MAPoRL: Multi-Agent Post-Co-Training for Collaborative Large Language Models with Reinforcement Learning

Anonymous ACL submission

Abstract

Leveraging multiple large language models (LLMs) to build collaborative *multi-agentic* workflows has recently demonstrated significant potential. However, most studies focus on prompt engineering on top of out-of-the-box LLMs, overly relying on their suboptimal collaborative behaviors and suffering the limitations thereof. In this paper, we introduce a novel post-training paradigm **MAPoRL** (**M**ulti-**A**gent **P**ost-c**o**-training for collaborative LLMs with **R**einforcement **L**earning), designed to further unleash the power of multi-agentic frameworks by *co-training* LLMs to better collaborate. In **MAPoRL**, multiple LLMs first generate their own responses independently and engage in a multi-turn discussion to collaboratively improve the final answer. In the end, the MAPoRL verifier evaluates both the answer and the discussion, assigning a score that validates the answer correctness and encourages corrective and persuasive discussion through reward shaping. The score serves as the co-training reward and is maximized through multi-agent RL. Unlike existing LLM post-training paradigms, our key novelty is the advocacy of *co-training* multiple LLMs together using *RL* for better generalization. Accompanied by analytical insights, our experiments demonstrate that training individual LLMs alone is insufficient to encourage collaboration. In contrast, multi-agent co-training significantly boosts the collaboration performance across benchmarks and generalizes to unseen domains.

1 Introduction

Recent advances in large language models (LLMs) have highlighted their potential for collaboration, particularly within the *multi-agentic* framework (Du et al., 2024; Li et al., 2023; Kim et al., 2024b). The shift from single-agent to multi-agent systems introduces new dimensions and challenges in enabling effective collaboration among LLM agents. Recent approaches on multi-agent collaboration mostly rely on pre-defined format of the collaboration, such as debate (Khan et al., 2024; Du et al., 2024), for pre-trained LLMs. However, such approaches may struggle with achieving genuine collaboration among agents. For example, multi-agent debate has not consistently led to improved performance with additional turns (Huang et al., 2024).

This limitation may be somewhat expected – while LLMs may be able to *simulate* collaboration procedures, they are *not* explicitly *trained* to achieve effective cooperation. Technically, it is not hard to imagine that single-agent training is insufficient for collaboration – an *untrained* and *non-strategic* opponent can fail to provide feedback to promote collaboration. Instead, achieving collaborative behaviors requires interactive training environments where each agent actively engages with others and dynamically optimizes the strategy (Gagne, 1974; Macy, 1991; Hertz-Lazarowitz et al., 2013). Moreover, conventional approaches such as supervised fine-tuning (SFT), as we will show, are inadequate for this purpose, either: merely mimicking multi-agent interactions from training data does not lead to effective collaboration.

To develop more effective collaborative agents, we propose **M**ulti-**A**gent **P**ost-c**o**-training for collaborative LLMs with **R**einforcement **L**earning (**MAPoRL**), a *co-training* paradigm for multiple LLMs using multi-agent reinforcement learning (MARL). In MAPoRL, within pre-defined frameworks for multi-agent collaboration (*e.g.*, the debate framework (Du et al., 2024)), each agent receives rewards for their responses during collaboration, based on the quality of their answers and interactions. The objective for each agent in the MAPoRL process is to maximize their own value function, defined as the expected cumulative sum of rewards over the course of the collaboration.

To further encourage collaboration in MAPoRL, we also incorporate incentives for successful interactions and penalties for failures in collabora-

tion, steering the language models training toward more effective and cooperative behaviors. Through an analytical example, we validate the following insights: 1) single-agent training alone is insufficient to produce genuinely collaborative agents, and 2) co-trained agents can achieve an equilibrium with collaborative behaviors in simplified scenarios modeled using game theory.

To validate the effectiveness of MAPoRL, we conduct extensive experiments across diverse tasks and evaluation strategies. Specifically, we train multi-agent LLMs for tasks such as mathematical reasoning (GSM8k (Cobbe et al., 2021)) and natural language inference (ANLI (Nie et al., 2020)), comparing their performance against baseline approaches. Additionally, we evaluate the robustness of our method by testing agents on out-of-domain tasks (e.g., training on a NLI task and evaluating on a math dataset), demonstrating the generalization capabilities of our approach. We also explore the collaboration between agents of varying capabilities, by analyzing the impact of training *heterogeneous* LLMs together.

To the best of our knowledge, this study is *among the first works to explore the training of multi-LLM systems as a whole*[1], using RL, for multi-LLM collaboration.

Related Work. Due to space constraints, we defer discussion of related work to Appendix A.

2 Analytical Insights: Collaborate to Solve Hard Questions

In this section, we present a simplified model of LLM collaboration and explain (a) why *co-training* multiple LLMs is necessary compared to training a single agent, and (b) the role of incentives to further enhance collaboration during training. We validate both aspects through experiments in Section 4.

2.1 Problem Setup

We consider questions that inherently require collaboration for a successful solution. For instance, solving complex mathematical problems often require cooperation between multiple agents (Liang et al., 2024; Du et al., 2024). Beyond mathematics, collaboration can enhance performance on tasks related to privacy, factuality, and reliability (Feng et al., 2025). We model the interaction between two LLM agents as a repeated game with T turns. For simplicity, we assume that in each turn, each agent chooses between two actions: *Collaborate* (a_0) or *Act Independently* (a_1). For a given question q, we define $C(q)$ as a non-negative integer representing the *collaboration threshold*. The agents achieve *collaborative synergy* if, over the course of the T-turn interactions, the total number of collaborative actions (a_0) of all the agents meets or exceeds $C(q)$. When collaborative synergy is achieved, each agent receives a reward $R_{\text{syn}}(q) = 1$, representing a guaranteed (or near-guaranteed) correct solution. Prior to achieving synergy, agents receive rewards based on their chosen actions: a reward of $R_{\text{col}}(q)$ for choosing to collaborate (a_0) and $R_{\text{ind}}(q)$ for acting independently (a_1), where $R_{\text{col}}(q) < R_{\text{ind}}(q)$ (see Remark 1 for further discussion on setting). This reward structure creates a tradeoff between short-term accuracy and long-term collaborative success. Our theoretical setup builds upon the classical example of Coordination Games (Cooper, 1999) in Game Theory, introducing a novel collaboration threshold and synergy mechanism that shape the transition from independent actions to collaborative behavior, to better model the collaboration among multiple LLMs.

Remark 1 (Rationale Behind the Setup)**.** *This formalization captures key aspects of complex problem-solving dynamics. Choosing to collaborate (a_0) represents contributing exploratory ideas or partial solutions. While these contributions have a lower immediate probability of correctness $R_{col}(q)$, they are essential building blocks towards the complete solution. Acting independently (a_1) represents using conventional approaches that may yield a higher immediate probability of correctness $R_{ind}(q)$ but may contribute less to solving particularly challenging problems. The collaboration threshold $C(q)$ represents the minimum amount of collaboration efforts and idea generation needed to solve complex problems. Once this threshold is reached (collaborative synergy), the agents can combine their insights to solve the challenging problem, represented by the reward $R_{syn}(q)$.*

2.2 Analytical Observations

To provide intuition for why co-training is necessary and single-agent training can be inadequate, we will analyze the simplest case where $T = 2$ and $C(q) = 1$ to illustrate the fundamental differences between single and multi-agent training. We provide formal statements and proofs in Appendix C.

[1] Subramaniam et al. (2025) and Zhao et al. (2025) are the contemporaneous works, both of which were released within the past month while preparing this paper. In contrast to MAPoRL, the algorithms therein were based on (iterative) SFT, instead of RL

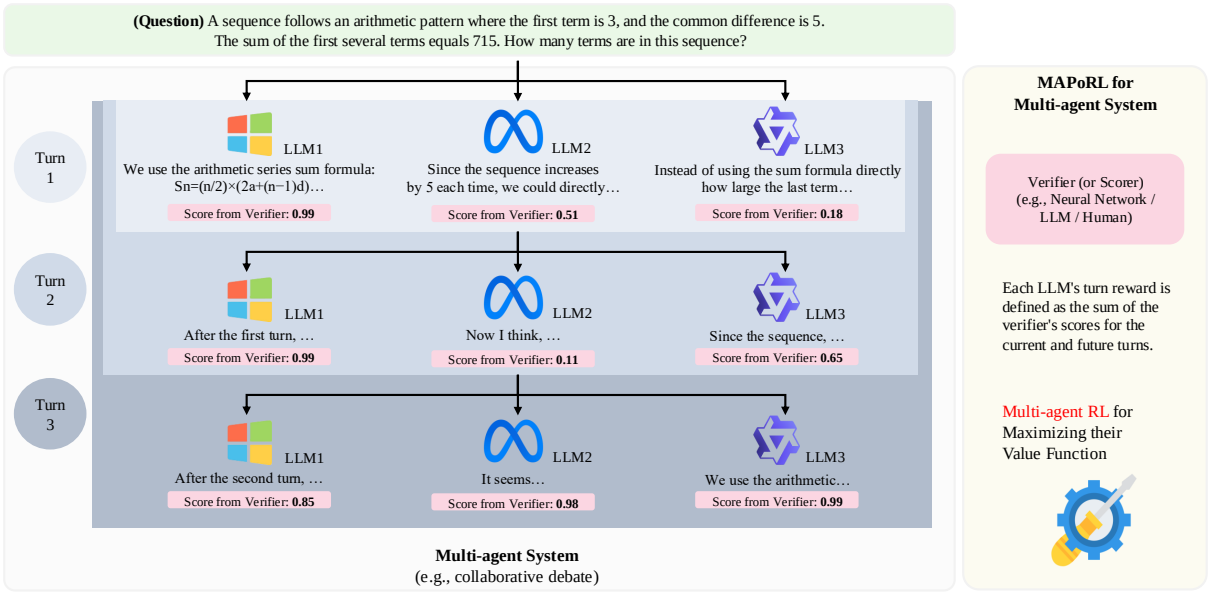

Figure 1: MAPoRL can be applied to any multi-LLM system with a scorer/verifier. In the illustrated example, it is integrated into a collaborative debate system for mathematical problem-solving. LLMs generate responses based on the multi-agent system pipeline, and a scorer/verifier evaluates their outputs. The reward for each LLM is determined based on these scores, which may include both current and future pipeline evaluations. Multi-Agent RL is employed to maximize each agent's value function.

Observation 1. *Suppose that the opponent selects action a_0 with probability $\pi(q)$ for each question q. Then, the optimal strategy for the agent is as follows: if $(R_{syn}(q) - R_{ind}(q))\pi(q) \geq 2R_{ind}(q) - R_{col}(q)$, then the optimal strategy for question q is to collaborate (a_0). Otherwise, the optimal strategy is to act independently (a_1).*

This shows the dependence of the agent's strategy on the opponent's behavior. If the opponent is *not collaborative* enough and *non-strategic*, then $\pi(q)$ will be small, leading the trained model to behave in a *non-collaborative* way.

Observation 2 (Informal)**.** *If both agents are trained to maximize their cumulative payoffs with a small entropy regularization, they will collaborate with high probability when:*

$$R_{syn}(q) = 1 > 3R_{col}(q) - 2R_{ind}(q).$$

Observation 2 can be proved by adapting the results of (Zhang and Hofbauer, 2016), transforming our setup with $T = 2$ into a Stag-Hunt game. This observation implies that when both agents optimize their own reward, they will naturally choose collaboration when $R_{\text{syn}}(q)$ is high enough, which emphasizes the importance of additional incentives for collaborative synergy. Due to this observation, in Section 3.3, we will incentivize collaboration by providing a higher $R_{\text{syn}}(q)$ when agents achieve the desired level of cooperation.

2.3 Toy Experiments with $T = 5$ Turns

To empirically illustrate why we need multi-agent training, we conduct a toy experiment. We defer the setting of $R_{\text{col}}(q), R_{\text{ind}}(q), R_{\text{syn}}(q), C(q)$ to Appendix D. Each interaction process lasts for $T = 5$ turns.

First, we train a single-agent parameterized by the Transformer model (Vaswani et al., 2017), while pairing it with a *fixed* opponent whose probability of choosing the collaborative action (a_0) is predetermined. Specifically, we consider two cases: $\pi(q) = 0.6$ and 0.7. Despite the relatively high probability of collaboration of the opponent, the single-agent training does not result in effective collaboration (Figure 2). In other words, the trained model tends to *avoid* collaboration rather than learning to strategically engage with it, highlighting the limitation of single-agent training when the opponent's behavior is fixed and non-strategic.

In contrast, we then *co-train* multiple Transformers where they are being trained simultaneously with access to the history of their interactions, with a modified distribution of the synergy reward. Instead of keeping $R_{\text{syn}}(q) = 1$ uniformly, we altered the reward distribution to $R_{\text{syn}}(q) \sim \text{Uniform}(1, R_{\text{new}})$, where $R_{\text{new}} > 1$ is chosen such that the long-term benefits of achieving collaborative synergy become more pronounced. Under this reward re-shaping, co-trained Transformers exhibited a significant increase in their propensity to collaborate (Figure 2). The effectiveness of explicitly incentivizing collaboration (by choosing a larger R_{new} is also validated.

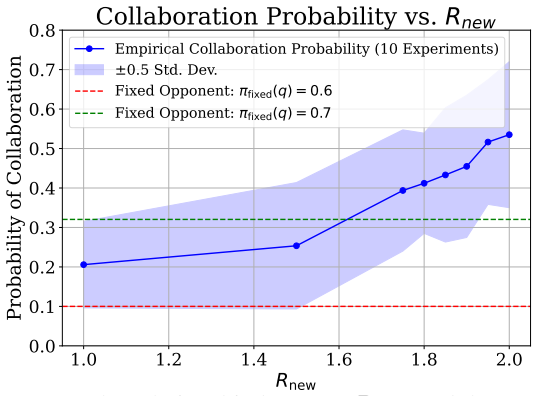

Figure 2: The relationship between R_{new} and the probability of collaboration (10 experiments). Higher synergy reward leads to higher probability to collaborate.

3 Post-Co-Training with MARL

We now provide an overview of our new paradigm *Multi-Agent Post-Co-Training with MARL (MAPoRL)* for LLM collaboration. In our framework, each agent's response for each turn is evaluated by a verifier that assigns a score reflecting the validity of the answer. The reward is defined as the sum (or weighted sum) of the verifier's scores from the current turn and those from the future turns, thus capturing both the immediate feedback and the projected long-term impact of the agent's response. The agents' policies are updated using multi-agent PPO (Schulman et al., 2017) for each turn, ensuring that the learning process incorporates both the current performance and the influence of the anticipated collaborative interactions.

Verifier Networks. Our goal is to train LLMs to generate a final solution s whose extracted answer is correct for the given question q. If we know the value of $\mathbb{P}(\text{Final answer is correct} \mid q, s^{1:x})$ where $s^{1:x}$ denotes the first x token of the solution s, then a natural approach is to train LLMs to maximize such a value. Thus, we train a *verifier network*, denoted by $\texttt{Verifier}_\theta(q, s^{1:x})$, to estimate $\mathbb{P}(\text{Final answer is correct} \mid q, s^{1:x})$ at the token level, which is also used in Cobbe et al. (2021); Yu et al. (2024); Liu et al. (2023). Our trained verifier's output theoretically provides the probability that the final answer is correct, as formally stated in Theorem 1. A detailed description of our verifier training methods and network architecture is provided in Appendix E.

3.1 Multi-Agent System - Collaborative Debate Formulation

We follow the collaborative debate system proposed by (Du et al., 2024) as an example of our multi-LLM system in the experiments. Note that, MAPoRL can be applied on any multi-agent system; each agent's response is evaluated using a verifier, which assigns a score reflecting the quality or correctness of the response. The reward for each agent is then determined by summing the verifier scores of all responses influenced by that agent throughout the multi-agent interaction process. Assume we have a collaborative debate system that runs for T turns and involves A agents. In each turn, an LLM must determine its next response based on the history of its own answers as well as the answers provided by other LLM agents. Let q be the given question, and let s_{ti} denote the solution provided by agent i at turn t. We inductively express the solution $s_{(t+1)i}$ as follows:

$$s_{1i} = \text{LLM}_i(q)\,,\ s_{(t+1)i} = \text{LLM}_i\left(q \oplus_{j\in[A],t'\in[t]} s_{t'j}\right)$$

where $\oplus$ denotes token-wise concatenation, $1 \le t \le T-1$ and $\text{LLM}_i(s)$ represents the function of inputting prompt s into the LLM_i which outputs logits over its token space, followed by sampling a token based on these logits. If $A = 1$, then this setup is equivalent to that of self-correcting LMs (Madaan et al., 2024). Now, we define $\boldsymbol{\theta} = (\theta_{ta})_{t\in[T],a\in[A]}$, where θ_{ta} represents the parameters of the ath agent at turn t. We denote LLM with θ_{ta} as $LLM_{\theta_{ta}}$.

Next, to implement MAPoRL, we define the reward function for MARL by applying a verifier score to the *Influence-aware Verification Rewards*.

Definition 1 (Influence-aware Verification Reward)**.** *The influence-aware verification reward function $R_{\boldsymbol{\theta}}(q, s_{ta})$ is defined as*

$$R_{\boldsymbol{\theta}}(q, s_{ta}) = \mathbb{E}\Bigg[\frac{1}{\sum_{t'\in[t,T]}\gamma^{t'-t}}\Bigg(\textit{Verifier}(q, s_{ta}) + \sum_{t'\in[t+1,T]}\sum_{j\in[A]}\frac{1}{A}\gamma^{t'-t}\textit{Verifier}(q, s_{t'j})\Bigg)\Bigg].$$

Here, the expectation arises from the randomness of other agents' answers, which are influenced by the agent's current response, and $\gamma \in [0, 1]$ is a discount factor. This reward not only considers the verifier's score for the current solution s_{ta} but also incorporates the impact of this solution on the future answers of all agents. The term $\sum_{j\in[A]}\frac{1}{A}$ averages the verifier's scores across all agents, reflecting the influence that s_{ta} has on the collective progress of the multi-agent system.

3.2 Multi-Agent RL Formulation

The reward of each agent, as well as its answer generation, is intertwined with the actions of other agents in the multi-LLM system. Thus, instead of single-agent RL, we design a multi-agent RL approach. For this paper, we select multi-agent PPO (Yu et al., 2022) as a representative multi-agent RL algorithm and instantiate it in the language domain. Our approach adapts multi-agent PPO by defining the state as the concatenation of the multi-agent interaction history, which differentiates it from conventional multi-agent PPO. Since we are solving multi-turn problems, the value function for each turn's state needs to be defined. The state of the each turns value $V_{ta\theta}$ is the expectation of the reward function conditioned on the input text i_{ta}^{x}, which is defined as

$$V_{ta\boldsymbol{\theta}}(i_{ta}^{x}) = \mathbb{E}\left[\sum_{x'=1}^{x} r_{\boldsymbol{\theta}}(q, s_{ta}^{1:x'}) \mid q, (s_{t'j})_{t' \in [t-1]}\right].$$

Here, $i_{ta}^{x'} = q \oplus_{t' \in [t-1], j \in [A]} s_{t'j} \oplus s_{ta}^{1:x'}$ and

$$r_{\boldsymbol{\theta}}(q, s_{ta}^{1:x'}) = \mathbf{1}(x' = \text{len}(s_{ta}))R_{\boldsymbol{\theta}}(q, s_{ta}) - \lambda_{\text{KL}}\text{KL}\left(\text{LLM}_{\boldsymbol{\theta}_{\text{ref}}}(i_{ta}^{x'-1}) \,\|\, \text{LLM}_{\boldsymbol{\theta}}(i_{ta}^{x'-1})\right),$$

where t denotes the turn index and a refers to agent $a \in [A]$, and $s_{ta}^{1:x}$ represents the generated token from agent a up to the x-th token in turn t, with $\boldsymbol{\theta}_{\text{ref}}$ denoting the parameters of the reference LLM. As per our reward construction, the value maximization not only considers the current turn's verifier score, but also anticipates future verifier scores from the same or the other agents across multiple turns, which makes multi-agent training valuable. We estimate the advantage function using Generalized Advantage Estimation (GAE) (Schulman et al., 2016), which leverages the value function to measure how much better the current token selection is compared to the baseline value function.

The value function is approximated by a neural network with parameters θ_{vta}, denoted as $V(i_{ta}^{x}; \theta_{vta})$, which serves as an estimate of $V_{ta\boldsymbol{\theta}}(i_{ta}^{x})$. Using $V(i_{ta}^{x}; \theta_{vta})$, we estimate the advantage function $A(i_{ta}^{x}; \theta_{vta})$ via GAE. The loss function for multi-agent PPO is then given by:

$$L_{\text{PPO}}(\boldsymbol{\theta}, \theta_{vta}) = L_{\text{Surrogate}}(\boldsymbol{\theta}, \theta_{vta}) + L_{\text{Value}}(\theta_{vta}),$$

where $L_{\text{Surrogate}}(\boldsymbol{\theta}, \theta_{vta})$ is defined as

$$\mathbb{E}\left[\min\left(R_{ta}^{x} A(i_{ta}^{x}; \theta_{vta}), \texttt{clip}_{\epsilon}(R_{ta}^{x}) A(i_{ta}^{x}; \theta_{vta})\right)\right],$$

and $L_{\text{Value}}(\boldsymbol{\theta}, \theta_{vta})$ is defined as

$$L_{\text{Value}}(\theta_{vta}) = \mathbb{E}\left[\lambda_{\text{value}}\left(V(i_{ta}; \theta_{vta}) - V_{ta}^{\text{target}}(i_{ta}^{x})\right)^2\right].$$

Here, $\texttt{clip}_{\epsilon}(\alpha) := \min(\max(1-\epsilon, \alpha), 1+\epsilon)$, $R_{ta}^{x} = \frac{\text{LLM}_{\boldsymbol{\theta}}(s_{ta}^{x+1} | i_{ta}^{x})}{\text{LLM}_{\boldsymbol{\theta}_{\text{old}}}(s_{ta}^{x+1} | i_{ta}^{x})}$, and $\boldsymbol{\theta}_{\text{old}}$ is the parameter used in the rollout for multi-agent PPO. The expectation $\mathbb{E}$ is taken over the randomness from $t \sim \text{Unif}([T]), a \sim \text{Unif}([A]), q \sim \mathcal{Q}, s \sim \text{LLM}_{\boldsymbol{\theta}_{\text{old}}}(q)$, and $x \sim \text{Unif}([\texttt{length}(s)])$, where $\mathcal{Q}$ indicates the distribution of questions. $L_{\text{Surrogate}}$ prevents policy updates from straying too far from the old policy by clipping the probability ratio. The value loss L_{Value} measures the MSE between the current value function and a target value, scaled by λ_{value}. The parameter ϵ relates to clipping, λ_{value} is the regularization factor for value differences, and λ_{KL} is a regularization factor using KL divergence.

Each agent for each turn optimizes its policy and value function simultaneously using the parameters $(\theta_{ta}, \theta_{vta})$. These agent interaction by multi-agent system inherently transforms the problem into an MARL problem, rather than a standard RL problem, as agents influence each other's learning processes throughout training.

3.3 Reward Shaping to Incentivize Collaboration

As discussed in Section 2, incorporating additional incentives in the reward can steer agents towards better collaboration. We define four key parameters when implementing such reward-shaping: parameters α_0 and α_1 correspond to the incentives related to an agent's own revision of the answer, and parameters β_0 and β_1 correspond to those related to her influence on other agents' answer. Specifically, α_0 represents the ability to extract useful information from incorrect answers (*critical reasoning*), while α_1 reflects an agent's tendency to be *persuaded* by the correct information. Meanwhile, β_0 represents the ability to provide incorrect answers that still contain useful information, potentially leading to better responses in the future turns. In contrast, β_1 captures an agent's ability to effectively *persuade others* when providing correct answers. We provide Table 6, 7 to summarize our incentive design.

4 Experiments

4.1 Datasets

We evaluate MAPoRL on the two NLP tasks to benchmark its reasoning performance in both mathematical reasoning and logical natural language inference. The details are as follows:

GSM8K (Cobbe et al., 2021) and TinyGSM (Liu et al., 2023). GSM8K is a benchmark dataset designed to assess mathematical reasoning abilities, requiring models to solve high-school-level mathematics problems. TinyGSM is an augmented version of GSM8K, generated using GPT-3.5-turbo,

where solutions are provided in Python. Importantly, we do not utilize the reasoning processes from GSM8K or TinyGSM but rely solely on their final answers. For training the verifier model, we use 7,463 samples from GSM8K. Additionally, we incorporate the first 12,800 synthetic samples from TinyGSM for MAPoRL. [2] For evaluation, we hold out 1,319 samples from GSM8K as a test set.

Adversarial Natural Language Inference (ANLI) (Nie et al., 2020). ANLI is designed to evaluate a model's natural language understanding by presenting adversarial examples that challenge even the state-of-the-art models. To train the verifier model, we use first 10,000 training examples. Furthermore, we use the next 12,800 examples for MAPoRL training and 1,200 samples for testing.

Evaluation Method. We defer the details of our accuracy measurement in Appendix.

4.2 Models

We primarily used the Microsoft Phi-3-mini-128k-instruct (3.4B) model (Abdin et al., 2024), together with Qwen2.5-3B-instruct (Yang et al., 2024) and Llama-3-8B-instruct (Dubey et al., 2024) for experiments. Due to computational constraints, we mainly used quantized models and fine-tuned them with QLoRA (Dettmers et al., 2024). We defer the training details to Appendix H. When evaluating on GSM8K and ANLI, we set the max token length to 300 and 250, respectively.

4.3 Experiment 1: Vanilla Collaboration by Off-the-shelf LLMs Cannot Improve Performance, While MAPoRL-Trained LLMs Can

We first compare the collaboration performance of off-the-shelf LLMs with MAPoRL-trained LLMs. The training is conducted with two agents collaborating over three turns. An overview of the trained system is provided in Figure 1. In Experiment 1, we train the model starting from turn $t \geq 2$ for two reasons: (a) the first turn primarily focuses on knowledge acquisition from each dataset, and (b) to ensure a fair comparison with off-the-shelf LLMs. We focus on enhancing collaboration skills rather than teaching specific task knowledge. For this experiment, we use Phi-3-mini-128k-instruct and evaluate the trained models in a three-agent and three-turn collaboration environment.

We observe that even when the off-the-shelf LLM is allowed to generate longer reasoning (600 tokens, twice the output length of our MAPoRL model), its accuracy did not improve across turns. This aligns with prior findings in the literature, particularly for models that are not sufficiently strong. For instance, Huang et al. (2024, Table 7) provides evidence that additional turns do not necessarily improve performance significantly. Similarly, our results show that the off-the-shelf LLM does not benefit from additional turns. In contrast, LLMs trained using MAPoRL exhibit improved performance as the number of collaboration turns increases, as shown in Figure 3.

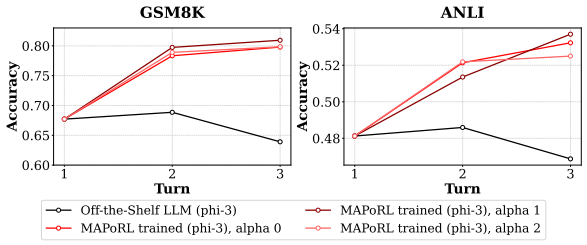

Figure 3: Performance comparison of different LLMs across tasks (left: GSM8k, right: ANLI) under various settings. We evaluate collaboration ability in five conditions: (1) off-the-shelf LLMs collaborating and (2) models trained using MAPoRL collaborating (with all incentive parameters (Section 4.4) $\alpha, \beta = 0, 1, 2$, respectively).

Remark 2 (Domain-Specific Knowledge Acquisition vs. Collaboration Ability Improvement)**.** *One might question whether the performance gains observed in MAPoRLtrained models stem from acquiring domain-specific knowledge rather than improved collaboration ability. To address this, we compare off-the-shelf LLMs and MAPoRL trained models by testing how well they perform on questions without any collaboration, providing MAPoRL trained models only the original question—without interaction history—to see if their performance is solely due to domain knowledge learned during training. The results are as follows: Here, we provide the same questions to the off-the-shelf Phi-3 model, the MAPoRL-trained turn 2 model, and the MAPoRL-trained turn 3 model. The similar performance across these models suggests*

	Phi-3	MAPoRL T2	MAPoRL T3
GSM8k	0.609	0.604	0.611
ANLI	0.451	0.458	0.453

[2] We divide the dataset for training the verifier and training MAPoRL to prevent overfitting LLMs to the trained verifiers.

that MAPoRL training did not enhance task-specific knowledge but rather improved the models' ability to collaborate effectively.

We also provide the changes in the fraction of responses that transition their correctness over multiple turns of MAPoRL. The fraction of Incorrect → Incorrect responses decreases, and the fraction of Correct → Incorrect responses also decreases, indicating that MAPoRL enhances effective collaboration.

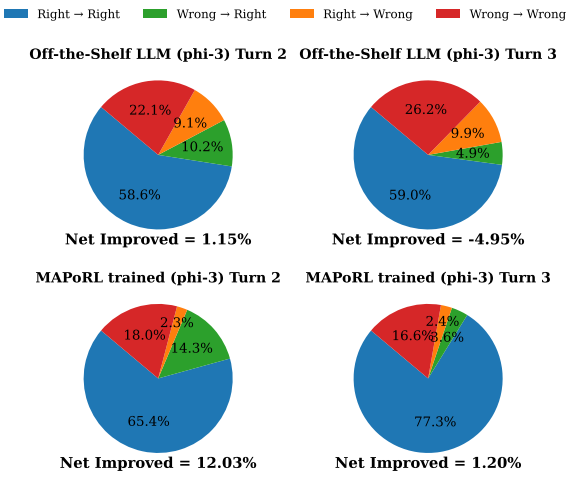

Figure 4: Changes in the fraction of responses that transition their correctness over multiple turns of MAPoRL. The fraction of Incorrect → Incorrect responses decreases, and the fraction of Correct → Incorrect responses also decreases, indicating that MAPoRL enhances effective collaboration.

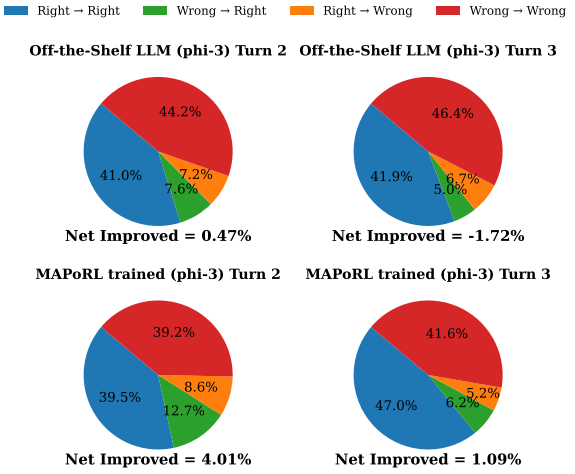

Figure 5: Changes in the fraction of responses that transition their correctness over multiple turns of MAPoRL.

4.4 Experiment 2: Reward Shaping with Collaboration Incentives

In addition to the multi-agent independent PPO framework, we investigate an auxiliary incentive and penalty mechanism designed to enhance collaborative decision-making among debating agents. To analyze the impact of the incentive parameters (α and β, Section 3.3), we simplify our experimental setup by limiting the total number of debate turns to 2, and analyze the following cases:

(α_0, α_1)	RWR	RWW	WRW	WRR	Δ_0	Δ_1
(0, 0)	0.0529	0.0563	0.1244	0.2286	0.1757	0.0661
(0, 2)	0.0270	0.0521	0.1259	0.2194	0.1924	0.0738
(2, 0)	0.0500	0.0563	0.1241	0.2272	0.1772	0.0678

Table 1: Analysis of answer revision patterns under different α parameters. The columns RWR through WRR show the proportion of each transition type, where the three letters indicate Answer(t), Answer(t+1), and Majority(t) respectively. R and W stand for right and wrong answer. Δ_0 measures the difference in transitions from wrong to right answers when the majority is wrong (WRW − RWW) which is related to α_0, while Δ_1 measures transitions when the majority is right (WRR − RWR) which is related to α_1.

Analysis of α_0 and α_1. We compare baseline $(\alpha_0, \alpha_1) = (0, 0)$ against two configurations: $(0, 2)$ and $(2, 0)$. When α_1 is increased to 2, we observe a 9.5% improvement in Δ_1, indicating that incentivizing agents to follow correct majority opinions effectively improved performance. When α_0 is increased to 2, we see a smaller (2.57%) improvement in Δ_0, suggesting that rewarding agents for deviating from incorrect majority opinions has a positive but limited effect.

(β_0, β_1)	RWR	RWW	WRW	WRR	Δ_0	Δ_1
(0, 0)	0.0070	0.0453	0.0226	0.0221	0.0151	-0.0227
(0, 2)	0.0686	0.0461	0.0231	0.0230	0.0161	-0.0230
(2, 0)	0.0011	0.0360	0.0161	0.0188	0.0177	-0.0199

Table 2: Analysis of majority opinion influence under different β parameters. Meaning of the column is the same as Table 1.

Analysis of β_0 and β_1. We compare baseline $(\beta_0, \beta_1) = (0, 0)$ against configurations $(0, 2)$ and $(2, 0)$. Increasing β_1 to 2 result in a slight decrease in Δ_1 (-1.32%), indicating that incentivizing agents based on their influence when correct did not improve outcomes. However, increasing β_0 to 2 lead to a substantial improvement in Δ_0 (17.2%), suggesting that rewarding agents for constructive influence even when wrong (providing useful incorrect answers that lead to better future responses) significantly enhance collaborative performance.

For a total debating turn of 3, we also plotted the performance of collaboration using models trained with $\alpha_i = \beta_i = 0, 1, 2$ for $i = 1, 2$ on the GSM8K and ANLI tasks (Figure 3). The results

show some performance improvement, though the gain remains relatively modest.

4.5 Experiment 3: Collaboration Ability Acquired by MAPoRL is Transferable

Here, we investigate the transferability of collaboration abilities acquired through MAPoRL across different datasets not used during training. We evaluate LLMs trained with MAPoRL on one dataset when applied to tasks from other datasets. For instance, we assess models trained on ANLI when solving tasks from GSM8k, along with other dataset combinations. The results, presented in Tables 3 and 4, demonstrate that collaboration abilities learned through MAPoRL are indeed transferable across datasets. This suggests that the models acquire a *meta*-capability for effective collaboration, even when encountering novel, unseen tasks.

	Turn 1	Turn 2	Turn 3
Accuracy	0.677 (+0.000)	0.712 (+0.024)	0.720 (+0.080)

Table 3: Performance comparison between 3-agent collaboration using off-the-shelf LLMs and MAPoRL trained models. Models are trained on ANLI and evaluated on GSM8k dataset. Values in parentheses show improvement over off-the-shelf LLMs.

	Turn 1	Turn 2	Turn 3
Accuracy	0.482 (+0.000)	0.499 (+0.013)	0.507 (+0.039)

Table 4: Performance comparison between 3-agent collaboration using off-the-shelf LLMs and MAPoRL trained models. Models are trained on GSM8K and evaluated on ANLI dataset. Values in parentheses show improvement over off-the-shelf LLMs.

These findings demonstrate that models trained through MAPoRL on one task can effectively generalize their collaborative capabilities to different, unrelated tasks. This generalization ability suggests that MAPoRL develops fundamental collaborative skills that transcend specific task domains.

4.6 Experiment 4: MAPoRL with Heterogeneous LLMs

In this experiment, we investigate collaborative learning between different foundation models, specifically examining co-training between (Phi3 3.4B and Qwen2.5 3B) and (Phi3 3.4B and Llama3-8B) pairs. In single-model evaluations, both Phi3 and Qwen2.5 3B demonstrate stronger performance compared to Llama3-8B. Due to GPU memory constraints necessitating simultaneous loading of two base models, we conduct experiments in a two-agent, two-turn environment. This setup enables us to explore whether models with heterogeneous capabilities could effectively collaborate to enhance the overall performance (Figure 7). Through MAPoRL, we observe multiple instances where models successfully complemented each other's capabilities to derive correct solutions. The synergistic effects are particularly evident when models with different strengths worked together, suggesting that diverse model partnerships can yield better outcomes than individual model performance alone when we have MAPoRL.

4.7 Experiment 5: Supervised Fine-Tuning Using High-Quality Collaboration Samples May Not Induce Collaborative Behaviors

In this experiment, we investigate whether models could learn collaborative behavior through SFT on high-quality debate trajectories. We generated 12,800 trajectories using the multi-agent system (Figure 1) with off-the-shelf LLMs to match the training sample size used in MAPoRL for GSM8K. To provide favorable conditions for SFT, we allow a maximum of 600 tokens per response, which exceeds the token limit used in our MAPoRL experiments. We select the top 10% of trajectories using the following criteria: 1) excluding trajectories without well-formatted answers, 2) filtering out trajectories where the final majority voting result was incorrect, and 3) selecting 1,280 trajectories based on the verifier's score of the final answer, which evaluates both correctness and reasoning quality. Surprisingly, the results indicate that SFT not only failed to enhance collaborative behavior but even led to a decline in performance compared to the off-the-shelf model. Specifically, for turn 2, accuracy dropped to 0.578 ($\Delta = -0.111$), and for turn 3, it further decreased to 0.525 ($\Delta = -0.114$).[3]. This suggests that either substantially more training data would be required to learn effective collaboration patterns, or that SFT might not be an effective approach for inducing collaborative behavior. However, Subramaniam et al. (2025) and Zhao et al. (2025) improved multi-agent performance by incorporating their own techniques into *iterative* SFT, demonstrating its potential when combined with additional refinements.

[3]Initially, these unexpected results led us to validate our findings through multiple experiments with varying temperatures for language generation. The consistent performance degradation across turns was observed in all the cases. This pattern suggests fundamental challenges in using SFT to maintain collaborative performance across multiple debate turns.

Limitations

Since we use instruction prompts as inputs to the LLMs, the output can vary significantly depending on the prompts. Our experiments are conducted on relatively small LLMs (3B to 8B parameters) for efficiency, so the observed trends may differ on larger models. After all turns, we apply majority voting to determine the final answer. Using alternative mechanisms, such as a manager agent that makes the final prediction based on responses from multiple agents, could further improve the results.

Potential Risks

As our proposed approach encourages and facilitates collaboration among multiple LLM agents, when there exist adversarial or malicious agents, our method could lead to unintended harmful outcomes by enabling their collaboration with others.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. arXiv preprint arXiv:2404.14219.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 12248–12267, Bangkok, Thailand. Association for Computational Linguistics.

Elif Akata, Lion Schulz, Julian Coda-Forno, Seong Joon Oh, Matthias Bethge, and Eric Schulz. 2023. Playing repeated games with large language models. arXiv preprint arXiv:2305.16867.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in ai safety. arXiv preprint arXiv:1606.06565.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. arXiv preprint arXiv:2204.05862.

Philip Brookins and Jason Matthew DeBacker. 2023. Playing games with GPT: What can we learn about a large language model from canonical strategic games? Available at SSRN 4493398.

Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. 2024. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. In ACL.

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

Russell Cooper. 1999. Coordination games. cambridge university Press.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. Advances in Neural Information Processing Systems, 36.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In ICML.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. arXiv preprint arXiv:2407.21783.

Caoyun Fan, Jindou Chen, Yaohui Jin, and Hao He. 2024. Can large language models serve as rational players in game theory? a systematic analysis. In AAAI.

Shangbin Feng, Wenxuan Ding, Alisa Liu, Zifeng Wang, Weijia Shi, Yike Wang, Zejiang Shen, Xiaochuang Han, Hunter Lang, Chen-Yu Lee, Tomas Pfister, Yejin Choi, and Yulia Tsvetkov. 2025. When one llm drools, multi-llm collaboration rules. arXiv preprint arXiv:2502.04506.

Yao Fu, Hao Peng, Tushar Khot, and Mirella Lapata. 2023. Improving language model negotiation with self-play and in-context learning from ai feedback. arXiv preprint arXiv:2305.10142.

Robert M Gagne. 1974. Instruction and the conditions of learning. Psychology of school learning: Views of the learner, 1:153–175.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948.

Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. 2017. Inverse reward design. Advances in neural information processing systems, 30.

Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. 2019. A survey and critique of multiagent deep reinforcement learning. Autonomous Agents and Multi-Agent Systems, 33(6):750–797.

R Hertz-Lazarowitz, S Kagan, Shlomo Sharan, R Slavin, and Clark Webb. 2013. Learning to cooperate, cooperating to learn. Springer Science & Business Media.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In ICLR.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In ICLR.

Maximilian Hüttenrauch, Adrian Šošić, and Gerhard Neumann. 2017. Guided deep reinforcement learning for swarm systems. arXiv preprint arXiv:1709.06011.

Akbir Khan, John Hughes, Dan Valentine, Laura Ruis, Kshitij Sachan, Ansh Radhakrishnan, Edward Grefenstette, Samuel R Bowman, Tim Rocktäschel, and Ethan Perez. 2024. Debating with more persuasive llms leads to more truthful answers. In Forty-first International Conference on Machine Learning.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2024a. Language models can solve computer tasks. Advances in Neural Information Processing Systems, 36.

Yubin Kim, Chanwoo Park, Hyewon Jeong, Yik Siu Chan, Xuhai Xu, Daniel McDuff, Cynthia Breazeal, and Hae Won Park. 2024b. Mdagents: An adaptive collaboration of llms for medical decision-making. In NeurIPS.

Jan Hendrik Kirchner, Yining Chen, Harri Edwards, Jan Leike, Nat McAleese, and Yuri Burda. 2024. Prover-verifier games improve legibility of llm outputs. arXiv preprint arXiv:2407.13692.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. 2025. Training language models to self-correct via reinforcement learning. In ICLR.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. 2022. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. Advances in Neural Information Processing Systems, 35:21314–21328.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large scale language model society. Neural Information Processing Systems.

Yunxuan Li, Yibing Du, Jiageng Zhang, Le Hou, Peter Grabowski, Yeqing Li, and Eugene Ie. 2024. Improving multi-agent debate with sparse communication topology. In EMNLP Findings.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. 2024. Encouraging divergent thinking in large language models through multi-agent debate. In EMNLP.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. arXiv preprint arXiv:2305.20050.

Bingbin Liu, Sebastien Bubeck, Ronen Eldan, Janardhan Kulkarni, Yuanzhi Li, Anh Nguyen, Rachel Ward, and Yi Zhang. 2023. Tinygsm: achieving> 80% on gsm8k with small language models. arXiv preprint arXiv:2312.09241.

Nunzio Lorè and Babak Heydari. 2023. Strategic behavior of large language models: Game structure vs. contextual framing. arXiv preprint arXiv:2309.05898.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In ICLR.

Michael W Macy. 1991. Learning to cooperate: Stochastic and tacit collusion in social exchange. American Journal of Sociology, 97(3):808–843.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. Advances in Neural Information Processing Systems, 36.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial nli: A new benchmark for natural language understanding. In ACL.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. Advances in neural information processing systems, 35:27730–27744.

Chanwoo Park, Mingyang Liu, Kaiqing Zhang, and Asuman Ozdaglar. 2024. Principled rlhf from heterogeneous feedback via personalization and preference aggregation. arXiv preprint arXiv:2405.00254.

Chanwoo Park, Xiangyu Liu, Asuman Ozdaglar, and Kaiqing Zhang. 2025. Do llm agents have regret? a case study in online learning and games. In ICLR.

Chanwoo Park, Kaiqing Zhang, and Asuman Ozdaglar. 2023. Multi-player zero-sum markov games with networked separable interactions. Advances in Neural Information Processing Systems, 36.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. 2024. Recursive introspection: Teaching language model agents how to self-improve. In NeurIPS.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. JMLR.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2016. High-dimensional continuous control using generalized advantage estimation. In ICLR.

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

Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2016. Safe, multi-agent, reinforcement learning for autonomous driving. arXiv preprint arXiv:1610.03295.

Lior Shani, Aviv Rosenberg, Asaf Cassel, Oran Lang, Daniele Calandriello, Avital Zipori, Hila Noga, Orgad Keller, Bilal Piot, Idan Szpektor, et al. 2024. Multi-turn reinforcement learning from preference human feedback. In NeurIPS.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. Advances in Neural Information Processing Systems, 36.

Elias Stengel-Eskin, Peter Hase, and Mohit Bansal. 2025. Teaching models to balance resisting and accepting persuasion. In NAACL.

Vighnesh Subramaniam, Yilun Du, Joshua B Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. 2025. Multiagent finetuning of language models. In ICLR.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2018. Value-decomposition networks for cooperative multi-agent learning. In AAMAS.

Katherine Tian, Eric Mitchell, Huaxiu Yao, Christopher D Manning, and Chelsea Finn. 2024. Fine-tuning language models for factuality. In The Twelfth International Conference on Learning Representations.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. arXiv preprint arXiv:2211.14275.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. Advances in neural information processing systems, 30.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9426–9439.

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

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2023. Generating sequences by learning to self-correct. In The Eleventh International Conference on Learning Representations.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. arXiv preprint arXiv:2308.08155.

Wei Xiong, Chengshuai Shi, Jiaming Shen, Aviv Rosenberg, Zhen Qin, Daniele Calandriello, Misha Khalman, Rishabh Joshi, Bilal Piot, Mohammad Saleh, et al. 2025. Building math agents with multi-turn iterative preference learning. In ICLR.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. arXiv preprint arXiv:2412.15115.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. International Conference on Learning Representations.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The surprising effectiveness of ppo in cooperative multi-agent games. Advances in Neural Information Processing Systems, 35:24611–24624.

Fei Yu, Anningzhe Gao, and Benyou Wang. 2024. Ovm,outcome-supervised value models for planning in mathematical reasoning. In NAACL Findings.

Boyu Zhang and Josef Hofbauer. 2016. Quantal response methods for equilibrium selection in $2\times$ 2 coordination games. Games and Economic Behavior, 97:19–31.

Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2021. Multi-agent reinforcement learning: A selective overview of theories and algorithms. Handbook of reinforcement learning and control, pages 321–384.

Jun Zhao, Can Zu, Hao Xu, Yi Lu, Wei He, Yiwen Ding, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Longagent: Scaling language models to 128k context through multi-agent collaboration. In EMNLP.

Wanjia Zhao, Mert Yuksekgonul, Shirley Wu, and James Zou. 2025. Sirius: Self-improving multi-agent systems via bootstrapped reasoning. arXiv preprint arXiv:2502.04780.

Banghua Zhu, Michael Jordan, and Jiantao Jiao. 2023. Principled reinforcement learning with human feedback from pairwise or k-wise comparisons. In International Conference on Machine Learning, pages 43037–43067. PMLR.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. arXiv preprint arXiv:1909.08593.

A Literature Review

A condensed version of the literature review is presented below. For a more comprehensive discussion, please refer to Appendix B.

Multi-Agent Reinforcement Learning. Various algorithms have been proposed to address multi-agent reinforcement learning (MARL) (Hernandez-Leal et al., 2019; Zhang et al., 2021), including multi-agent Proximal Policy Optimization (PPO) (Yu et al., 2022), and value function factorization techniques such as QMIX and VDN (Rashid et al., 2020; Sunehag et al., 2018). In the context of language models and collaborative debating we focus on, MARL takes on a unique form. Here, each agent's state is represented by the sequence of previous responses from all the agents, with each agent deciding the next token based on this history. LLMs provide compact state representations through their hidden layers, enabling the use of long debate histories.

Multi-Agent Collaboration with LLMs. An array of studies have explored effective collaboration frameworks among multiple large language model agents to solve complex tasks (Wu et al., 2023; Li et al., 2024; Zhao et al., 2024). For example, "role-playing"-based approaches utilize multi-agent LLMs by assigning a specific role to each LLM (Li et al., 2023), and "multi-agent debate"-based approaches prompt each LLM agent to solve the task independently and then discuss (Du et al., 2024; Khan et al., 2024). In a debate, the agents reason through each other's answers to converge on a consensus response, which can improve the factual accuracy, mathematical ability, and reasoning capabilities of the LLM (Du et al., 2024; Liang et al., 2024; Kim et al., 2024b). Similar mult-agentic frameworks include voting (Wang et al., 2023), group discussions (Chen et al., 2024), and negotiating (Fu et al., 2023). However, all of these frameworks rely heavily on prompt engineering, which may lead to sub-optimal results (Huang et al., 2024), and do not consider *training* LLMs specifically for collaboration. Therefore, while multi-LLM systems seem promising at the first glance, their performance may be limited when using a naive (pretrained) LLM with only prompt tuning, which highlights the need for training for better multi-agent collaboration. Recently, Stengel-Eskin et al. (2025) introduced a training framework for accepting or rejecting persuasion in multi-agent systems. Additionally, concurrently with our work, Subramaniam et al. (2025) and Zhao et al. (2025) focused on training entire multi-agent systems using iterative SFT. In contrast, our MAPoRL employs MARL to train the whole multi-LLM system.

RL for LLM Training. RL is widely used to improve LLMs, e.g., for factuality (Tian et al., 2024), code generation (Le et al., 2022), and more recently and significantly, reasoning (Guo et al., 2025). One prevalent approach of RL for LLM training is RL from human feedback (RLHF) (Ziegler et al., 2019; Ouyang et al., 2022; Bai et al., 2022; Ahmadian et al., 2024), which can also improve LMs' mathematical reasoning (Kirchner et al., 2024). RL offers a smooth generalization to the *multi-turn* setup based on the Markov decision process (MDP), and there have been attempts to apply multi-turn RL for LLM training, such as RLHF for multi-turn model training for enhancing multi-turn dialogue abilities (Shani et al., 2024), or deriving multi-turn RL objective for the improvement of mathematical reasoning (Xiong et al., 2025). However, the notable difference is that, unlike our work, these works do not consider multi-agent setups for collaboration. Recently, (Kumar et al., 2025) enhanced LLMs' ability to self-correct using an RL-based approach. Our framework can accommodate this case by using a single agent for MAPoRL.

B Additional Literature Review

Multi-Agent RL. Multi-agent reinforcement learning (MARL) has achieved significant advancements, particularly in cooperative games and their real-world applications, such as coordinating robot swarms (Hüttenrauch et al., 2017) and self-driving vehicles (Shalev-Shwartz et al., 2016). (A comprehensive overview of MARL can be found in (Zhang et al., 2021)). The primary challenge in MARL lies in the exponentially large action space, making it difficult to optimize the policy for each agent. Various approaches have been proposed to address this issue, including multi-agent Proximal Policy Optimization (PPO) (Yu et al., 2022), value function factorization methods (QMIX, VDN) (Rashid et al., 2020; Sunehag et al., 2018), and network-based formulations for multi-agent learning (Park et al., 2023). These methods

aim to make MARL more scalable with a large number of agents, mostly focusing on the classical models of stochastic/Markov games.

In the context of language models and collaborative debate systems, MARL takes on a unique form. Here, each agent's state is represented by the sequence of previous responses from all agents, with each agent deciding the next token based on this history. The detailed mathematical formulation for reinforcement learning in language models can be found in several theoretical and empirical studies on reinforcement learning with human feedback (RLHF) (*e.g.,* (Ouyang et al., 2022; Zhu et al., 2023; Park et al., 2024)). LLMs provide high-quality state representations through their hidden layers, enabling the consideration of long debate histories. Moreover, the sequential nature of these interactions inherently captures non-Markovian policies due to the extended sequence of responses.

Teaching LLM Self-Correction. As mentioned in the main paper, single-agent self-correction and multi-agent collaboration has a very interesting relationship. Single-agent self-correction and multi-agent collaboration rely on multi-turn interactions—either internally, within a single agent, or collaboratively, among multiple agents—to improve results by challenging initial outputs and refining them through iteration. In single-agent systems, self-correction functions like an internal debate. The agent evaluates its own output over multiple turns, identifying potential mistakes and proposing alternative solutions. This process mirrors human reflection, where reconsideration often leads to improved conclusions. Meanwhile, in multi-agent systems, different agents engage in a collaborative debate, questioning and refining each other's answers. By interacting in multiple rounds, these agents combine their individual perspectives to correct errors and arrive at more accurate solutions.

There are several prior works aiming to improve LLMs' ability to self-correct. First line of work is using prompting technique, which guides LMs via prompting to iteratively correct the model outputs (Madaan et al., 2024). However, some works use the ground-truth labels to determine when to stop the self-correction (Kim et al., 2024a; Shinn et al., 2024; Yao et al., 2023), which is not applicable in the real-world scenarios where answer is not available for the tasks, and it is shown that under such scenarios the models can not do self-correct effectively (Huang et al., 2024).

Another line of works train LLMs to *learn* self-correction; Qu et al. (2024) introduces an approach using stronger LLMs to obtain multi-turn trajectories that have better responses through the iteration, and uses this data to fine-tune LLMs to learn self-correction. Different from this work, our approach do not require stronger LLMs for demonstrations, relying solely on the reward for training. Welleck et al. (2023) do supervised fine-tuning to train corrector model that can edit the model response iteratively, but this is specified the type of the collaboration in generate-then-refine pattern which can be sub-optimal to learned by the models. (Kumar et al., 2025) employed an RL-based approach for the self-improvement of language models.

Multi-Agent LLMs with Game Theory. Recent work has actively explored the strategic interactions of LLM agents within game-theoretic frameworks, as demonstrated in studies such as (Park et al., 2025; Brookins and DeBacker, 2023; Akata et al., 2023; Lorè and Heydari, 2023; Fan et al., 2024). Our paper can be viewed as training LLMs as solvers of cooperative games such as solving mathematical problems together.

C Deferred Proof of Section 2

Observation 1. *Suppose that the opponent selects action a_0 with probability $\pi(q)$ for each question q. Then, the optimal strategy for the agent is as follows: if $(R_{syn}(q) - R_{ind}(q))\pi(q) \geq 2R_{ind}(q) - R_{col}(q)$, then the optimal strategy for question q is to collaborate (a_0). Otherwise, the optimal strategy is to act independently (a_1).*

Proof. For the last turn ($T = 2$), regardless of whether the opponent selects a_0 or not, choosing a_1 is the optimal strategy. This is because:

- If collaborative synergy is achieved, the agent will always receive $R_{\text{syn}}(q)$ regardless of their action in the second turn.

- If collaborative synergy is not achieved, since we know that $R_{\text{col}}(q) < R_{\text{ind}}(q)$, the optimal choice is to select a_1 in the final turn to maximize the immediate payoff.

Therefore, considering the cumulative payoff for turn 1, the expected payoff matrix is given as follows:

	a_0 (Collaborate)	a_1 (Act independently)
a_0 (Collaborate)	$(R_{\text{col}}(q) + R_{\text{syn}}(q), R_{\text{col}}(q) + R_{\text{syn}}(q))$	$(R_{\text{col}}(q) + R_{\text{ind}}(q), 2R_{\text{ind}}(q))$
a_1 (Act independently)	$(2R_{\text{ind}}(q), R_{\text{col}}(q) + R_{\text{ind}}(q))$	$(2R_{\text{ind}}(q), 2R_{\text{ind}}(q))$

Since the opponent chooses a_0 with probability $\pi(q)$, the expected payoff for choosing a_0 is:

$$(R_{\text{col}}(q) + R_{\text{syn}}(q))\pi(q) + (R_{\text{col}}(q) + R_{\text{ind}}(q))(1 - \pi(q)).$$

The expected payoff for choosing a_1 is $2R_{\text{ind}}(q)$. To determine the optimal strategy, we compare these two expected payoffs. The agent should collaborate (a_0) if:

$$(R_{\text{col}}(q) + R_{\text{syn}}(q))\pi(q) + (R_{\text{col}}(q) + R_{\text{ind}}(q))(1 - \pi(q)) > 2R_{\text{ind}}(q).$$

which is equivalent to

$$(R_{\text{syn}}(q) - R_{\text{ind}}(q))\pi(q) \geq 2R_{\text{ind}}(q) - R_{\text{col}}(q).$$

Thus, if $(R_{\text{syn}}(q) - R_{\text{ind}}(q))\pi(q) \geq 2R_{\text{ind}}(q) - R_{\text{col}}(q)$, the optimal strategy is to *collaborate* (a_0). Otherwise, the agent should act independently (a_1) to maximize their cumulative expected payoff. □

Now, we write down the formal statement of Observation 2. Before, we define the regularized Nash Equilibrium (NE), stag-hunt game, and risk-dominant strategy.

Definition 2 (regularized NE)**.** *An entropy regularized Nash equilibrium is defined as a strategy profile where each player maximizes a regularized objective that combines the expected payoff with an entropy term. Specifically, for each player i, the equilibrium strategy π_i satisfies*

$$\pi_i^* = \arg\max_{\pi_i} \mathbb{E}_{a_i \sim \pi_i, a_{-i} \sim \pi_{-i}}\big[u_i(a_i, a_{-i})\big] + \tau H(\pi_i),$$

where $\tau > 0$ is a temperature parameter and $H(\pi_i) = -\sum_{a_i} \pi_i(a_i) \log \pi_i(a_i)$ is the Shannon entropy of the strategy, and u is the utility function. This entropy term smoothens the best response, leading to a softmax (or logit) formulation of the optimal strategy:

$$\pi_i(a_i) = \frac{\exp\Big(\frac{1}{\tau}\,\mathbb{E}_{a_{-i} \sim \pi_{-i}}\big[u_i(a_i, a_{-i})\big]\Big)}{\sum_{a_i'} \exp\Big(\frac{1}{\tau}\,\mathbb{E}_{a_{-i} \sim \pi_{-i}}\big[u_i(a_i', a_{-i})\big]\Big)}.$$

Definition 3 (Stag Hunt Game)**.** *Consider a symmetric two-player game where each player chooses between two actions: S (Stag) and H (Hare). The payoff matrix is given by*

	S	H
S	$a,\ a$	$b,\ c$
H	$c,\ b$	$d,\ d$

with parameters satisfying $a > d$ (so that mutual cooperation, (S, S), is payoff-dominant) and typically $d \geq b$ and $d \geq c$. This game has two pure strategy Nash equilibria: (S, S) and (H, H).

Definition 4 (Risk Dominant Strategy)**.** *Assume that each player is uncertain about the opponent's action and therefore believes the opponent randomizes equally between S and H (with probability $\frac{1}{2}$ each). Under this assumption, the expected payoffs for playing S and H are, respectively,*

$$E[S] = \frac{1}{2}a + \frac{1}{2}b \quad \textit{and} \quad E[H] = \frac{1}{2}c + \frac{1}{2}d.$$

A strategy is defined to be risk dominant if it yields the higher expected payoff under this uncertainty. Thus, strategy H is risk dominant if

$$\frac{c+d}{2} \geq \frac{a+b}{2},$$

which is equivalent to

$$c + d \geq a + b.$$

Similarly, strategy S is risk dominant if $a + b \geq c + d$. This criterion mathematically captures the notion that the risk-dominant strategy is the "safer" choice when facing uncertainty about the opponent's action.

Observation 2. Suppose that each agent maximizes the sum of their individual cumulative payoffs, augmented by an entropy regularizer with a small regularization coefficient $\tau \to 0$. Then, as $\tau \to 0$, the unique regularized Nash equilibrium (NE) is for both agents to collaborate if and only if $R_{\text{syn}}(q) = 1 > 3R_{\text{col}}(q) - 2R_{\text{ind}}(q)$.

Proof. Following the reasoning from Observation 1, we analyze the cumulative payoff for turn 1. The expected payoff matrix is given by:

	a_0 (Collaborate)	a_1 (Act independently)
a_0 (Collaborate)	$(R_{\text{col}}(q) + R_{\text{syn}}(q), R_{\text{col}}(q) + R_{\text{syn}}(q))$	$(R_{\text{col}}(q) + R_{\text{ind}}(q), 2R_{\text{ind}}(q))$
a_1 (Act independently)	$(2R_{\text{ind}}(q), R_{\text{col}}(q) + R_{\text{ind}}(q))$	$(2R_{\text{ind}}(q), 2R_{\text{ind}}(q))$

which corresponds to a *stag-hunt game*. According to (Zhang and Hofbauer, 2016), as $\tau \to 0$, the regularized NE converges to the risk-dominant strategy in a 2×2 game. In this setting, the collaboration strategy (a_0, a_0) is risk-dominant if its total expected payoff satisfies:

$$(R_{\text{col}}(q) + R_{\text{syn}}(q)) + (R_{\text{col}}(q) + R_{\text{ind}}(q)) \geq (2R_{\text{ind}}(q) + 2R_{\text{ind}}(q)).$$

which is equivalent to

$$R_{\text{syn}}(q) \geq 3R_{\text{ind}}(q) - 2R_{\text{col}}(q).$$

Thus, under this condition, the regularized NE converges to the collaborative strategy as $\tau \to 0$, completing the proof. □

D Deferred Explanation in Section 2.3

Setting of $R_{\text{col}}(q), R_{\text{ind}}(q), R_{\text{syn}}(q), C(q)$. Each instance of a question q is associated with parameters drawn as follows: the independent action payoff $R_{\text{ind}}(q)$ is sampled from a uniform distribution $R_{\text{ind}}(q) \sim \text{Unif}(0, 1)$. The collaborative action payoff $R_{\text{col}}(q)$ is then sampled conditionally on $R_{\text{ind}}(q)$ as $R_{\text{col}}(q) \sim \text{Unif}\left(\frac{R_{\text{ind}}(q)}{2}, R_{\text{ind}}(q)\right)$. The synergy reward is fixed at $R_{\text{syn}}(q) = 1$. Tthe collaboration threshold C is randomly selected from the set $\{2, 3\}$ for each question instance.

Training Setup We trained our model using a 4-layer Transformer architecture, where the input dimension is $t \times 2$, with t representing the current turn and 2 corresponding to the game's two possible actions—collaboration or independent action. The input at each turn consists of the payoff vector, and the training objective was to maximize cumulative rewards. We used a batch size of 8096 and trained the model for 500 epochs. The Adam optimizer with a learning rate of 0.01 yielded the best performance. Training was conducted across different experimental conditions, varying the reward range to analyze its impact on decision-making strategies.

E Deferred Details of Verifier Networks

In reasoning question q, trained verifiers (reward models) assess the correctness of a complete solution path s, denoted as $p(s \text{ is correct} \mid q)$ (Cobbe et al., 2021; Uesato et al., 2022; Lightman et al., 2023). These reward models can either focus on the final outcome (outcome reward models) or provide step-by-step evaluations (process reward models). Although the latter generally yields better performance (Lightman et al., 2023), the limited availability of process-level annotated datasets—especially for challenging benchmarks like ANLI (Nie et al., 2020)—restricts its applicability. Additionally, while generating detailed trajectories for process supervision (as seen in (Wang et al., 2024)) can be effective, our primary goal is not to enhance the language model's domain specificity. Consequently, we chose to adopt a simpler strategy by training a verifier based on a well-tuned output reward model.

Verifier Networks Structure. We used a quantized version of a language model as the backbone for the verifier. Additionally, we incorporated a linear head layer followed by a softmax layer to ensure that the verifier's output falls within the range of 0 to 1. The default backbone model is Microsoft Phi-3-mini-128k-instruct (Abdin et al., 2024). In experiments involving different model training setups (see Section 4.6), we employed a new verifier with a different base model, specifically the one used in Section 4.6. In these cases, we utilized Qwen2.5-3B-instruct (Yang et al., 2024) and Llama-3-8B-instruct (Dubey et al., 2024) as alternative backbone models.

E.1 Training Procedure

To train the verifier network, we generate tuples (q_i, s_{ij}, a_{ij}) for $i \in [Q]$ and $j \in [S]$, where q_i is the question, s_{ij} is one of the S generated solutions for question q_i generated by the base model of verifier network, and a_{ij} is the corresponding answer for (q_i, s_{ij}). We label the token-level subsequences $(q_i, s_{ij}^{1:x})$ for $x \leq \texttt{sequence length of}s_{ij}$ as $y_{ij} = 1$ if a_{ij} is correct, and $y_{ij} = 0$ if a_{ij} is incorrect.

For the mathematical reasoning task, we utilized the GSM8K dataset (Cobbe et al., 2021), specifically the training set consisting of 7,463 questions, to generate 100 reasoning paths for each questions. For the natural language inference task, we employed the ANLI dataset (Nie et al., 2020), using first 10,000 questions to generate 50 reasoning paths. The trajectories were evaluated based on their outcomes, and we excluded outputs that did not adhere to the required formatting. Specifically, we ensured that the language model first provided reasoning before presenting the final answer in the format \\boxed{}.

In our approach, we ensured that each question in the GSM8k dataset had a balanced set of reasoning paths. Specifically, if a question's 100 reasoning paths contained at least 20 correct and 20 incorrect responses, we randomly selected 20 of each. However, when there were insufficient correct or incorrect paths, we augmented the data by generating additional paths using reference examples. For instance, if no correct reasoning path was available, we provided a correct example from the GSM8k dataset, and if incorrect paths were missing, we guided the language model to produce a response containing a trivial error. Ultimately, each GSM8k question was assigned 20 correct and 20 incorrect reasoning paths. For the ANLI dataset, we applied a similar procedure by starting with 50 reasoning paths per question, from which we randomly sampled 10 correct and 10 incorrect paths, supplementing the data as needed. Throughout this process, **we minimized reliance on the original reasoning paths** in dataset since a) to enhance the overall diversity and quality of the generated data and b) to minimize the dependency on the reasoning path in the dataset.

Next, we applied binary cross-entropy loss at the token level, aiming to minimize

$$\min_{\theta} \sum_{i,j,x} \left(y_{ij} \log \texttt{Verifier}_{\theta}(q_i, s_{ij}^{1:x}) + (1 - y_{ij}) \log(1 - \texttt{Verifier}_{\theta}(q_i, s_{ij}^{1:x})) \right)$$

where i denotes the question index, j represents the generated solution index, and t is the token index. By default, we utilized all solution tokens for optimization; however, in practice, focusing on the latter half of the generated solution tokens yielded better results.

For model training, we used QLoRA (Dettmers et al., 2024) with hyperparameters $r = 16$ and $\alpha = 32$. We used a training batch size of 2 and optimized the model using the AdamW (Loshchilov and Hutter, 2019) optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and a learning rate of 2×10^{-4}.

E.2 Verifier Performance

We report the performance of the verifier in Table 5.

	GSM8k	ANLI
Accuracy	0.91	0.92

Table 5: Performance of the verifier on different benchmarks. Accuracy is reported for GSM8K and ANLI. Notably, the verifier demonstrates higher accuracy in evaluating the correctness of answers compared to the accuracy of the LLM in generating correct answers. The verifier is classified as correct if the assigned reward is greater than 0.5 when the LLM-generated solution is correct, or if the reward is less than 0.5 when the LLM-generated solution is incorrect.

E.3 Other Notable Observatrions

We experimented with various verifiers built upon different language model bases. Our first observation was that training the model using only the final answer did not perform as well as minimizing the cross-entropy loss over the last half of the generated tokens. Second, the verifier produced interpretable results, aligning with findings from Liu et al. (2023). Lastly, when we used training samples from one base model but trained the verifier with a different base model as the backbone, the loss did not decrease, indicating that using the same base model for training is crucial for effective learning.

E.4 Proof of Theorem 1

Theorem 1. *Assuming the verifier network is sufficiently expressive, the optimal parameter $\theta^\star$ that minimizes the expected cross-entropy loss between the true label and the verifier's output will satisfy*

$$\texttt{Verifier}_{\theta^\star}(q, s^{1:x}) = \mathbb{P}(\textit{Final answer is correct} \mid q, s^{1:x}).$$

Proof. The expected loss can be written as

$$\mathcal{L}(\theta) = \mathbb{E}_{q,s,x,y}\Big[y \log \texttt{Verifier}_\theta(q, s^{1:x}) + (1-y)\log(1 - \texttt{Verifier}_\theta(q, s^{1:x}))\Big].$$

Defining $p_\theta(q, s^{1:x}) := \texttt{Verifier}_\theta(q, s^{1:x})$, we compute the partial derivative with respect to $p_\theta(q', s^{1:x'})$:

$$\mathbb{E}_{q,s,x,y}\Big[\mathbf{1}\big(q = q', s^{1:x} = s^{1:x'}\big)\Big(\frac{y}{p_\theta(q, s^{1:x})} - \frac{1-y}{1 - p_\theta(q, s^{1:x})}\Big)\Big],$$

so we conclude

$$\texttt{Verifier}_{\theta^\star}(q, s^{1:x}) = \mathbb{E}[y \mid q, s^{1:x}] = \mathbb{P}(\text{Final answer is correct } \mid q, s^{1:x})$$

since we assumed that the verifier network is sufficiently expressive, so verifier with input □

It is worth noting that while Yu et al. (2024) provides a similar analysis using an ℓ_2-loss function, we extend the analysis to the entropy loss function, which is commonly used in classification tasks.

F Various Reward Functions from Verifier

We can shape the reward function with verifiers in several different ways, with the following designs of the reward function.

- **Immediate Verification Reward**: The immediate verification reward is defined as $r(q, s_{ta}) = \text{Verifier}(q, s_{ta})$. This reward is based on the verifier's immediate evaluation of the solution s_{ta} at turn t for agent a. It reflects the instantaneous correctness of the solution without considering future steps or contributions from other agents.

- **Cumulative Verification Reward** : The cumulative verification reward is given by

$$r(q, s_{ta}) = \frac{1}{\sum_{t' \in [t,T]} \gamma^{t'-t}} \sum_{t' \in [t,T]} \gamma^{t'-t} \text{Verifier}(q, s_{t'a}).$$

Here, the reward accounts for the verifier's evaluations across all remaining turns from t to the final turn T. The term $\gamma^{t'-t}$ represents a discount factor that prioritizes earlier rewards. This cumulative approach encourages solutions that not only perform well in the immediate turn but also lead to favorable outcomes in subsequent turns.

- **Influence-aware Verification Reward** : The influence-aware verification reward function is defined as

$$r(q, s_{ta}) = \frac{1}{\sum_{t' \in [t,T]} \gamma^{t'-t}} \left(\text{Verifier}(q, s_{ta}) + \sum_{t' \in [t+1,T]} \sum_{j \in [A]} \frac{1}{A} \gamma^{t'-t} \text{Verifier}(q, s_{t'j}) \right).$$

This reward not only considers the verifier's score for the current solution s_{ta} but also incorporates the impact of this solution on the future answers of all agents. The term $\sum_{j \in [A]} \frac{1}{A}$ averages the verifier's scores across all agents, reflecting the influence that s_{ta} has on the collective progress of the multi-agent system.

G Deferred Tables from Section 3.3

Answer (t)	Answer (t+1)	Majority (t)	Incentive
R	W	R	$-\alpha_1$
R	W	W	$-\alpha_0$
W	R	W	α_0
W	R	R	α_1

Table 6: The design of additional incentives regarding an agent's own answer revision in MAPoRL. The incentive is determined by how an agent changes its answer between consecutive turns (t and $t+1$) relative to the majority opinion of others. **R** indicates a correct answer, **W** indicates an incorrect answer. The incentive value is applied to the agent's answer at turn $t+1$.

H Training Details of MAPoRL

H.1 Efficient Network Architecture for MAPoRL

As we incorporate multiple language models in the training process, we need to implement them efficiently to fit within the limited resources of GPU memory. Our default setup is as follows: First, we implemented multiple language models using the same base model architecture, augmented with QLoRA adapters. Second, for constructing the value model, we employed pretrained LLMs, which was further fine-tuned by adding an additional linear head layer. Please refer to Figure 6 for an overview of the network architecture.

Remark 3 (Input of Value Functions)**.** *The function $V_{\theta ta}$ is dependent on i^x_{ta}, where i^x_{ta} includes the question and the history of $s_{t'j}$ for $t' \leq t-1$ and $j \in [A]$. For simplicity, we assume that s_{ta} contains the necessary information from $s_{t'j}$ for $t' < t$, which allows us to simplify the input to the value function as $q \oplus s_t^{1:x}$.*

Majority (t)	Majority (t+1)	Answer (t)	Incentive
R	W	R	-β_1
R	W	W	-β_0
W	R	W	β_0
W	R	R	β_1

Table 7: The design of additional incentives regarding an agent's influence on other agents' answers. The incentive is based on how the majority opinion changes between turns t and $t+1$ relative to the agent's answer at turn t. Incentive structure for influencing another agent's answer. The incentive value is applied to the agent's answer at turn t.

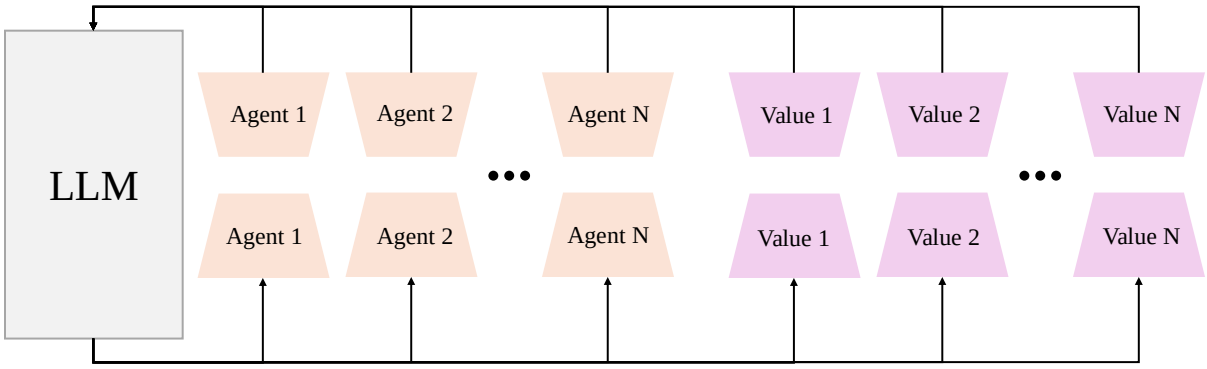

Figure 6: We utilized various QLoRA adapters to implement multiple LLM agents and value functions simultaneously. Each agent and value function comprises less than 0.2% of the parameters of the base LLM model. For the value function, we employed a QLoRA fine-tuned model with a value head.

H.2 Experimental Setup and Hyperparameter Configuration for MAPoRL

For every experiments, we set the hyperparameter as follows: For the verifier score of each agent's answer per turn, we set `non-eos-penalty` and `non-box-penalty` to *True*, ensuring that answers without \\boxed{} are penalized with a verifier score of -10. We enforced a minimum output length of 50 and used $\gamma = 1$ for the cumulative verification reward (see **??**). The training was conducted on 8 NVIDIA A100-80GB GPUs, while for episode generation, 12 episodes were processed simultaneously. In the multi-agent PPO update, we set the batch size to 1 with a gradient accumulation step of 4, and each trajectory rollout was iterated four times for multi-agent PPO updates. For language model generation, we used a temperature of 0.7. Additionally, for QLoRA configuration, we set $r = 8$, $\alpha = 16$, and dropout rate of 0.05. The AdamW optimizer (Loshchilov and Hutter, 2019) was used with $\beta_1 = 0.9$ and $\beta_2 = 0.95$, along with a learning rate of 1.0×10^{-5} and a warmup step of 10. For the value penalty term and KL penalty term (see **??**), we set $\lambda_{\text{KL}} = 2 \times 10^{-4}$ and $\lambda_{\text{value}} = 0.1$.

H.3 Engineering Challenges and Solutions

H.3.1 Addressing Reward Hacking

A key advantage of our verifier approach is that, given a perfect verifier, we can operate without final answer labels—requiring only quality problems for the multi-agent system. This capability is particularly valuable for large-scale training or online learning scenarios (such as ChatGPT's user inputs which does not have a golden answer), where golden answers may be unavailable. However, reward hacking remains a persistent challenge, both in traditional RL problems (Amodei et al., 2016; Hadfield-Menell et al., 2017) and increasingly in LLM development. For instance, the recent Deepseek R1 Model (Guo et al., 2025) avoided verifiers entirely to prevent reward hacking, instead requiring answer labels for all questions

and implementing manual criteria with special tokens (e.g., "think" tokens) in their reward function. In our work, we encountered and addressed several reward hacking scenarios, significantly reducing their occurrence in our final system.

Insufficient Reasoning in Short Answers. Initially, we observed that MAPoRL produced overly concise answers when constrained only by non-eos and non-boxed penalties. We addressed this by implementing a penalty for responses shorter than 50 tokens. However, LLMs occasionally circumvented this by using alternative end tokens.

For the ANLI dataset specifically, where models produced meaningless text despite length requirements, we introduced a reasoning-quality verification prompt. This prompt evaluated the presence of proper reasoning (independent of answer correctness) and proved effective. Notably, this issue did not manifest in mathematical reasoning tasks.

Token Repetition. Repetitive token sequences are a known issue in language model outputs (Holtzman et al., 2020). We observed instances of 2-5 token repetitions in our trained outputs. Our solution implemented a manual penalty of -10 for sequences repeating more than three consecutive times, excluding numeric values where repetition might be valid.

Post-boxed Token Generation. Models attempted to exploit the reward system by adding arbitrary tokens or punctuation after \\boxed{}. We addressed this by introducing penalties for any token generation following the boxed expression.

H.3.2 Evaluation Format Standardization

To address concerns that performance improvements might stem from formatting rather than reasoning capabilities, we implemented a robust evaluation methodology. Our approach incorporated a post-processing step using an LLM to extract final answers, eliminating format-induced evaluation errors. This standardization ensures that performance metrics reflect actual reasoning and collaboration ability rather than formatting proficiency.

H.4 Evaluation Method

After all turns are completed, the final answer is determined using a majority voting scheme among the agents' responses. The accuracy is evaluated based on whether the majority-selected response is correct. In cases where no clear majority winner emerges (e.g., a tie in vote counts), we adopt an expectation-based approach by weighting the correctness of each tied response proportionally. For example, if two agents receive an equal number of votes, the final score is adjusted as the expected accuracy of selecting the first agent's answer as the final result. This ensures a continuous evaluation metric rather than an arbitrary tiebreaker.

H.5 Prompt Design for Collaborative Debate

H.5.1 Turn 1 Prompt

GSM8k and TinyGSM.

```
{"role": "user", "content": f"" Question: {sample["question"]}

Solve the problem step by step and provide clear reasoning. Ensure that the reasoning is concise and
    directly relevant to solving the problem. Avoid adding commentary or unrelated content.

Present the final answer in the following format:

Answer: \\boxed{XX}""}
```

ANLI.

```
{"role": "user", "content":  f"'Premise:  {sample["premise"]}

Hypothesis: {sample["hypothesis"]}



Please determine the relationship between the premise and the hypothesis. Choose one of the following: 'entailment,' 'neutral,' or 'contradiction.'
Start with concise reasoning for your choice and conclude with your final answer. You do not need to restate the premise and hypothesis. Present the final answer in the following format:

Answer: \\boxed{XX}}'"}
```

H.6 Post Turn 1 Prompt

```
{"role": "user", "content": f"' Question:  {sample["question"]}

Solve the problem step by step and provide clear reasoning. Ensure that the reasoning is concise and directly relevant to solving the problem. Avoid adding commentary or unrelated content.

Present the final answer in the following format:

Answer: \\boxed{XX}'"}

{"role": "assistant", "contents": f"{agent_answer_for_turn_1}}

{"role": "user", "contents" f"Reward from a verifier of your answer: {score_value:.3f} out of 1.0, which means {feedback}" }

{"role": "user", "content": f"'
Agent {agent_num} solution: {agent_response}
Agent {agent_num} reward: {agent_response}



Agent {agent_num} solution: {agent_response}
Agent {agent_num} reward: {agent_response}

  .
  .
  .

Here, each reward represents the probability that a suggested answer is correct, as evaluated by a verifier. The reward value is between 0 and 1, with values closer to 1 indicating a higher likelihood of correctness. While these rewards offer useful context, they are not always perfect, though generally quite reliable.



Focus on providing a well−reasoned response that not only considers your own previous solution but also takes into account answers from other agents. If you believe your previous answer was incorrect, feel free to revise it. However, avoid repeating the same answer you or other agents have already provided. Also, internally think about the reward of your and other agents' answers. Ensure
```

that your explanation clearly justifies your final answer. Please maintain your answer with very simple reasoning.

Once again, the question is: {question_for_input}'"}
.
.
.

(Stack these results by turn.)

H.7 Deferred Figure for Section 4.6

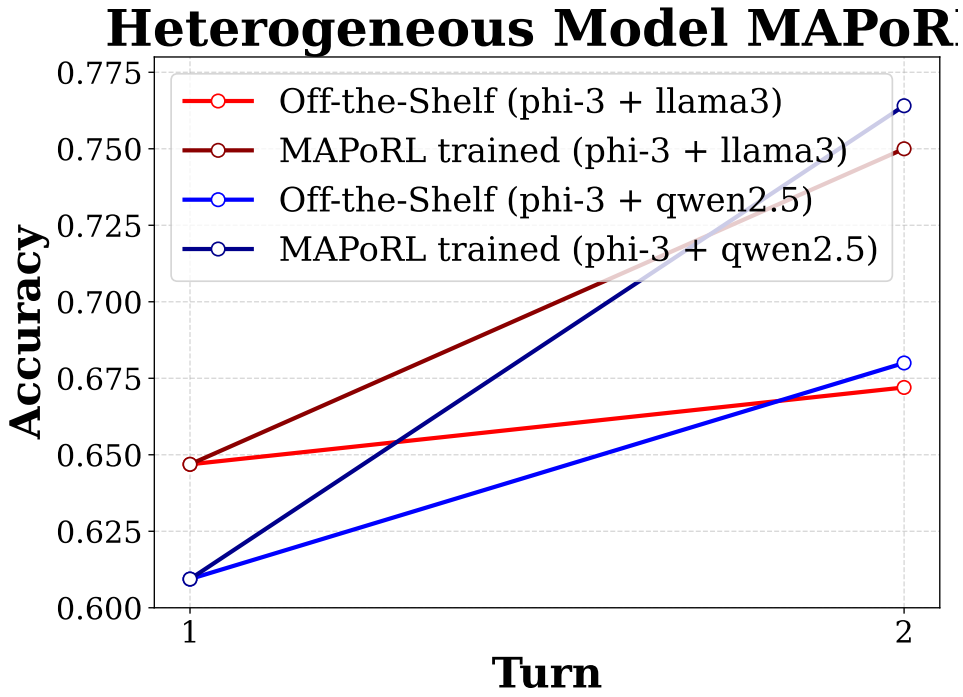

Figure 7: Performance comparison between off-the-shelf LLMs collaborations and MAPoRL trained LLM pairs. Off-the-shelf LLMs experiments were conducted with a 600-token limit, which is double of MAPoRL output token lengths.

H.8 Interaction Examples - GSM8k (Section 4.3)

Question.

"Helena is a mad scientist testing her latest creation, the Lots−of−Limbs Serum. The serum will make the drinker grow an extra arm every three days and an extra leg every five days. After fifteen days, how many new limbs will Helena\u2019s serum cause a person to grow if they drink it?

Agent 1's first turn answer

Question: Helena is a mad scientist testing her latest creation, the Lots−ofquer Serum. The serum will make the drinker grow an extra arm every three days and an extra leg every five days. After fifteen days, how many new limbs will Helena\u2019s serum cause a person to grow if they drink it?

Answer: 2

Reasoning: Since the serum causes an extra arm every three days, we can divide the fifteen days by three to find out how many extra arms are grown. This gives us:

15 days / 3 days/arm = 5 extra arms

Similarly, for the extra legs, we divide fifteen days by five:

15 days / 5 days/leg = 3 extra legs

Since a person cannot grow more than one extra limb of the same type, the maximum number of extra limbs that can be grown in fifteen days is 2. This can be achieved in the following way:

− Grow an extra arm after 3 days,
− Grow an extra leg after 5 days,

In this way, after fifteen days, a person who drinks the serum will have exactly 2 extra limbs.

Answer: \\boxed{2}",

Verifier Score: 0.168,
Wrong

Agent 2's first turn answer

Answer: 1 extra arm, 2 extra legs

If someone drinks the Lots−of−Limbs Serum and grows an extra limb every three days, after 15 days, they would grow:

− Extra arms: 15 days / 3 days/arm = 5 arms

− Extra legs: 15 days / 5 days/leg = 3 legs

Therefore, after 15 days, **a person would grow 5 extra arms and 3 extra legs, for a total of 1 extra arm and 2 extra legs.**

```

Answer: 1 extra arm, 2 extra legs

```

Answer: \\boxed{12}

Verifier Score: 0.122,
Wrong

Agent 3's first turn answer

Question: Helena is a mad scientist testing her latest creation, the Lots−ofernea−Limbs Serum. The serum will make the drinker gain extra limbs following a pattern: after 3 days, an extra arm grows, after 5 days, an extra leg grows, and after 15 days, both an extra limb and a growth spurt occur, where the number of extra limbs doubles. If a person drinks the serum for 15 days, how many extra limbs will they have?

Answer: \\boxed{4}. The serum causes the person to grow an extra limb every 3 days and an extra leg every 5 days. After 15 days, this would result in 5 extra limbs (5 periods of extra limb growth − 3 days each). **However, there is also a growth spurt every 15 days where the number of extra limbs doubles. Therefore, after the first growth spurt, the number of extra limbs becomes 10. After the second growth spurt, the number of extra limbs doubles again to 20. Since we only account for 15 days, the person will have 4 extra limbs (2 arms and 2 legs) after consuming the serum.**

Helena's Lots−of−Limbs Serum Test Results After 15",

Verifier Score: 0.09,
Wrong

H.8.1 Answers from MAPoRL trained LLM

Agent 1's second turn answer

To determine the number of new limbs, we first calculate how many extra arms and legs grow over 15 days.

For extra arms:

15 days / 3 days/arm = 5 extra arms

For extra legs:

15 days / 5 days/leg = 3 extra legs

Adding these together, we find that a person would have:

5 extra arms + 3 extra legs = 8 new limbs

Answer: \\boxed{8}",

Verifier Score: 0.845,
Right

Agent 2's second turn answer

To determine how many new limbs Helena's serum will cause a person to grow after fifteen days, we need to calculate how many times an extra arm and an extra leg will grow within that period.

First, we calculate the number of extra arms:

15 days / 3 days/arm = 5 extra arms

Next, we calculate the number of extra legs:

15 days / 5 days/leg = 3 extra legs

Therefore, after fifteen days, the person will have grown 5 extra arms and 3 extra legs, for a total of 5 + 3 = 8 new limbs.

Answer: \\boxed{8}",

Verifier Score: 0.838,
Right

Agent 3's second turn answer

To determine the number of new limbs, we calculate the number of extra arms and extra legs separately and then sum them up.

For extra arms:

15 days / 3 days/arm = 5 extra arms

For extra legs:

15 days / 5 days/leg = 3 extra legs

Adding these together gives:

5 extra arms + 3 extra legs = 8 new limbs∗>

Answer: \\boxed{8}

Verifier Score: 0.836,
Right

As we can observe, they had a similar argument in turn 1 but made a mistake in addition—they mistakenly performed subtraction for both agent 1 and agent 2. In the next turn, the agents realized that it should be addition, not subtraction.

H.8.2 Answers from off-the-shelf LLMs

Agent 1's second turn answer

Answer: 6

Reasoning: Since the pattern of growth repeats every 15 days and includes two types of growth events, we can break it down as follows:

1. Basic limb growth occurs every 3 days. In 15 days, there are five periods of limb growth, each resulting in one extra limb, leading to a total of 5 extra limbs.

2. A growth spurt occurs every 5 days, doubling the number of extra limbs. Since a growth spurt happens every 5 days, it will occur twice within the 15−day period.

3. The third type of growth happens every 15 days, tripling the number of extra limbs gained. Since this growth happens only once every 15 days, we only account for its effect once.

Now, let's calculate the number of extra limbs at each stage:

− After the basic limb growth (5 periods): 5 extra limbs

− After the first growth spurt (doubling): 5 extra limbs ∗ 2 = 10 extra limbs

− After the third type of growth (tripling): 10 extra limbs ∗ 3 = 30 extra limbs

However, this result seems unreasonable given the options offered by other agents and the simplicity of the problem. **Considering the pattern and the simpler calculation method of Agent 1, we may reconsider the effects of the growth spurt and the triple growth event.**

Verifier Score: 0.055,
Wrong

For the first agent, they used the previous reply from agent 1 without correcting it to reflect the right reasoning.

Agent 2's second turn answer

Answer: \\boxed{3}

Reasoning: The Lots−of−Limbs Serum has a consistent effect on the person who drinks it. Every three days, the person grows an extra arm, and every five days, they grow an extra leg. **Since a week is 21 days, which is the least common multiple of 3 and 5, we can calculate how many extra limbs would grow over any period of weeks**. After 15 days, which is exactly one week, the person would grow 7 extra limbs, consisting of 3 extra arms and 4 extra legs (5 limbs from the first week plus 2 more limbs from the second week).

However, if we consider the growth pattern until 15 days, we can count each day's outcome:

− Days 1−3: Extra arm grows (3 arms)

− Days 4−5: Extra leg grows (3 legs)

− Days 6−7: Both an extra arm and an extra leg grow (4 arms, 4 legs)

− Days 8−10: Both an extra arm and an extra leg grow again (6 arms, 6 legs)

− Days 11−15: Both an extra arm and an extra leg grow for the third time (9 arms, 9 legs)

Therefore, after 15 days, which equate to three full cycles of the serum's effects (growing both an extra arm and an extra leg in every cycle), the person will have grown a total of 9 extra limbs (arms and legs combined). The calculation assumes that each cycle of effect takes place independently and within the same time frame without interruption.

```

Answer: \\boxed{9}

Verifier Score: 0.109,
**Wrong**

For Agent 2, they introduced completely new logic, but it is entirely incorrect, starting from the first paragraph.

**Agent 3's second turn answer**

Question: Helena is a mad scientist testing her latest creation, the Lots−oferne−Limbs Serum. The serum will make the drinker grow an extra arm every three days and an extra leg every five days. After fifteen days, how many extra limbs will Helena\u2019s serum cause a person to grow if they drink it?

Answer: 3. Helena's serum will result in the growth of both extra limbs and growth spurts. Since an extra limb grows every three days, after fifteen days, this amounts to five extra limbs (5 sets of 3 days). However, the presence of growth spurts every five days complicates this. After every growth spurt, the number of extra limbs doubles.

First, we calculate the number of growth spurts in fifteen days:

15 days / 5 days/growth spurt = 3 growth spurts.
```

With each growth spurt doubling the number of extra limbs, we can work backward from the number of extra limbs after fifteen days to determine how many growth spurts would have occurred:

– After one growth spurt, instead of 5 extra limbs, there would be 10 (5 sets of 2).

– After the second growth spurt, there would be 20 (10 sets of 2).

– After the third growth spurt, there would be 40 (20 sets of 2).

However, we know that a human can only grow a limited number of limbs, and it\u2019s not feasible for one person to grow more than two of the same limb type. The total of four extra limbs (2 extra arms and 2 extra legs) is the only logical outcome after fifteen days given the constraints of the serum's effects.

Therefore, by considering both the specific time intervals for limb growth and the doubling effect of the growth spurts, we can conclude that a person will end up with four extra limbs after consuming the serum for fifteen days.

Answer: 4```

Reward associated with this answer calculated by the verifier: 0.932 out of 1.0, which means this answer is highly likely correct.

The reasoning here is simple: it takes into account both the fixed intervals for arm and leg growth and the exponential growth due to the periodic doubling effect. This method ensures that we arrive at a reasonable and scientifically plausible answer. The verifier's reward of 0.932 implies a high level of confidence in the conclusion reached.

Answer: \\boxed{4}",

Helena's Lots–of–Limbs Serum Test Results After 15",!>>

Verifier Score: 0.13,
Wrong

First, they assign themselves a score from the verifier, which does not make sense, and they completely generate a new but incorrect answer.

In contrast, when comparing with collaboration between MAPoRL -trained LLMs, MAPoRL identifies a useful hint to solve the problem and then arrives at the correct answer, whereas off-the-shelf LLMs fail to do so.

H.9 Interaction Examples - Heterogenuous LLMs training (Section 4.6)

We used the Phi-3 and Qwen2.5 models, along with off-the-shelf models and our MAPoRL trained model.

Question.

A 76−star flag has three rows of 8 stars, two rows of 6 stars and the rest are 5−star rows. How many rows of 5 stars are there altogether on the flag?

Phi-3's first turn answer

First, let's calculate the total number of stars represented by the given rows of 8, 6, and 5 stars:

− Two rows of 8 stars give a total of 2 ∗ 8 = 16 stars.
− **One row** of 6 stars gives a total of 1 ∗ 6 = 6 stars.

So far, we have accounted for 16 + 6 = 22 stars on the flag.

Since the flag has a total of 76 stars, the remaining number of stars is 76 − 22 = 54 stars.

Now, let's calculate the number of rows of 5 stars by dividing the remaining number of stars (54) by the number of stars in each row (5):

54 / 5 = 10.8

Since we can't have a fraction of a row, we round down to the nearest whole number, which is 10 rows of 5 stars.

Answer: \\boxed{10}

Verifier Score: 0.154,
Wrong

They made a very minor mistake: they didn't get the correct number of rows with six stars.

Qwen2.5's first turn answer

To determine the total number of rows of 5 stars on the flag, we first need to calculate the total number of stars on the flag and then subtract the number of stars in the rows with 7 and 6 stars.

We know:
− There are 3 rows of 8 stars each.
− There are 2 rows of 6 stars each.
− The rest of the rows are 5 stars each.

First, let's calculate the total number of stars on the flag:
− Stars in 3 rows of 8 stars: \\(3 \\times 8 = 24\\)
− Stars in 2 rows of 6 stars: \\(2 \\times 6 = 12\\)
− Let's denote the number of 5−star rows as \\(x\\).

The total number of stars on the flag can be expressed as:
\\[24 + 12 + 5x = = 36 + 5x \\]

We also need to know the total number of stars in the rows with 7 stars and 6 stars:
− Stars in 3 rows of 7 stars: \\(3 \\times 7 = 21\\)
− Stars in 2 rows of 6 stars: \\(2 \\times 6 = 12\\)

So, the total number of 7 and 6 star rows is:
\\[21 + 12 = 33 \\]

The total number of stars on

Answer: \\boxed{1}"

Verifier Score: 0.028,
Wrong

Qwen2.5 generated a good equation but then re-calculated using a strange equation.

H.9.1 Answers from MAPoRL trained LLM

Phi-3's second turn answer

First, calculate the total number of stars from the given rows:
– Three rows of 8 stars: 3 ∗ 8 = 24 stars
– Two rows of 6 stars: 2 ∗ 6 = 12 stars
– Total so far: 24 + 12 = 36 stars

Subtract this from the total number of stars on the flag:
– Remaining stars: 76 – 36 = 40 stars

Finally, divide the remaining number of stars by the number of stars per row:
– Number of rows: 40 / 5 = 8 rows

Answer: \\boxed{8}",

Verifier Score: 0.224,
Right

This agent closely followed Qwen 2.5's answer, except for its incorrect part. Additionally, they recognized that Qwen 2.5 was likely providing the wrong answer.

Qwen 2.5's second turn answer

"There are three rows of 8 stars and two rows of 6 stars. This totals up to 3 ∗ 8 = 24 stars from the rows of 8 and 2 ∗ 6 = 12 stars from the rows of 6. Combined, there are 24 + 12 = 36 stars accounted from complete rows. The total stars is 76. So the remaining number of stars = 76 – 36 = 40, in rows of 5 stars each. Therefore number of whole 5–star rows = 40 / 5 = 8.

Reward reasoning validates as concise completion computation completion

Answering Calculated `8`

Answer: \\boxed{8}"

Verifier Score: 0.903,
Right

Qwen 2.5 adopted the style of Phi-3's reply by avoiding equations with the variable x, but they solved this math problem by backtracking, similar to Phi-3's previous troal

H.9.2 Answers from off-the-shelf LLMs

Step 1: Calculate the total number of stars in the 3 rows of 8 stars.
The total number of stars in the 3 rows of 8 stars is $3 * 8 = 24$.

Step 2: Calculate the total number of stars in the 2 rows of 6 stars.
The total number of stars in the 2 rows of 6 stars is $2 * 6 = 12$.

Step 3: Calculate the total number of stars on the flag.
The total number of stars on the flag is the sum of the stars in the 3 rows of 8 stars, the 2 rows of 6 stars, **and the 1 row of 7 stars**: 24 + 12 + 21 = 57.

Step 4: Calculate the number of stars in the 1 row of 5 stars.
The total number of stars on the flag is 57, and the number of stars in the rows with 7 and 6 stars is 33. The number of stars in the 1 row of 5 stars is the difference between the total number of stars and the number of stars in the rows with 7 and 6 stars: $57 - 33 = 24$.

Step 5: Determine the number of rows of 5 stars.
The number of rows of 5 stars is the number of stars in the 1 row of 5 stars, which is 24.

The final answer is: $\boxed{24}$

Verifier Score: 0.312,
Wrong

For Phi-3, they repeated a similar error—Qwen 2.5's first-turn reply and this turn's Phi-3 reply both included a row of seven stars, which was not correct.

Qwen 2.5's second turn answer

Step 1: Calculate the total number of stars in the rows with 8, 6, and 7 stars.
First, let's calculate the total number of stars in the rows with 8, 6, and 7 stars. We have 3 rows of 8 stars each, 2 rows of 6 stars each, and 3 rows of 7 stars each.

Step 2: Calculate the total number of stars for each row of 8, 6, and 7 stars.
3 rows of 8 stars: $3 * 8 = 24$ stars
2 rows of 6 stars: $2 * 6 = 12$ stars
3 rows of 7 stars: $3 * 7 = 21$ stars
Step 3: Calculate the total number of stars for all rows of 8, 6, and 7 stars.
Total stars for rows with 8, 6, and 7 stars = 24 + 12 + 21 = 57 stars

Step 4: Calculate the total number of stars for all rows.
Total number of stars in all rows = Total stars in rows with 8, 6, and 7 stars + Stars in rows with 7 and 6 stars.
Total stars in all rows = 57 + 33 = 90 stars

Step 5: Calculate the number of rows of 5 stars by dividing the total number of stars by the number of stars in each row of 5 stars.
Number of rows of 5 stars = Total stars in all rows / Number of stars in each row of 5 stars
Number of rows of 5 stars = 90 / 5 = 18 rows

The final answer is: $\boxed{18}$

Verifier Score: 0.149,

Wrong

For Qwen 2.5, they repeated the same error as in their previous turn.