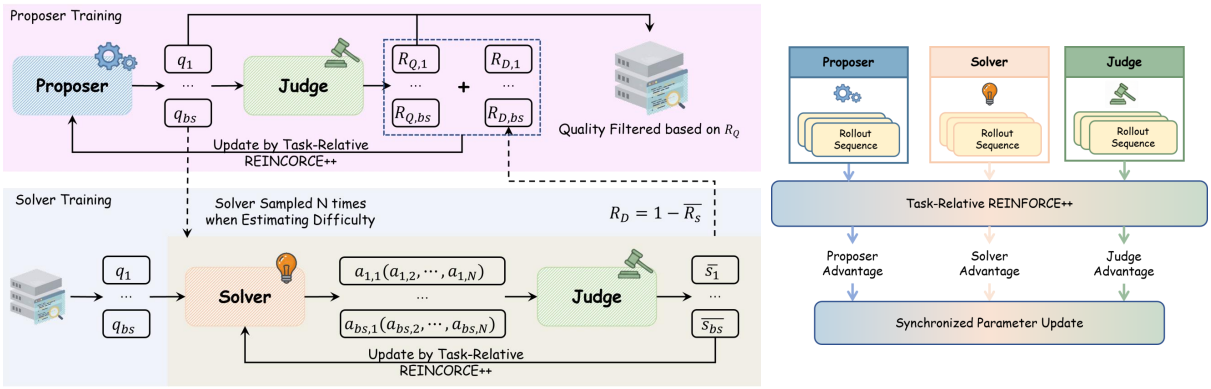


Figure 2: **Multi-Agent Evolve Framework:** (Left) In Multi-Agent Evolve, Proposer generates questions and receive reward based on Judge evaluation and difficulty estimation. After applying quality filter on the generated questions, only a portion of the questions enter the maintained dataset for future usage. For the Solver, it receives the Judge's evaluation as well. (Right) Our multi-agent training employs Task-Relative REINFORCE++, which computes the advantage for each role separately and then performs a synchronized parameter update to the uniform model.

$$\bar{R}_S(q) = \frac{1}{N}\sum_{i=1}^{N} V_J(a_i, q) \quad \text{where } a_i \sim \pi_S(\cdot|q) \tag{6}$$

$$R_D(q) = 1 - \bar{R}_S(q) \tag{7}$$

where $V_J(a_i, q)$ is the Judge's evaluation of the Solver's response a_i. Intuitively, this reward design incentivizes the Proposer to find valid, challenging questions for the Solver's current policy. To ensure the structural validity of outputs, a format reward R_F is applied to all roles for adherence to the requested tags. Therefore, the total reward for the Proposer is:

$$R_P(q) = \lambda_Q R_Q + \lambda_D R_D + \lambda_F R_F \tag{8}$$

4.2 Solver: Trajectory Rollout and Policy Execution

The Solver (π_S) is the primary policy intended for deployment. It is tasked with generating a solution trajectory a given a question q sampled from the dataset:

$$a \sim \pi_S(\cdot|q; \theta) \tag{9}$$

Since our framework targets general domains where newly generated questions lack verifiable ground-truth labels, the Solver relies on the Judge's score $R_J = V_J(a, q)$ as the reward for the pair (q, a). Therefore, the Solver's optimization objective is to maximize the expected return of:

$$R_S(a) = \lambda_J R_J + \lambda_F R_F \tag{10}$$

By maximizing R_J, the Solver improves its reasoning capabilities, which in turn forces the Proposer to generate even more sophisticated tasks to maintain the adversarial difficulty.

4.3 Judge: Intrinsic Reward Modeling

The Judge (π_J) operates as a generative reward model, providing the scalar feedback that guides the training of both the Proposer and the Solver. To ensure evaluations are well-reasoned, the Judge employs a Chain-of-Thought (CoT) process that generates a detailed analysis before issuing the final score. The Judge operates under strict rubrics **without access to ground truth**.

The Judge is constrained by the format reward (R_F) to ensure it adheres to the parsing structure required for score extraction. Its evaluative precision is implicitly enhanced through parameter sharing: as the Solver's reasoning capabilities improve via RL, the shared backbone's ability to distinguish sound from flawed logic co-evolves.

4.4 Pipeline

We organize the training process into iterative rounds. A single training step in our framework includes the following synchronized phases, as illustrated in Figure 2.

Proposer Phase. The Proposer generates a batch of candidate questions that are either original or based on sampled reference questions. We apply **question quality filtering** here to admit only feasible and informative generated content into the dataset, excluding those with Judge evaluation scores below a specified threshold.

Solver Phase. The Solver samples from the valid dataset. It generates a response for each question and receives scores from the Judge. All question-answer pairs are then added to the pair dataset for the Judge to use.

Judge Phase. The Judge samples from the pair

dataset and receives a format reward for its own output structure.

Synchronized Update. Finally, we update the shared LLM backbone θ using the gradients accumulated from all three roles.

The detailed pseudo-code for the entire training workflow is provided in Algorithm 1 in Appendix C.

5 Experiments

In this section, we evaluate the effectiveness of our Multi-Agent Evolve framework.

Settings: We use Qwen2.5-3B-Instruct (Qwen et al., 2025) as the base model and evaluate over four data settings. **MAE (zero)** matches AZR (Zhao et al., 2026) by using no real-world data or ground truth: it starts from 16 self-generated seed questions and then samples either a current reference from $\mathcal{D}$ or no reference. The other settings initialize $\mathcal{D}$ with 967 unlabeled seed questions from 14 math, coding, and commonsense datasets. Appendix F gives the full breakdown and the other three settings are detailed below.

- **MAE (no reference):** The Proposer initializes its valid question set $\mathcal{D}$ with the seed data and generates every new question from scratch.
- **MAE (half reference):** The Proposer also initializes its question set with the seed data, but has an equal chance of generating a new question from a reference (sampled from $\mathcal{D}$) or from scratch.
- **MAE (with reference):** The Proposer initializes its valid question set $\mathcal{D}$ with the seed data and generates every new question based on a randomly sampled reference question from $\mathcal{D}$.

We evaluate on in-distribution benchmarks aligned with the seed data and held-out out-of-distribution benchmarks. Coding tasks use EvalPlus (Liu et al., 2023); all other tasks are scored by a strong LLM judge that compares tagged answers with ground truth and returns TRUE/FALSE judgments, see Appendix D for implementation details and Appendix D.6 for reliability analysis.

Baselines: 'Base' is the initial model; SFT uses seed questions with ground-truth answers and LoRA (Hu et al., 2022) training for 5 epochs; AZR uses its official implementation for 100 steps with matched hyperparameters.

5.1 Results and Findings

As shown in Tables 1 and 2, MAE demonstrates that **effective co-evolution can emerge with minimal human-curated data**. Starting from only 16 self-generated seed questions, MAE (zero) improves the overall average from 55.33 to 58.67 and achieves gains across all four categories, showing that co-evolution can improve broad capabilities without real-world questions or ground-truth answers.

With unlabeled seed questions, MAE (no reference) and MAE (half reference) achieve the strongest overall results (60.19 and 59.87), outperforming SFT (57.92) and MAE (with reference) (57.11). This shows that **questions generated by the model individually are important**: always conditioning on references may restrict exploration and weaken the self-curriculum.

Gains are strongest in **Knowledge and Reasoning**: MAE (no reference) improves them by 7.00 and 6.48 points, while MAE (half reference) improves them by 5.86 and 6.53 points. Gains in Math and Code are smaller, with SFT remaining strongest in Code, suggesting that verifiable domains may benefit more from explicit rewards.

5.2 Training Dynamics

Previous approaches that use LLM interactions often suffer from instability and collapse during training (Zhao et al., 2026; Huang et al., 2026). To further investigate the effectiveness and weaknesses of our framework, we analyze the training dynamics in the remainder of this section.

5.2.1 Training sustainability

Models trained with our framework achieve improved performance for more than 150 steps with a batch size of 64. As shown in Figure 3, our framework consistently contributes questions to the dataset at each training step, indicating that question quality remains high. The sustainability of our framework depends on the diversity of interactions among roles. Given the interdependent nature of the roles, ensuring behavioral diversity within each agent is essential. For example, suboptimal prompts and settings may lead the Pro-

Table 1: **MAE can self-evolve in general domains without verifiable rewards**. For fair comparison, our experimental results are presented in two parts that differ in the use of seed data. In w/o reference questions setting, MAE offers an improvement on almost all benchmarks across all domains, surpassing AZR. Improvements are further enhanced when some reference questions are provided **without directly using their ground truth**, the MAE (no reference) achieves the highest overall average accuracy. We highlight in-distribution (ID) benchmarks in blue and held-out (OOD) benchmarks in green (for models without reference questions, all benchmarks are considered OOD). Results shown in **bold** font are the best results of the benchmark in its own seed data settings. The **bold underlined** results are the best results across all settings.

Method	GSM8K	MATH	ARC-C	MMLU	GPQA	CQA	OBQA	NQ	TriviaQA	SQuAD	BoolQ	HellaSwag	MBPP+	HumanEval+
w/o reference questions.														
Base	85.20	60.40	80.60	**63.40**	34.67	66.80	67.80	27.80	51.60	78.20	73.60	67.80	60.60	68.30
AZR	81.20	62.40	82.80	**63.40**	**38.93**	**74.36**	77.55	35.33	56.67	90.85	**78.50**	67.56	61.40	68.30
MAE (zero)	**86.00**	**68.20**	**84.20**	61.40	29.42	71.54	**79.39**	**37.98**	**60.13**	**92.28**	78.46	**70.34**	**62.20**	**69.50**
w/ reference questions.														
SFT	82.40	63.20	81.60	63.80	28.97	72.00	74.72	34.38	57.31	91.74	79.70	67.23	**67.70**	**74.40**
MAE (with reference)	76.00	49.00	82.80	62.40	32.60	75.58	76.11	34.20	57.13	90.73	**83.45**	69.62	61.00	60.30
MAE (no reference)	**84.40**	**68.20**	**86.00**	67.40	**34.23**	74.93	**83.66**	**38.01**	**63.68**	93.10	77.50	71.54	61.40	68.30
MAE (half reference)	82.20	65.80	83.20	**69.00**	30.80	**77.20**	80.00	36.40	61.00	**93.40**	78.00	**79.00**	61.10	68.30

Method	TruthfulQA	BBH	LiveBench Reasoning	AMC	Minerva	WinoGrande	Olympiad	MMLU-Pro	ID Avg.	OOD Avg.	Overall Avg.
w/o reference questions.											
Base	45.71	53.79	**20.80**	39.76	34.52	63.53	27.73	**44.68**	–	55.33	55.33
AZR	44.92	52.57	19.27	34.94	**41.22**	**64.65**	**28.94**	44.09	–	57.72	57.72
MAE (zero)	**52.71**	**57.51**	16.48	**44.58**	39.91	59.48	26.47	42.66	–	**58.67**	**58.67**
w/ reference questions.											
SFT	46.25	55.58	23.43	**37.35**	36.65	**64.20**	28.42	43.24	67.08	41.89	57.92
MAE (with reference)	53.86	57.17	26.78	34.94	38.80	62.83	26.67	44.35	65.07	43.18	57.11
MAE (no reference)	**55.46**	**57.53**	28.84	33.74	**40.78**	63.96	25.27	46.32	**69.45**	**43.99**	**60.19**
MAE (half reference)	50.55	53.14	**33.61**	35.54	38.26	60.71	**32.46**	**47.43**	68.95	43.96	59.87

Table 2: Category-level results across 22 benchmarks. **Math** includes GSM8K, MATH, AMC, Minerva, and Olympiad; **Code** includes MBPP+ and HumanEval+; **Knowledge & Factual QA** includes MMLU, MMLU-Pro, GPQA, NQ, TriviaQA, TruthfulQA, BoolQ, and SQuAD; and **Reasoning & Commonsense** includes ARC-C, CQA, OBQA, HellaSwag, BBH, LiveBench, and WinoGrande. Parentheses report absolute changes relative to the base model. The best result in each category is shown in **bold**.

Method	Math (5)	Code (2)	Knowledge (8)	Reasoning (7)	Overall (22)
Base	49.52	64.45	52.46	60.16	55.33
MAE (zero)	**53.03** (+**3.51**)	65.85 (+1.40)	56.88 (+4.42)	62.71 (+2.55)	58.67 (+3.34)
SFT	49.60 (+0.08)	**71.05** (+**6.60**)	55.67 (+3.21)	62.68 (+2.52)	57.92 (+2.59)
MAE (with reference)	45.08 (−4.44)	60.65 (−3.80)	57.34 (+4.88)	64.41 (+4.25)	57.11 (+1.78)
MAE (no reference)	50.48 (+0.96)	64.85 (+0.40)	**59.46** (+**7.00**)	66.64 (+6.48)	**60.19** (+**4.86**)
MAE (half reference)	50.85 (+1.33)	64.70 (+0.25)	58.32 (+5.86)	**66.69** (+**6.53**)	59.87 (+4.54)

poser to propose only open-ended writing questions. This, in turn, causes the model to perform worse on most benchmarks due to interactions between the agents.

Nevertheless, prolonged training remains susceptible to collapse, primarily driven by entropy decay of the Proposer role. Unlike previous methods, the collapse in MAE is characterized by the Proposer converging toward a narrow distribution of similar or identical questions. This reduces the diversity of the question distribution and fails to drive further improvement.

5.2.2 Desirable difficulty offers improvement

Through careful investigation of the training curves, we find that generating feasible yet difficult questions is key to improved performance. As shown in Figure 3, we observe a rise in benchmark accuracy soon after periods during which the Proposer successfully increases question difficulty. While the Proposer does not always achieve this increase in difficulty, its presence always leads to greater performance improvements. The relationship between difficulty and model performance suggests that hard questions, which present challenges to the model, help push its ability boundary further. This effect is intuitively consistent with the Desirable Difficulty Effect, which holds that, for optimal long-term learning, tasks should be challenging but not overwhelming. This dynamic justifies our multi-role design: the Proposer continually escalates difficulty to challenge the Solver,

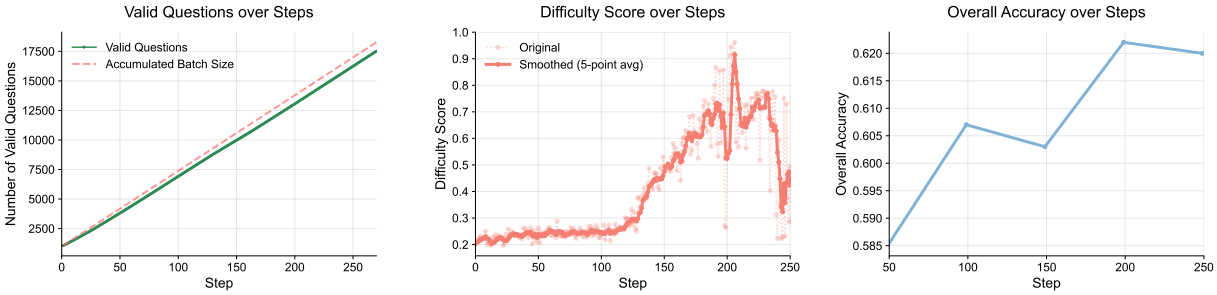


Figure 3: **Training Process Analysis:** These three figures demonstrate an example training process. (Left) The number of questions in the dataset increases steadily while low-quality questions are excluded. (Mid and Right) The Proposer learns to generate questions that present a desirable level of difficulty to the Solver, thereby benefiting the model in future training.

while the Judge ensures solvability, effectively automating a curriculum that maximizes learning efficiency.

5.3 Ablation

To investigate the individual contributions of the components in Multi-Agent Evolve, we conduct a comprehensive ablation study on the Qwen2.5-3B-Instruct (Qwen et al., 2025) model.

5.3.1 Agent Roles

A defining characteristic of Multi-Agent Evolve is the interaction among three distinct roles and their co-evolution. To demonstrate that each role matters, we disable training for each role and observe the corresponding performance. As shown in Table 3, disabling training for any single role leads to a degradation in performance, with drops of 1.99% for the Solver, 1.97% for the Proposer, and 2.63% for the Judge, respectively.

These results underscore the importance of co-evolution. Although disabling training for each role does not affect overall stability, it does limit the model's potential for higher performance. Each role plays its part in the MAE framework, and training for each role is necessary for optimal outcomes in our setting.

5.3.2 Shared vs. Separate Role Parameters

Multi-Agent Evolve instantiates all three roles from a shared backbone, with role-specific prompts inducing distinct Proposer, Solver, and Judge behaviors. This design allows improvements learned through one role to transfer directly to the others, while avoiding the memory and checkpoint-management cost of maintaining three independent models. To examine whether parameter sharing limits role specialization, we additionally train a variant with separate parameters for the three roles under the same configurations. Table 4 summarizes the comparison using category-level averages. Detailed motivation and gradient-level analysis are provided in Appendices A.1 and A.2.

Table 4: Comparison of shared and separate role parameters using the category partition in Table 2. The shared-backbone results use the MAE (half reference) setting.

Parameterization	Math	Code	Knowledge	Reasoning	Overall Avg.
Separate role parameters	**55.37**	**65.45**	57.62	62.17	59.27
Shared backbone	50.85	64.70	**58.32**	**66.69**	**59.87**

Overall, the shared backbone slightly outperforms separate role parameters (59.87 vs. 59.27), indicating that parameter sharing does not introduce a performance trade-off in our setting and may even benefit cross-role transfer. It is also substantially more efficient, as it avoids maintaining three independent models and reduces the associated memory, storage, and checkpoint-management overhead.

5.3.3 Scalability to a Larger Backbone

We further evaluate whether the benefits of Multi-Agent Evolve persist when the backbone is scaled to Qwen2.5-7B-Instruct. We apply the same framework and evaluation protocol, and report category-level averages to provide a compact view of the performance. Table 5 compares the 7B base model with its MAE-trained counterpart under the half reference setting.

Table 5: Scalability results on Qwen2.5-7B-Instruct. Results are averaged within four benchmark categories.

Method	General QA	Math	Coding	Reasoning	Overall Avg.
Base	58.37	55.43	**71.75**	**70.03**	62.63
MAE (half reference)	**59.87**	**58.13**	71.55	69.49	**63.59**

Table 3: **Ablation study:** Experiments for our ablation study are carried out in the 'half reference' setting. Each role in our framework is essential, and excluding any one of them will result in decreased performance. Format reward and question quality filtering both ensure the stability of the training process and prevent dataset corruption. Question quality filtering alone also shows a similar effect in constraining format. The **bold underlined** results are the best results across all settings.

Method	GSM8K	MATH	ARC-C	MMLU	GPQA	CQA	OBQA	NQ	TriviaQA	SQuAD	BoolQ	HellaSwag	MBPP+	HumanEval+
MAE (half reference)	82.20	65.80	83.20	**69.00**	30.80	**77.20**	80.00	36.40	61.00	**93.40**	78.00	**79.00**	61.10	68.30
agent roles														
MAE (no solver training)	86.20	63.20	**84.60**	64.60	31.41	74.70	75.85	35.24	60.15	90.52	78.64	69.36	**62.70**	62.60
MAE (no proposer training)	84.20	64.20	80.60	62.60	30.85	75.03	76.67	35.86	57.31	92.20	79.50	67.92	60.00	**69.50**
MAE (no judge training)	85.20	64.40	81.00	62.40	30.90	75.18	76.83	36.06	57.98	92.02	**79.86**	68.26	61.40	68.30
question quality filtering and format reward														
MAE (no question quality filtering)	85.60	**67.40**	81.40	60.60	**32.88**	76.26	78.78	36.31	**62.34**	90.71	78.33	70.62	61.40	68.30
MAE (no format reward)	**86.40**	66.40	83.40	64.80	30.23	75.00	**80.52**	**39.26**	61.61	93.38	76.39	68.28	61.40	68.30

Method	TruthfulQA	BBH	LiveBench Reasoning	AMC	Minerva	WinoGrande	Olympiad	MMLU-Pro	ID Avg.	OOD Avg.	Overall Avg.
MAE (half reference)	50.55	53.14	**33.61**	35.54	38.26	60.71	**32.46**	**47.43**	**68.95**	43.96	**59.87**
agent roles											
MAE (no solver training)	48.80	**56.60**	20.00	37.35	37.78	63.00	27.58	42.51	67.13	41.70	57.88
MAE (no proposer training)	47.80	54.80	24.00	40.96	38.89	61.25	27.70	41.98	66.89	42.17	57.90
MAE (no judge training)	40.20	54.80	18.50	37.35	32.58	64.52	27.66	43.80	67.13	39.93	57.24
question quality filtering and format reward											
MAE (no question quality filtering)	36.02	48.54	27.33	32.53	24.15	60.05	19.40	36.40	67.92	35.55	56.15
MAE (no format reward)	**53.54**	56.55	26.42	**42.17**	**39.44**	**64.72**	25.91	43.45	68.24	**44.03**	59.44

5.3.4 Format Reward and Question Quality Filtering

In MAE, we incorporate format rewards and question quality filtering as dual safeguards to ensure training stability. We conduct ablation studies to investigate their contributions (Table 3).

Intriguingly, we observe a functional overlap between these components: quality filtering implicitly covers the role of format verification. Malformed generations (e.g., those with missing tags) are typically deemed low-quality by the Judge and discarded. Consequently, removing explicit format rewards yields only a marginal decrease in performance (59.87 $\rightarrow$ 59.44). In contrast, removing quality filtering results in a significant 3.72% drop (59.87 $\rightarrow$ 56.15). This sharp decline occurs because the dataset becomes polluted with "hacking" samples. Low-quality questions (*e.g.*, unsolvable or ambiguous) are accumulated in the training set, which then disrupt the whole training process.

These results demonstrate the importance of quality filtering and reinforce our decision to adopt it in the asymmetric self-play scheme. Relying solely on difficulty-based rewards could incentivize the Proposer to generate unsolvable questions or ambiguous contexts to hack the system and earn higher rewards. The Judge's direct evaluation, however, provides oversight of the dataset from a general perspective on feasibility and quality.

6 Conclusion

We presented Multi-Agent Evolve, a closed-loop, self-evolving RL framework that instantiates a Proposer–Solver–Judge triad from a single backbone LLM. Without relying on external verifiers or human-curated labels, Multi-Agent Evolve delivers consistent gains over base and SFT baselines across math, coding, reasoning, and general-knowledge benchmarks on Qwen2.5-3B-Instruct, with ablations confirming the necessity of each role and the benefits of question quality filtering. Our analysis highlights stability considerations, particularly the entropy decay of the Proposer role, and identifies a crucial turning point that improves performance during our training process. Future work includes scaling to larger backbones, applying more fine-grained entropy control, and integrating verifiable environments to build a unified platform where models can evolve across all general domains without human supervision.

Limitations

While our framework demonstrates effective self-improvement, we acknowledge a few limitations for future exploration. First, although our current system prompts proved stable, we did not extensively test the framework's sensitivity to prompt variations. Second, the Proposer-Solver dynamic currently focuses on single-turn tasks; extending this design to support multi-turn interactions for complex agentic environments remains an exciting

next step. Finally, due to computational resource constraints, our experiments were conducted at 3B and 7B parameter scale. We anticipate that scaling this approach to frontier models with richer intrinsic knowledge would unlock even stronger co-evolutionary capabilities.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Matteo Bettini, Amanda Prorok, and Vincent Moens. 2024. BenchMARL: Benchmarking multi-agent reinforcement learning. *Journal of Machine Learning Research*, 25(217):1–10.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. 2026. Why do multi-agent LLM systems fail? *Advances in Neural Information Processing Systems*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2024. AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

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

Anthony DiGiovanni and Ethan C Zell. 2021. Survey of self-play in reinforcement learning. *arXiv preprint arXiv:2107.02850*.

Wenkai Fang, Shunyu Liu, Yang Zhou, Kongcheng Zhang, Tongya Zheng, Kaixuan Chen, Mingli Song, and Dacheng Tao. 2026. SeRL: Self-play reinforcement learning for large language models with limited data. *Advances in Neural Information Processing Systems*.

Carlos Gómez-Rodríguez and Paul Williams. 2023. A confederacy of models: A comprehensive evaluation of LLMs on creative writing. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14504–14528.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. 2026. A survey on LLM-as-a-Judge. *The Innovation*, 7(6).

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025a. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Meng-Hao Guo, Jiajun Xu, Yi Zhang, Jiaxi Song, Haoyang Peng, Yi-Xuan Deng, Xinzhi Dong, Kiyohiro Nakayama, Zhengyang Geng, Chen Wang, Bolin Ni, Guo-Wei Yang, Yongming Rao, Houwen Peng, Han Hu, Gordon Wetzstein, and Shi-Min Hu. 2025b. R-Bench: Graduate-level multi-disciplinary benchmarks for LLM & MLLM complex reasoning evaluation. *arXiv preprint arXiv:2505.02018*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

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

Dan Hendrycks, Dawn Song, Christian Szegedy, Honglak Lee, Yarin Gal, Erik Brynjolfsson, Sharon Li, Andy Zou, Lionel Levine, Bo Han, Jie Fu, Ziwei Liu, Jinwoo Shin, Kimin Lee, Mantas Mazeika, Long Phan, George Ingebretsen, Adam Khoja, Cihang Xie, and 14 others. 2025. A definition of AGI. *arXiv preprint arXiv:2510.18212*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. 2025. REINFORCE++: An efficient RLHF algorithm with robustness to both prompt and reward models. *Preprint*, arXiv:2501.03262.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. 2026. R-Zero: Self-evolving reasoning LLM from zero data. In *International Conference on Learning Representations*.

Earl Hunt. 2010. *Human intelligence*. Cambridge University Press.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, Kai Shu, Lu Cheng, and Huan Liu. 2025. From generation to judgment: Opportunities and challenges of LLM-as-a-Judge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 2757–2791, Suzhou, China. Association for Computational Linguistics.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024. NuminaMath. Hugging Face repository.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Zi Lin, Sheng Shen, Jingbo Shang, Jason Weston, and Yixin Nie. 2025. Learning to solve and verify: A self-play framework for code and test generation. *Preprint*, arXiv:2502.14948.

Bo Liu, Simon Yu, Zichen Liu, Leon Guertler, Penghui Qi, Daniel Balcells, Mickel Liu, Cheston Tan, Weiyan Shi, Min Lin, Wee Sun Lee, and Natasha Jaques. 2026. SPIRAL: Self-play on zero-sum games incentivizes reasoning via multi-agent multi-turn reinforcement learning. In *International Conference on Learning Representations*.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*.

Mickel Liu, Liwei Jiang, Yancheng Liang, Simon Shaolei Du, Yejin Choi, Tim Althoff, and Natasha Jaques. 2025. Chasing moving targets with online self-play reinforcement learning for safer language models. *Preprint*, arXiv:2506.07468.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

Sumeet Ramesh Motwani, Chandler Smith, Rocktim Jyoti Das, Rafael Rafailov, Ivan Laptev, Philip H. S. Torr, Fabio Pizzati, Ronald Clark, and Christian Schroeder de Witt. 2025. MALT: Improving reasoning with multi-agent LLM training. *Preprint*, arXiv:2412.01928.

OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov,

Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, and 7 others. 2019. Dota 2 with large scale deep reinforcement learning. *Preprint*, arXiv:1912.06680.

Jing-Cheng Pang, Pengyuan Wang, Kaiyuan Li, Xiong-Hui Chen, Jiacheng Xu, Zongzhang Zhang, and Yang Yu. 2024. Language model self-improvement by reinforcement learning contemplation. In *International Conference on Learning Representations*.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, and 24 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Shreyas Sundara Raman, Vanya Cohen, Ifrah Idrees, Eric Rosen, Raymond Mooney, Stefanie Tellex, and David Paulius. 2024. CAPE: Corrective actions from precondition errors using large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14070–14077. IEEE.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A graduate-level Google-Proof Q&A benchmark. *Preprint*, arXiv:2311.12022.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. WinoGrande: An adversarial Winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *Preprint*, arXiv:1712.01815.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, pages 387–395. PMLR.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-Bench tasks and whether Chain-of-Thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.

Yashar Talebirad and Amirhossein Nadiri. 2023. Multi-agent collaboration: Harnessing the power of intelligent LLM agents. *Preprint*, arXiv:2306.03314.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158.

Yiding Wang, Yuxuan Chen, Fangwei Zhong, Long Ma, and Yizhou Wang. 2025. Simulating human-like daily activities with desire-driven autonomy. In *International Conference on Learning Representations*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh Agrawal, Sandeep Singh Sandha, Siddartha Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. 2025. LiveBench: A challenging, contamination-limited LLM benchmark. In *International Conference on Learning Representations*.

Ziyu Ye, Rishabh Agarwal, Tianqi Liu, Rishabh Joshi, Sarmishta Velury, Quoc V. Le, Qijun Tan, and Yuan Liu. 2025. Scalable reinforcement post-training beyond static human prompts: Evolving alignment via asymmetric self-play. *Preprint*, arXiv:2411.00062.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason

Weston. 2025. Self-Rewarding Language Models. *Preprint*, arXiv:2401.10020.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Yifan Zeng, Yiran Wu, Xiao Zhang, Huazheng Wang, and Qingyun Wu. 2024. AutoDefense: Multi-agent LLM defense against jailbreak attacks. *Preprint*, arXiv:2403.04783.

Andrew Zhao, Yiran Wu, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. 2026. Absolute Zero: Reinforced self-play reasoning with zero data. *Advances in Neural Information Processing Systems*.

Yifei Zhou, Sergey Levine, Jason Weston, Xian Li, and Sainbayar Sukhbaatar. 2026. Self-challenging language model agents. *Advances in Neural Information Processing Systems*.

A Additional Analysis and Discussion

A.1 Motivation for Shared Parameters and Training Stability

We instantiate the Proposer, Solver, and Judge from a single shared-parameter backbone for two reasons. First, the underlying capabilities required by the three roles are correlated: a stronger Solver should also be better at judging correctness and proposing suitable questions, and improvements in judging or proposing can in turn create better learning signals for the Solver. Role-specific prompts and rubrics specialize the shared model behavior while preserving the central goal of self-evolution: the model improves itself through multi-role interaction rather than relying on separate external components.

Second, shared parameters make the method substantially simpler and more cost-effective. Maintaining independent parameters for each role would increase memory and storage costs and slow training due to frequent role-specific checkpoint loading and saving. A shared backbone therefore makes Multi-Agent Evolve more practical in resource-constrained settings.

Training stability is primarily ensured by system-level design rather than parameter sharing alone. The role design, reward structure, and safeguard mechanisms regulate the co-evolutionary loop. In particular, question quality filtering prevents dataset corruption by discarding malformed or adversarially "hacked" questions; removing it causes a 3.72% drop in overall performance (Table 3). These mechanisms stabilize interaction dynamics regardless of whether parameters are shared or separated.

A.2 Gradient Conflicts

To assess whether the shared backbone suffers from destructive gradient conflicts, we collect role-specific gradients every 5 training steps, covering 60 out of 300 total steps, and compute pairwise cosine similarities between role-gradient directions.

Comparison	Min	Max	Mean
Proposer vs. Solver	0.7612	0.9177	**0.8492**
Proposer vs. Judge	0.7237	0.9002	**0.8114**
Solver vs. Judge	0.7857	0.9740	**0.8855**

Table 6: Pairwise cosine similarity between role gradients across training.

As shown in Table 6, all role pairs maintain high cosine similarity throughout training, with mean similarities above 0.81. This indicates that role-wise updates are broadly aligned and do not introduce severe gradient conflicts.

We further compare average similarity between the first and last quarters of training. The similarities decline mildly, suggesting that roles specialize as training progresses, but remain sufficiently aligned to avoid destructive updates.

Comparison	Start Avg.	End Avg.	Change
Proposer vs. Solver	0.8813	0.8378	-0.0435
Proposer vs. Judge	0.8532	0.7938	-0.0594
Solver vs. Judge	0.9309	0.8573	**-0.0736**

Table 7: Gradient cosine similarity from the first to the last quarter of training.

A.3 A Local Surrogate-Optimization View of Shared Updates

A natural concern is whether the Solver is limited by a fixed Proposer or Judge. In Multi-Agent Evolve, all three roles are instead role-conditioned behaviors of a shared backbone updated synchronously, so improvements in one role can change the data and learning signals available to the others. The following formulation is intended as an optimization view of this coupling, rather than a convergence guarantee for the full non-stationary self-evolution process.

At update step t, let $\mathcal{B}_t^{(r)}$ denote the samples for role $r \in \{P, S, J\}$. Consistent with Task-Relative REINFORCE++, each role contributes an empirical clipped surrogate objective

$$\widehat{\mathcal{L}}_t^{(r)}(\theta) = \frac{1}{|\mathcal{B}_t^{(r)}|} \sum_{i \in \mathcal{B}_t^{(r)}} \min\Big(\rho_{t,i}(\theta)\widehat{A}_{t,i}^{(r)}, \text{clip}(\rho_{t,i}(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}_{t,i}^{(r)}\Big), \tag{11}$$

where $\widehat{A}_{t,i}^{(r)}$ is normalized using the reward statistics of role r. The joint surrogate optimized through the shared backbone can then be written as

$$\widehat{\mathcal{L}}_t(\theta) = \sum_{r \in \{P,S,J\}} \alpha_r \widehat{\mathcal{L}}_t^{(r)}(\theta). \tag{12}$$

Before the optimizer transformation, accumulating the role-wise gradients is therefore equivalent to differentiating this weighted empirical surrogate:

$$\widehat{g}_t = \sum_{r \in \{P,S,J\}} \alpha_r \nabla_\theta \widehat{\mathcal{L}}_t^{(r)}(\theta_t) = \nabla_\theta \widehat{\mathcal{L}}_t(\theta_t). \tag{13}$$

This identity explains how the three role objectives are coupled by shared parameters. When their gradients are positively aligned, progress on one role is less likely to oppose progress on another; the empirical cosine similarities in Tables 6 and 7 support this interpretation for our training runs.

Importantly, $\widehat{g}_t$ should not be interpreted as an unbiased estimator of the gradient of the underlying role-wise expected returns. PPO-style clipping, batch-dependent advantage normalization, finite-sample estimation, and the evolving question and judged-pair distributions all introduce bias or non-stationarity. If clipping is locally inactive and the sampled batches and normalization statistics are held fixed, the update behaves locally like ascent on the weighted role-wise surrogate. This view clarifies the objective actually optimized by the synchronized shared-parameter update and complements the empirical gradient-alignment analysis above.

B Prompts, Detailed Reward Design and Training Hyperparameters

B.1 Prompts for Agents

Listing 1: Solver Prompt

```
## Task: Generate a High-Quality Response to a Given Task

You will be given a cognitive, creative, logical, mathematical, or planning-related task. Your
    job is to generate a complete, high-quality response that satisfies the task's constraints
    and demonstrates clear, structured reasoning or creativity.

### Instructions:
- Carefully read and understand the task.
- Think step by step - break down the task, simulate it mentally if needed, and reason through
    constraints.
- Then directly write your final response inside a pair of <answer></answer> tags(no need to
    restate or reformat the task).
- Your output should:
  * Be **correct** or **plausibly optimal**, given the task
  * **Fulfill all constraints** in the task
  * Be **clear** and **structured**
  * Avoid any vagueness or randomness
```

```
### Good Response Traits:
- For reasoning tasks: shows logical progression and result
- For generation tasks: respects the given constraints
- For math/logic/planning: includes a final answer that could be evaluated
- For creative tasks: coherent and original
```

Listing 2: Proposer_No_Ref_No_Answer_Generation Prompt

```
## Task: Create a Challenging and Original Task

Design a new and intellectually demanding task that tests **complex reasoning, creative thinking
    , structured planning, or deep understanding**. The task should be suitable for evaluation
    in general intelligence, reasoning benchmarks or instruction following.

You may design a task that resembles a quiz, puzzle or symbolic reasoning prompt. Focus on
    structure, challenge, and clarity.

---

### Task Requirements:

- The task must be:
  * **Self-contained** and clearly described
  * **Non-trivial**, requiring multiple reasoning steps, constraints, or synthesis
  * **Deterministic** or tightly constrained (even if open-ended in form)
  * **Free from cultural bias or real-time information**
  * **Difficult** but **not impossible** to solve

- Accepted Domains include:
  * Logic puzzles and reasoning tasks
  * Context comprehension
  * Common knowledge Q&A
  * Pattern-based math or symbolic challenges
  * Spatial planning or constraint problems
  * Instruction followings tasks

- Avoid:
  * Trivia questions or subjective writing
  * Ambiguous or taste-based open-ended prompts
  * Any dependency on web access or recent knowledge
  * Tasks with no clear solvability path

---

Follow the following format:

<question>
[Your generated question here]
</question>

MAKE SURE THAT EVERY GENERATED QUESTION AND **ONLY THE GENERATED QUESTION** IS INSIDE THE <
    question></question> TAGS!
USING THE CORRECT FORMAT AS GIVEN IS IMPORTANT!
```

Listing 3: Proposer_With_Ref_No_Answer_Generation Prompt

```
## Task: Create a Challenging and Modified Version of a Reference Task

Given one **reference task**, your goal is to design a **new, more challenging task** by making
    **controlled perturbations** to the original. The modifications should **increase reasoning
    depth, introduce extra constraints, or add multi-step dependencies** while keeping the
    problem **self-contained and solvable**.

You must preserve the **core domain or reasoning type** of the reference (e.g., if it's a logic
    puzzle, keep it a logic puzzle) but ensure the **surface content and structure are new**.
    You may:
- Add additional constraints or intermediate steps
- Replace elements with analogous but more complex structures
- Introduce distractors or traps that require careful reasoning
- Change numerical values, symbolic rules, or conditions to increase difficulty

---

### Task Requirements:

- The modified task must be:
  * **Self-contained** and clearly described
  * **Significantly different in surface form** from the reference, but maintain same reasoning
    type
  * **More challenging** - requiring additional steps or deeper analysis than the reference
  * **Deterministic** or tightly constrained
  * **Free from cultural bias, real-time info, or factual recall**

- Avoid:
  * Trivia questions or subjective writing
  * Ambiguous or taste-based open-ended prompts
  * Any dependency on web access or recent knowledge
  * Tasks with no clear solvability path

---

Follow the following format:

<question>
[Your generated question here]
</question>

MAKE SURE THAT EVERY GENERATED QUESTION AND **ONLY THE GENERATED QUESTION** IS INSIDE THE <
    question></question> TAGS!
USING THE CORRECT FORMAT AS GIVEN IS IMPORTANT!
```

Listing 4: Judge Answer Prompt

```
Please evaluate the following solution to a question/problem following a strict rubric.

Question/Problem: {question}

Generated Solution: {answer}

First, analyze the solution in the <think> and </think> tags below:

<think>
STRICT RUBRIC:
- Factual correctness is mandatory. ANY factual error (including arithmetic mistakes, violated
    common sense, wrong units, invalid assumptions, or faulty reasoning steps) => final score
    MUST be in [1,3].
- Meaningless or excessive repetition, rambling, or filler content => final score MUST be in
    [1,3].
- Hallucinated references, fabricated data, unsupported claims, or contradictions => final score
     MUST be in [1,3].
- Completeness & directness: The answer must fully address the asked question. Missing key steps
    , incomplete derivations, or only partially addressing the prompt => score in [4,7] unless
    there are factual errors (then [1,3]).
- Only answers that are entirely correct, follow instructions, are concise (no redundant
    repetition) may receive [8,10].

[Write your detailed analysis here: identify any factual errors, faulty reasoning steps or
    repetition issues. If ANY such issue exists, note that the score band must be [1,3]. If none
    , justify why it qualifies for 8-10; otherwise place it in 4-7.]
</think>

Then provide a score from 1 to 10 between <score> and </score> consistent with your thoughts
    where:
- 8-10: Flawless or near-flawless: entirely correct; no factual/logic/common-sense/calculation
    errors; concise with no meaningless repetition; fully answers the question.
- 4-7: Generally on-topic and partially correct, but has omissions or clarity problems (NO
    factual errors).
- 1-3: ANY factual/logic/common-sense/calculation error OR hallucination OR large meaningless
    repetition OR severe irrelevance.

<score>X</score> (where X is an integer from 1 to 10)
```

Listing 5: Judge Question Prompt

```
Please evaluate the quality of the following question generation following a strict rubric.

Question: {question}

First, analyze the question in the <think> and </think> tags below:

<think>
STRICT RUBRIC (Question):
- If the question is unsolvable as stated (insufficient information, internal contradiction,
    presupposition failure), violates common sense, logically incoherent or simply not in the
    form of a question => final score MUST be in [1,3].
- If the question is unsafe/inappropriate or not actually a question => final score MUST be in
    [1,3].
- If the question is too open-ended => final score MUST be in [1,3].
- Clarity & completeness: ambiguous wording, missing constraints, or unclear objectives =>
    typically [4,7] (unless it's unsolvable/incoherent, then [1,3]).
- Only questions that are fully clear, well-formed, logically sound, feasible/solvable, relevant
    , and concise (no redundant repetition) may receive [8,10].

[Write your detailed analysis here: check solvability, logical soundness, clarity, relevance,
    and redundancy. If ANY unsolvability or commonsense violation exists, score must be [1,3].]
</think>

Then provide a score from 1 to 10 between <score> and </score> consistent with your thoughts
    where:
- 8-10: Perfectly clear, feasible, self-contained, logically sound, and concise; appropriate and
     relevant.
- 4-7: Generally reasonable but with notable ambiguity, missing details, or minor issues (NO
    unsolvability/commonsense violations).
- 1-3: Unsolvable/contradictory/commonsense-violating/unsafe/irrelevant/not a valid question.

<score>X</score> (where X is an integer from 1 to 10)
```

B.2 Detailed Reward Design

In this section, we provide the specific reward formulations and hyperparameter configurations for the three functional roles in our Multi-Agent Evolve (MAE) framework: the Proposer, the Solver, and the Judge.

B.2.1 Reward for the Proposer

The Proposer's objective is to generate questions that are both high-quality (well-formed, solvable) and challenging for the current Solver. The total reward R_P for a generated question q is a weighted sum of three components:

$$R_P(q) = \lambda_Q R_Q + \lambda_D R_D + \lambda_F R_F \tag{14}$$

1. Quality Reward (R_Q): This score is assigned by the Judge agent, which evaluates the intrinsic quality of the question q. The rubric prioritizes clarity, logical coherence, and feasibility.

$$R_Q(q) = V_J(q) \in [0, 1]$$

where $V_J(q)$ is the normalized score from the Judge.

2. Difficulty Reward (R_D): This reward incentivizes the Proposer to explore the frontier of the Solver's capabilities. We estimate difficulty by sampling N_{sample} responses from the current Solver policy π_S for the question q.

$$R_D(q) = 1 - \bar{R}_S(q) = 1 - \left(\frac{1}{N_{sample}} \sum_{i=1}^{N_{sample}} V_J(a_i, q) \right)$$

where $V_J(a_i, q)$ is the Judge's evaluation score for the Solver's answer a_i. A lower average solve rate results in a higher difficulty reward.

3. Format Reward (R_F): To ensure the automated pipeline functions correctly, we strictly penalize format errors. The Proposer must enclose the generated question within `<question>` tags.

$$R_F(\text{output}_P) = \begin{cases} 1.0 & \text{if exactly one valid } \texttt{<question>...</question>} \text{ pair is present} \\ 0.5 & \text{if multiple valid pairs are present} \\ 0.0 & \text{otherwise} \end{cases}$$

Configuration: In our experiments, we set the weights equally: $\lambda_Q = \lambda_D = \lambda_F = 1/3$.

B.2.2 Reward for the Solver

The Solver's goal is to generate accurate and comprehensive answers a for a given question q. Its reward R_S consists of a performance-based score from the Judge and a format constraint.

$$R_S(a) = \lambda_J R_J + \lambda_F R_F \tag{15}$$

1. Judge Reward (R_J): Since ground-truth labels are unavailable for the newly generated questions, the Solver relies on the Judge's evaluation.

$$R_J(a) = V_J(a, q) \in [0, 1]$$

The Judge uses a strict rubric to score the answer's correctness, completeness, and adherence to instructions.

2. Format Reward (R_F): The Solver must enclose its final answer within `<answer>` tags to facilitate parsing and evaluation.

$$R_F(\text{output}_S) = \begin{cases} 1.0 & \text{if exactly one valid } \texttt{<answer>...</answer>} \text{ pair is present} \\ 0.5 & \text{if multiple valid pairs are present} \\ 0.0 & \text{otherwise} \end{cases}$$

Configuration: In our experiments, we set the weights as $\lambda_J = 0.5$ and $\lambda_F = 0.5$.

B.2.3 Reward for the Judge

The Judge acts as the internal critic for the system. Unlike the other roles, the Judge is trained primarily to maintain structural consistency and to co-evolve its reasoning capabilities via the shared backbone updates. Its explicit reward is purely structural.

$$R_J(g) = R_F(g) \tag{16}$$

1. Format Reward (R_F): The Judge must output a numerical score within specific tags to be parsed by the training loop.

$$R_F(\text{output}_J) = \begin{cases} 1.0 & \text{if exactly one valid } \texttt{<score>X</score>} \text{ tag is present} \\ 0.5 & \text{if multiple valid tags are present} \\ 0.0 & \text{otherwise} \end{cases}$$

where X is an integer representing the score. While the Judge does not receive a direct "ground-truth" reward for its scoring accuracy (as none exists), the parameter sharing with the optimized Solver and Proposer implicitly refines its reasoning logic over time.

B.3 Training Hyperparameters

We show our training hyperparameters in Table 8. These parameters remain constant across all our experiments.

C Training Pseudo-code

Algorithm 1 details the complete training loop of the Multi-Agent Evolve framework, illustrating the interaction between the Proposer, Solver, and Judge, as well as the synchronized parameter update mechanism.

Table 8: Training hyperparameters used in our experiments.

Hyperparameter	**Value**
Max Prompt Length	8192
Max Response Length	8192
Train Batch Size	64
Learning Rate	1×10^{-6}
Optimizer	AdamW
Grad Clip	1.0
Training Steps	300
Algorithm	Task-Relative REINFORCE++
KL Loss	False
KL Reward	False
PPO Epochs	1
Entropy Coefficient	0.001
Actor Rollout Temperature	1.0
Actor Rollout Top-P	1.0
N Rollout	1
N Reference Problem	1
N samples to Estimate Difficulty	5

D Evaluation Prompts and Configuration

D.1 Prompts for evaluating math problems

Listing 6: Math Problem Evaluation Prompt

```
Please evaluate whether the following mathematical answer is correct.

Question: {question}

Ground Truth Answer: {ground_truth}

Model's Answer: {model_answer}

First, analyze both answers in the <think> tags below:

<think>
Compare the mathematical content of both answers:
- Do they arrive at the same final numerical result? (MOST IMPORTANT)
- Is the reasoning correct in the model's answer?
- Are there any mathematical errors?
- Do they represent the same solution even if expressed differently?

[Write your detailed analysis here]
</think>

Then determine if the model's answer is correct:
- TRUE means the model's answer is mathematically correct and matches the ground truth
- FALSE means the model's answer is mathematically incorrect or doesn't match the ground truth

<answer>TRUE</answer> or <answer>FALSE</answer>
```

Algorithm 1 Training Workflow for Multi-Agent Evolve

Require: Base LLM M, role instructions I_P, I_S, I_J, seed question dataset $\mathcal{D}_0$, training steps T, question quality threshold α

Initialize Proposer π_P, Solver π_S, and Judge π_J from M
Initialize question dataset $\mathcal{D} \leftarrow \mathcal{D}_0$, pair dataset $\mathcal{P} \leftarrow \emptyset$
for $t = 1, \ldots, T$ **do**
  {**— Proposer Phase —**}
  if Propose_With_Reference **then**
    Sample q_{ref} from dataset $\mathcal{D}$ {Sample reference question}
    $q_t, \text{output}_P \leftarrow \pi_P(I_P, q_{ref})$ {Generate new question}
  else
    $q_t, \text{output}_P \leftarrow \pi_P(I_P, \emptyset)$ {Generate new original question}
  end if
  Sample $a_i \sim \pi_S(\cdot \mid I_S, q_t)$ for $i = 1, \ldots, N$
  $\bar{R}_S(q_t) \leftarrow \frac{1}{N} \sum_{i=1}^{N} V_J(a_i, q_t)$ {Calculate avg solve score}
  $R_D(q_t) \leftarrow 1 - \bar{R}_S(q_t)$ {Calculate difficulty}
  $R_Q(q_t) \leftarrow V_J(q_t)$ {Evaluate question quality}
  if $R_Q(q_t) \geq \alpha$ **then**
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{q_t\}$ {Update question dataset if qualified}
  end if
  Reward: $R_P(q_t) \leftarrow \lambda_Q R_Q(q_t) + \lambda_D R_D(q_t) + \lambda_F R_F(\text{output}_P)$
  {**— Solver Phase —**}
  Sample q from dataset $\mathcal{D}$
  $a_t, \text{output}_S \leftarrow \pi_S(I_S, q)$ {Solver answers the question}
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{(q, a_t)\}$ {Update pair dataset}
  Reward: $R_S(a_t) \leftarrow \lambda_J V_J(a_t, q) + \lambda_F R_F(\text{output}_S)$
  {**— Judge Phase —**}
  Sample (q, a) from dataset $\mathcal{P}$
  $g \leftarrow \pi_J(I_J, q, a)$ {Generation for evaluation ($g = \text{output}_J$)}
  Reward: $r_J \leftarrow R_F(g)$
  {**— Update Phase —**}
  Update π_P, π_S, π_J based on R_P, R_S, r_J using Task-Relative REINFORCE++
end for
Return improved model M' aggregated from π_P, π_S, π_J

D.2 Prompts for evaluating multiple choice problems

Listing 7: Multiple Choice Problem Evaluation Prompt

```
Please evaluate whether the multiple choice answer is correct.

Question: {question}

Correct Answer: {ground_truth}

Model's Answer: {model_answer}

First, analyze the answers in the <think> tags below:

<think>
Compare the answers:
- Does the model select the same choice letter as the ground truth?
(If the model did not provide a letter, count it as incorrect)

[Write your detailed analysis here]
</think>
```

```
Then determine if the model's answer is correct:
- TRUE means the model selected the exact correct choice
- FALSE means the model selected the wrong choice

<answer>TRUE</answer> or <answer>FALSE</answer>
```

D.3 Prompts for evaluating truthfulness problems

Listing 8: Truthfulness Problem Evaluation Prompt

```
Please evaluate the truthfulness and accuracy of the answer.

Question: {question}

Ground Truth Answer: {ground_truth}

Model's Answer: {model_answer}

First, analyze the answers in the <think> tags below:

<think>
Evaluate the truthfulness:
- Is the model's answer factually accurate?
- Does it align with the ground truth answer?
- Are there any misleading or false statements?

[Write your detailed analysis here]
</think>

Then determine if the model's answer is truthful and accurate:
- TRUE means the answer is truthful and accurate
- FALSE means the answer contains false or misleading information

<answer>TRUE</answer> or <answer>FALSE</answer>
```

D.4 Prompts for evaluating other general problems

Listing 9: General Problem Evaluation Prompt

```
Please evaluate whether the answer is correct based on the ground truth answer.

Question: {question}

Ground Truth Answer: {ground_truth}

Model's Answer: {model_answer}

First, analyze the answers in the <think> tags below:

<think>
Compare the model's answers with the ground truth answer::
- Does it match the ground truth answer?

[Write your detailed analysis here]
</think>

Then determine if the model's answer is correct:
- TRUE means the answer is correct and appropriate
- FALSE means the answer is incorrect or inappropriate

<answer>TRUE</answer> or <answer>FALSE</answer>
```

D.5 Evaluation Configuration

As previously mentioned, we use a strong LLM as a judge for evaluating model outputs on most benchmarks. The configuration for this judge model is detailed in Table 9.

Table 9: LLM Judge Configuration

Parameter	Value
Model Name	`nvidia/llama-3.1-nemotron-70b-instruct`
Temperature	1.0
Top P	0.95
Max Tokens	1000

D.6 LLM-as-a-judge Reliability

Our use of LLM-as-a-judge in evaluation is limited to factual TRUE/FALSE matching against ground truth, rather than open-ended preference scoring. This setting is substantially simpler than general subjective judging, where reliability concerns are more pronounced. Coding benchmarks are evaluated separately through EvalPlus (Liu et al., 2023) using objective execution pass rates.

We further validate the reliability of this evaluation protocol with a human agreement study. The LLM judge reaches 85% agreement with rule-based matching and an average 95% agreement with human annotations across benchmarks. These results suggest that LLM-as-a-judge is reliable in our constrained evaluation setting while remaining flexible enough to handle diverse model answer formats.

For our evaluation, we select two groups of datasets. The first group consists of test sets corresponding to our reference data (in-distribution setting). These benchmarks include GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), ARC-Challenge (ARC-C) (Clark et al., 2018), MMLU (Hendrycks et al., 2020), GPQA (Rein et al., 2023), CommonsenseQA (CQA) (Talmor et al., 2019), OpenBookQA (OBQA) (Mihaylov et al., 2018), Natural Questions (NQ) (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), SQuAD (Rajpurkar et al., 2016), BoolQ (Clark et al., 2019), and HellaSwag (Zellers et al., 2019). The second group consists of held-out datasets from an out-of-reference-data distribution. These include TruthfulQA (Lin et al., 2022), BIG-Bench Hard (BBH) (Suzgun et al., 2023), LiveBench Reasoning (White et al., 2025), AMC (Li et al., 2024), Minerva (Lewkowycz et al., 2022), WinoGrande (Sakaguchi et al., 2021), OlympiadBench (He et al., 2024), and MMLU-Pro (Wang et al., 2024).

E Generation Examples

In this section, we provide some detailed example generations for each role of our framework.

E.1 Proposer

We separately provide examples of questions with low/high judge scores, only with the extracted question part.

Listing 10: Proposer Generation Example With Low Judge Score

```
What is the next shape in the following sequence?
<list>
<item>
Circle
<item>
Square
<item>
Triangle
<item>
Question mark (?)
<item>
Rectangle
<item>
Diamond
</list>
```

Listing 11: Proposer Generation Example With High Judge Score

```
Question: Which of the following water solutions is most suitable for gargling to soothe sore
    throats, considering you need to adjust it to eliminate bitterness and maintain a neutral pH
    ?

(A) Mix 30 grams of table salt with 40 ounces (1.18 liters) of water, then add 2 grams of baking
     soda to simplify the taste and adjust slightly towards neutrality.

(B) Dissolve 30 grams of table salt in 40 ounces (1.18 liters) of warm water, and then add 2
    grams of sugar to the mixture.

(C) Combine 15 grams of table salt with 25 ounces (0.75 liters) of warm water, and add 1 gram of
     baking soda to remove any harshness and increase pH slightly.

(D) Stir 30 grams of table salt into 40 ounces (1.18 liters) of tepid water, then sprinkle 1
    gram of granulated sugar on top to mask the taste and balance sweetness.
```

E.2 Solver

Listing 12: Solver Generation Example

```
Question: What is the smallest positive integer that is both a square and a cube, but not a
    perfect fifth power?
==========================================
Answer: <think> To find the smallest positive integer that is both a square and a cube but not a
     perfect fifth power, it must be a sixth power, as this is the lowest common multiple of 2
    and 3.</think>

<answer>1</answer>
```

Listing 13: Solver Generation Example

```
Question: If a square garden has a side length represented by the expression \( 2x + 3 \) meters
    , and there is another square garden with a side length of \( 5 \) meters. Find the total
    area of both gardens, provided that \( x = 4 \). What is the sum of the areas of the two
    gardens?
==========================================
Answer: <think> To find the total area of both gardens, first calculate the area of the first
    garden using \(2x + 3\) with \(x = 4\), then calculate the area of the second garden
    directly, sum these areas, and simplify.</think>

<answer>88</answer>
```

Listing 14: Solver Generation Example

```
Question: How many colors are needed to paint the map such that no two adjacent regions are the
    same color, and which regions use which colors? Given the map with five regions.

Region 1: adjacent to regions 2 and 3.
Region 2: adjacent to regions 1, 3, and 4.
Region 3: adjacent to regions 1, 2, and 4, and the same with region 5.
Region 4: adjacent to regions 2, 3, and 5.
Region 5: adjacent to regions 3 and 4.
==========================================
Answer: <think> To solve this, we can use the Four Color Theorem, which states that any map can
    be colored using at most four colors such that no two adjacent regions share the same color.
     We only need to show one possible coloring for the given map.</think>

<think>Let's start by assigning colors to the regions based on adjacent requirements:</think>

1. Region 1 can be colored with color A.
2. Region 2, being adjacent to Region 1 and Region 3, must be colored with color B.
3. Region 3, with being adjacent to Region 1 and Region 2, can be colored with color C,
    different from both A and B.
4. Region 4, adjacent to Region 2 and Region 3, must be colored with color D, different from
    color B.
5. Region 5, adjacent to Regions 3 and 4, can be colored with color A, different from color C.
6. Finally, Region 5 still needs color C, so we use an option to reuse previously assigned
    colors to maintain no two adjacent regions having the same color.

<think>Thus, the coloring will cycle using all four colors to ensure no two adjacent regions are
     the same color:</think>

<answer>
Region 1: A
Region 2: B
Region 3: C
Region 4: D
Region 5: A
</answer>
```

Listing 15: Solver Generation Example

```
Question: Lily had $200 in her account. She first spent $20 on a t-shirt. Then, she visited a
    second store where she purchased a hat, spending only one-third as much as she did on the t-
    shirt. After the hat, she went to a third shop and purchased a pair of headphones, spending
    twice the amount she spent on the hat. Finally, she went to a fourth shop to buy a camera,
    spending three times as much as she did on the headphones. How much money is left in Lily's
    account?
==========================================
Answer: <think>Let's break down Lily's expenses step-by-step and then determine how much money
    she has left in her account.

1. Expenditure on t-shirt: Lily spent $20 on a t-shirt.
2. Expenditure on hat: Lily spent one-third as much as she did on the t-shirt on the hat. Since
    she spent $20 on the t-shirt, she spends \( \frac{1}{3} \times $20 = $6.6667 \) on the hat.
3. Expenditure on headphones: The amount spent on headphones is twice the amount she spent on
    the hat, which is \( 2 \times $6.6667 = $13.3333 \).
4. Expenditure on camera: She spends three times the amount she did on the headphones on the
    camera, which is \( 3 \times $13.3333 = $40 \).

Now, let's calculate Lily's total expenditure and how much money she has left:
- Total expenditure on t-shirt: $20
- Total expenditure on hat: $6.6667
- Total expenditure on headphones: $13.3333
- Total expenditure on camera: $40

Total amount spent = $20 + $6.6667 + $13.3333 + $40 = $80

Since Lily initially had $200 in her account, the remaining amount in her account is:
- Initial amount in account: $200
- Total amount spent: $80

Remaining amount in account = Initial amount - Total amount spent = $200 - $80 = $120
</think>
<answer>
The amount of money left in Lily's account after her shopping spree is $120.
</answer>
```

E.3 Judge

Listing 16: Judge Generation Example For Question Evaluation

```
To evaluate the quality of the question generation, we need to carefully analyze it against the
    provided rubric. Let's break down the components:

1. **Solvability and Logical Soundness**:
    - The question specifies a 15x15 grid filled with integers ranging from 1 to 450.
    - It imposes several constraints on the arrangement of these integers:
      - Each row, column, and the two main diagonals must sum to a prime number.
      - The sum of the numbers in the top-left, bottom-right, and top-right corner cells must
    also be a prime number.
      - The first descending quadrant and the first ascending quadrant must be symmetric with
    respect to their centers.
      - The difference between the sum of elements in every quadrant and its immediate
    diagonally adjacent quadrant must be a perfect square.
      - No row, column, or diagonal can be in an ascending order.
    - These constraints are complex and interdependent, making it challenging to determine if a
    solution is possible without a detailed analysis.

 2. **Clarity and Completeness**:
    - The question is well-structured and provides a clear set of instructions.
    - However, the constraints are intricate and require a deep understanding of number theory
    and combinatorial mathematics.
    - The diagram provided helps visualize the quadrants and their centers, which aids in
    understanding the symmetry and perfect square conditions.

 3. **Feasibility and Relevance**:
    - The problem is mathematically complex and requires advanced problem-solving skills.
    - While the constraints are mathematically sound, the task of arranging the numbers to meet
    all these conditions simultaneously is highly non-trivial.
    - The question is relevant to the field of combinatorial mathematics and number theory.

 Given the complexity and the need for a detailed analysis to determine solvability, the
    question is not entirely clear and complete without a thorough examination. However, it is
    feasible and relevant.

 <score>4</score>
```

Listing 17: Judge Generation Example For Answer Evaluation

```
To evaluate the solution, let's break down the rubric and analyze each aspect:

 1. **Factual correctness**: The solution correctly identifies that the annual interest rate of
    10% compounded quarterly means a quarterly interest rate of 2.5%. This is accurate and
    correct. The calculation of the amount after each quarter is also done correctly,
    maintaining the compounded interest rate. There are no arithmetic mistakes or contradictions
     in the calculations.

 2. **Meaningless or excessive repetition, rambling, or filler content**: The solution is
    concise and to the point. It breaks down the problem into clear steps and provides detailed
    calculations for each quarter. There is no unnecessary repetition or rambling.

 3. **Hallucinated references, fabricated data, unsupported claims, or contradictions**: The
    solution does not introduce any fabricated data, unsupported claims, or contradictions. All
    calculations and steps are based on the given information and follow the correct
    mathematical principles.

 4. **Completeness & directness**: The solution fully addresses the question by providing the
    amount of money John will have in his account after two full years. It breaks down the
    problem into manageable steps and provides the final answer. The steps are clear and the
    calculations are detailed.

 5. **Factual errors, faulty reasoning steps, or repetition issues**: There are no factual
    errors, faulty reasoning steps, or repetition issues in the solution. The calculations are
    correct and the steps are logically consistent.

 Given that the solution is entirely correct, follows the instructions, is concise, and fully
    addresses the prompt, it qualifies for a score of 8-10.

 <score>8</score>
```

F Seed Data Composition

Table 10 shows the composition of the 1000 seed data points (967 after removing data with thinking prompt) used to initialize the valid question set for the MAE settings that utilize reference examples.

Table 10: Distribution of Seed Data Across Benchmarks

Benchmark	**Count**
CommonsenseQA (Talmor et al., 2019)	70
TriviaQA (Joshi et al., 2017)	71
Natural Questions (Kwiatkowski et al., 2019)	79
OpenBookQA (Mihaylov et al., 2018)	75
BoolQ (Clark et al., 2019)	88
SQuAD (Rajpurkar et al., 2016)	82
MATH (Hendrycks et al., 2021)	84
HellaSwag (Zellers et al., 2019)	90
GSM8K (Cobbe et al., 2021)	77
GPQA (Rein et al., 2023)	69
MBPP (Austin et al., 2021)	53
ARC-Challenge (Clark et al., 2018)	78
MMLU (Hendrycks et al., 2020)	69
HumanEval (Chen et al., 2021)	15
Total	**967**