entropy-regularized Nash equilibrium for $\tau = 0.1$ via backward induction. As shown in Figure 5 (solid red curves), the *jointly* optimized agents coordinate with significantly higher collaboration rates. Intuitively, this is because their learning process, which fosters high collaboration at larger t, shapes their strategic behavior from the very beginning, leading to increased cooperation even at the first turn. These toy results underscore the importance of *strategic interactions*: when both agents adapt their policies simultaneously, they learn to be more collaborative, despite the temptation of short-term independent rewards.

3 Post-Co-Training with MARL

We now provide an overview of our new paradigm of *Multi-Agent Post-Co-Training with RL (*`MAPoRL`*)* for LLM collaboration. In our framework, each agent's response per turn is evaluated by a verifier, which assigns a score reflecting the answer's validity. The reward is defined as the (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 for each turn, ensuring that the learning process incorporates both the performance in the current turn and the influence of the anticipated collaborative interactions in the future turns.

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 to other multi-LLM systems as long as each agent's response can be evaluated—for example, by a verifier that 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 the 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:

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

where $\oplus$ denotes token-wise concatenation, $1 \le t \le T-1$ and $\mathrm{LLM}_i(s)$ 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 a-th agent at turn t. We denote LLM parameterized by θ_{ta} as $LLM_{\theta_{ta}}$.

Next, to implement `MAPoRL`, we define the reward function for the multi-agent RL formulation, using the verifier score. Due to space constraints, we here introduce the *Influence-aware Verification Rewards*, and defer other choices of the reward functions that we will use in the experiments to Appendix F.

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

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

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 the agents, reflecting the influence that s_{ta} has on the overall 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 with the reward function (Definition 1). Thus, instead of single-agent RL, we design a multi-agent RL approach. For this paper, we choose multi-agent PPO (Yu et al., 2022) as a representative multi-agent RL algorithm and instantiate it in the language domain.

Conventional multi-agent PPO extends PPO to multi-agent settings by training decentralized policies, either with a shared critic or through independent learning, where each agent optimizes its policy based on local observations and rewards.

Typically, agents learn independently or with limited coordination mechanisms, such as centralized critics or shared value functions. Our approach adapts multi-agent PPO by defining the state as the concatenation of the multi-agent interaction history, allowing agents to condition their responses on past interactions. Additionally, we introduce reward structures that are aligned but not fully identical across agents, encouraging them to fulfill different roles while collectively working toward solving the task.

Since we are solving multi-turn problems, the value function for each turn's state needs to be defined. The state of each turn's value $V_{ta\boldsymbol{\theta}}$ is the expectation of the cumulated reward conditioned on the input text i_{ta}^x, which is defined as

$$V_{ta\boldsymbol{\theta}}(i_{ta}^x) = \mathbb{E}\left[\sum_{x'=x}^{\texttt{length}(s_{ta})} r_{\boldsymbol{\theta}}(q, s_{ta}^{1:x'}) \,\middle|\, q, i_{ta}^x\right].$$

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

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

where t denotes the turn index and a refers to agent $a \in [A]$, $s_{ta}^{1:x}$ represents the generated token from agent a up to the x-th token in turn t, with $\theta_{\text{ref},ta}$ denoting the parameter of a reference LLM, and $\lambda_{\text{KL}} \geq 0$ is some regularization coefficient. 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 other agents across multiple turns, which makes multi-agent training relevant. 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 parameter 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 $A(i_{ta}^x; \boldsymbol{\theta}, \theta_{vta})$ via GAE. The loss function for multi-agent PPO is then given by: $L_{\text{PPO}}(\theta_{ta}, \theta_{vta}) = L_{\text{Surrogate}}(\theta_{ta}) + L_{\text{Value}}(\theta_{vta})$, where $L_{\text{Surrogate}}(\theta_{ta})$ is defined as

$$\mathbb{E}\Big[\min\Big(R_{ta}^x \, A\big(i_{ta}^x; \boldsymbol{\theta}_{\text{old}}, \theta_{\text{old},vta}\big), \texttt{clip}_\epsilon\big(R_{ta}^x\big) \, A\big(i_{ta}^x; \boldsymbol{\theta}_{\text{old}}, \theta_{\text{old},vta}\big)\Big)\Big].$$

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

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

Here, $\texttt{clip}_\epsilon(\alpha) := \min(\max(1-\epsilon, \alpha), 1+\epsilon)$, $R_{ta}^x = \frac{\text{LLM}_{\theta,ta}(s_{ta}^{x+1} | i_{ta}^x)}{\text{LLM}_{\theta_{\text{old},ta}}(s_{ta}^{x+1} | i_{ta}^x)}$, $\boldsymbol{\theta}_{\text{old}} = (\theta_{\text{old},ta})_{t \in [T], a \in [A]}$ is the parameter used in the rollout for multi-agent PPO, and

$$V_{ta}^{\text{target}}(i_{ta}^x) = V(i_{ta}^x; \theta_{\text{old},vta}) + A(i_{ta}^x; \boldsymbol{\theta}_{\text{old}}, \theta_{\text{old},vta})).$$

The expectation $\mathbb{E}$ is taken over the randomness from

$$q \sim \mathcal{Q}, s_{t'a'} \sim \text{LLM}_{\theta_{\text{old},t'a'}}(q \oplus_{t'' \in [t'-1], j \in [A]} s_{t'j})$$

for all $t' \in [t], a \in [A]$, and $x \sim \text{Unif}([\texttt{length}(s_{ta})])$, where $\mathcal{Q}$ denotes the distribution of questions.

Each agent for each turn optimizes its policy and value function simultaneously, over the parameters $(\theta_{ta}, \theta_{vta})$. These agent interactions among multiple LLMs inherently lead to a multi-agent RL problem, rather than a single-agent RL one, as each agent influences others' 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 a 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' answers. 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 1 and Table 2 to summarize the design of these incentives.

4 Experiments

4.1 Datasets

We evaluate **MAPoRL** on two benchmark NLP tasks to validate its performance in both mathematical

Answer (t)	Answer (t+1)	Majority (t)	Incentive
R	W	R	-α_1
R	W	W	-α_0
W	R	W	α_0
W	R	R	α_1

Table 1: 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 $t+1$.

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 2: The design of additional incentives regarding an agent's influence on other agents' answers in **MAPoRL**. The incentive is based on how the majority opinion changes between consecutive turns t and $t+1$ relative to the agent's answer at turn t. The incentive value is applied to the agent's answer at turn t.

reasoning and logical natural language inference. The details are summarized as follows:

GSM8K (Cobbe et al., 2021) and TinyGSM (Liu et al., 2023). GSM8K is a benchmark dataset designed to assess a model's 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 did not utilize the reasoning processes from GSM8K or TinyGSM but rely solely on their final answers. For training the verifier model, we used 7,463 samples from GSM8K. Additionally, we incorporated 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 used first 10,000 training examples. Furthermore, we used the next 12,800 examples for **MAPoRL** training and 1,200 samples for testing.

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

Evaluation Method. After all turns are completed, the final answer is determined using a majority voting scheme among the agents' responses. The accuracy is 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.

4.2 Models

We primarily use 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 the experiments. Due to computational constraints, we mainly use quantized models and fine-tuned them with QLoRA (Dettmers et al., 2024). We defer the training details to Appendix G. 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 was conducted with two agents collaborating over three turns. An overview of the trained system is provided in Figure 1. In Experiment 1, we trained 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 used 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**-trained model 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,