Orchestrating Multi-Agent Alignment for Large Language Models Through External Memory and Adaptive Reasoning

Jiaxiang Liu
ID: 2024211331
jiaxiang24@mails.tsinghua.edu.cn

Wanlan Ren
ID: 2023210455
rwl23@mails.tsinghua.edu.cn

Abstract

This study presents a pioneering multi-agent framework for aligning large language models (LLMs), reshaping them into autonomous problem-solvers orchestrating external memory and skillfully refining their reasoning through carefully selected exemplars. By transparently storing intermediate outputs outside the model and reintroducing them as inputs in subsequent inference stages, our approach cultivates a flexible, iterative, and agent-driven problem-solving environment. Specialized agents coordinate to evaluate and guide the reasoning process: a dedicated evaluation agent acts as a dynamic reward filter, rigorously assessing interim results and prompting strategic regeneration when necessary. This synergy of autonomous decision-making, external memory integration, and adaptive quality control reduces the burden of manual oversight, while substantially enhancing the clarity, rigor, and reliability of the model's inferences. Together, these innovations define a new paradigm in scalable, high-fidelity LLM alignment, inspiring further exploration into multi-agent orchestration and its far-reaching potential.

1 INTRODUCTION

1.1 Background

Large language models (LLMs), such as GPT and more recent architectures, have revolutionized the field of natural language processing (NLP), enabling breakthroughs in a wide range of tasks, including text generation, machine translation, and question answering. The success of these models can be attributed to their ability to capture vast amounts of knowledge from large-scale text corpora and their ability to generate coherent and contextually appropriate responses based on the input they receive. However, despite these impressive capabilities, LLMs face significant challenges when tasked with complex reasoning, particularly in tasks that require multi-step logical inference, long-term dependencies, and the integration of diverse pieces of information across multiple domains.

While LLMs have been successful at surface-level tasks such as classification or translation, they often struggle with deeper reasoning tasks, where the answer is not directly available from the input but requires deriving conclusions from a chain of reasoning steps[1]. This becomes particularly problematic in scenarios where the model needs to reference multiple pieces of information, remember earlier steps in the process, and ensure logical consistency throughout the reasoning path. For example, tasks such as mathematical word problems, logical puzzles, and scientific hypothesis testing require models to maintain a coherent understanding of the problem over multiple inference steps and ensure that intermediate conclusions are valid and aligned with the final answer.

A critical limitation in current LLMs is their tendency to generate answers without clearly explaining the reasoning behind them. This lack of transparency poses a significant challenge, as users and

Preprint. Under review.

developers may not be able to fully trust the model's results or understand the rationale behind its decisions. Furthermore, while these models are trained to generate responses based on patterns learned from vast datasets, they may fail to account for subtle relationships between facts, errors in logical reasoning, or inconsistencies in long chains of thought. As such, there is a growing demand for more interpretable and robust systems that can not only generate answers but also demonstrate the logical steps taken to arrive at those answers.

Moreover, current models often lack a mechanism for self-correction. If a model generates a flawed answer due to a mistake in its reasoning, there is no immediate feedback loop for the model to self-correct and improve its output. This is especially problematic when it comes to real-time applications where accuracy and correctness are critical, such as in decision-making support systems or educational tools.

The motivation for this research lies in the desire to develop a solution that can overcome these limitations by incorporating multiple agents working collaboratively. These agents would not only generate answers but also assess the reasoning process and refine the outputs through continuous feedback, thus improving both the quality and transparency of the reasoning process. By integrating multiple intelligent agents focused on different aspects of reasoning, evaluation, and regeneration, we aim to create a framework that can handle complex, multi-step reasoning tasks more effectively and reliably. The addition of reinforcement learning algorithms such as Proximal Policy Optimization (PPO) [2]further enhances the system's ability to optimize performance and adapt to feedback, ensuring that the model continues to improve over time.

In this work, we focus on building a multi-agent system capable of handling complex reasoning tasks, with an emphasis on achieving greater transparency, accuracy, and efficiency in the reasoning process. This approach offers the potential to significantly advance the capabilities of LLMs, making them more suitable for a broader range of applications that require deep reasoning and multi-step logical deduction.

1.2 Research Objectives

The main objective of this research is to develop a novel multi-agent system [3] designed to enhance the reasoning capabilities of large language models (LLMs). Traditional LLMs, despite their impressive performance in a variety of tasks, often struggle with complex, multi-step reasoning and maintaining coherence in long chains of thought. In response to these limitations, this study proposes a multi-agent collaborative framework that leverages three core agents: the Reasoning Agent, the Evaluation Agent, and the Regeneration Agent. The Reasoning Agent is responsible for generating an initial answer by performing step-by-step reasoning based on the input query. The Evaluation Agent then assesses the reasoning process, ensuring its logical consistency and accuracy, and provides detailed feedback on any errors or weaknesses in the reasoning chain. Finally, the Regeneration Agent takes this feedback and refines the answer by adjusting the reasoning steps and generating a more accurate and coherent response. This iterative process of reasoning, evaluation, and regeneration is designed to ensure that the output not only meets the required accuracy but is also transparent and interpretable, providing a clear rationale for the generated answer.

In addition to the development of this multi-agent framework, a key objective of the research is to improve the efficiency of the training process using a combination of Proximal Policy Optimization (PPO) and Low-Rank Adaptation (LoRA) techniques. PPO is a reinforcement learning algorithm that enables the optimization of the model's decision-making process through feedback, while LoRA adapters allow for efficient updates to the model parameters, specifically targeting the components that are involved in the reasoning process. This combination aims to optimize the training process by making small, targeted updates to the model, rather than retraining the entire system, which would be computationally expensive and time-consuming. The research seeks to demonstrate that this approach can lead to significant improvements in the model's ability to reason accurately and efficiently, while also ensuring that the model can adapt to a variety of complex tasks and provide scalable solutions for real-world applications. Through this approach, the research aims to not only enhance the reasoning ability of LLMs but also improve the overall transparency, interpretability, and efficiency of the reasoning process, making LLMs more reliable and effective in tasks requiring deep logical inference.

1.3 Importance of the Problem

Traditional large language models (LLMs) have demonstrated impressive capabilities in various natural language processing (NLP) tasks. However, when faced with complex reasoning tasks, these models often exhibit significant limitations in terms of interpretability and robustness. The ability of LLMs to perform multi-step reasoning, maintain logical coherence across different stages of the reasoning process, and produce accurate results remains a challenging problem. Despite their large-scale data training, LLMs sometimes generate reasoning steps or answers that are not only incorrect but also difficult to trace or justify. This lack of transparency and consistency hinders their application in high-stakes scenarios where reasoning and accountability are critical, such as in scientific research, legal reasoning, or decision-making systems.

Furthermore, existing reasoning models typically struggle when dealing with tasks that involve multi-round reasoning, where each step depends on the preceding one. These models often fail to optimize the reasoning process dynamically as new information is incorporated, leading to errors that accumulate over time. As reasoning tasks become increasingly complex, it becomes evident that more sophisticated systems are needed, ones that can not only perform reasoning but also evaluate and improve their reasoning process iteratively. This creates a significant gap in the current state of LLMs and calls for new methodologies that can address these shortcomings effectively.

1.4 Innovations of This Research

This research proposes a novel multi-agent collaborative framework to address the limitations of traditional reasoning models. By introducing three core agents—the Reasoning Agent, the Evaluation Agent, and the Regeneration Agent—this framework aims to generate a more accurate and interpretable reasoning process. The Reasoning Agent will generate an initial set of reasoning steps based on the input question, while the Evaluation Agent will analyze these steps for logical consistency and correctness. If the reasoning process is deemed unsatisfactory, the Regeneration Agent will utilize the feedback from the Evaluation Agent to refine and optimize the answer. This iterative process ensures that errors are detected early and corrected in a transparent and efficient manner.

In addition to the multi-agent framework, this research leverages Proximal Policy Optimization (PPO) as a reinforcement learning technique to enhance the training of the reasoning model. PPO [2]is employed to update the model in a way that maximizes the long-term reward of accurate and consistent reasoning. To make the training process more efficient, LoRA (Low-Rank Adaptation) adapters are introduced to optimize only specific model parameters, particularly those involved in reasoning, without the need to retrain the entire system. This allows for faster and more computationally efficient updates, reducing the resources required for model optimization. By combining these innovations, the proposed system aims to improve both the quality and efficiency of reasoning tasks, making LLMs more reliable and scalable for complex, real-world applications.

2 RELATED WORK

The research in this field is heavily influenced by advances in large language models (LLMs), reinforcement learning (RL), and multi-agent systems (MAS). In this section, we review the key developments in each of these areas and their relevance to our work.

2.1 LLM Reasoning

Large Language Models (LLMs) have revolutionized the field of natural language processing by achieving remarkable performance across a wide range of tasks, from language translation to question answering and text generation. These models, typically based on transformer architectures[4], have demonstrated the ability to handle complex reasoning tasks that involve understanding context, drawing inferences, and generating coherent text.

A key challenge for LLMs is improving their reasoning capabilities, especially for tasks that require multi-step reasoning or the integration of external knowledge. While models like GPT-4 have been shown to excel in generating fluent and contextually relevant responses[5], they often lack the ability to perform complex reasoning over long documents or multi-turn dialogues. Researchers have

explored several strategies to address this limitation, including augmenting LLMs with structured knowledge sources, incorporating attention mechanisms to focus on relevant parts of the input, and using iterative reasoning strategies that refine answers through multiple stages.

Additionally, there has been growing interest in combining LLMs with symbolic reasoning methods. This hybrid approach aims to combine the flexibility and generalization power of neural models with the logical rigor and interpretability of symbolic systems. Some approaches have integrated reasoning engines that perform logical deduction, while others focus on enhancing LLMs with external memory systems to store and retrieve relevant knowledge during reasoning tasks.

Our work builds on these advancements by incorporating an iterative reasoning process within a multi-agent framework, where multiple agents cooperate to refine reasoning steps. The ability to improve reasoning accuracy and transparency through collaboration and feedback is a critical step forward in making LLMs more robust in complex tasks.

2.2 Reinforcement Learning

Reinforcement Learning (RL) is a paradigm in machine learning where an agent learns to make decisions by interacting with an environment[6]. The agent receives rewards or penalties based on its actions and aims to maximize the cumulative reward over time. This approach is particularly useful for sequential decision-making problems, where actions have long-term consequences and immediate feedback is sparse.

A core challenge in RL is ensuring the stability and efficiency of the learning process. This is particularly difficult in high-dimensional spaces or when dealing with complex tasks that require long-term planning. Early RL methods, such as Q-learning and SARSA, relied on value-based methods to estimate the value of different actions. These methods, while effective in simpler environments, often struggle with large state spaces and continuous action spaces.

The introduction of Deep Q-Networks (DQN)[7] was a significant milestone in RL, combining deep learning with traditional Q-learning. This allowed agents to handle high-dimensional input spaces, such as images, and learn directly from raw sensory data. Following this, advancements like Double DQN, Dueling DQN, and Prioritized Experience Replay improved the efficiency and stability of learning by addressing issues like overestimation bias and inefficient exploration.

Another critical development in RL is the advent of Policy Gradient methods, which focus on directly optimizing the policy rather than learning a value function. These methods have been shown to be more effective in handling complex, high-dimensional action spaces, such as those encountered in robotics and natural language processing tasks. One of the most popular Policy Gradient algorithms is Proximal Policy Optimization (PPO)[2], which strikes a balance between simplicity and performance by using a clipped objective function to ensure stable learning.

Our research leverages PPO within a multi-agent framework to improve reasoning accuracy by iteratively refining the agents' actions and decisions. By using RL to guide the optimization of reasoning processes, we aim to enhance the agent's ability to learn from feedback and adapt to complex reasoning tasks.

2.3 Multi-Agent Systems

Multi-Agent Systems (MAS) involve multiple autonomous agents that interact and collaborate to achieve a common goal or to address complex tasks. In these systems, agents can have different roles, knowledge, and objectives, and they communicate or act independently to contribute to the system's overall performance. MAS have been widely applied in areas such as distributed control, robotics, traffic management, and game theory.

One of the primary challenges in MAS is how to coordinate the actions of multiple agents to achieve an optimal outcome. In some cases, agents are in competition with each other, while in other cases, they cooperate to solve a problem. Coordination can be difficult when agents have incomplete knowledge or differing goals, which is why approaches like game theory, negotiation protocols, and distributed planning have been extensively studied in the context of MAS.

Recent research has focused on the integration of RL in multi-agent systems, leading to the development of Multi-Agent Reinforcement Learning (MARL). In MARL, each agent learns its policy

while considering the actions of other agents. One of the main challenges in MARL is dealing with non-stationarity, as the environment changes with each agent's actions. To address this, techniques like Centralized Training with Decentralized Execution (CTDE) and Independent Q-Learning (IQL) have been developed. These methods allow agents to learn collaboratively during training while maintaining the ability to act independently during execution.

The application of MARL to complex, real-world problems has shown promise, particularly in areas like multi-robot systems, autonomous vehicles, and resource management. In these scenarios, agents must coordinate their actions in a dynamic environment, often under constraints such as limited communication or computational resources. Communication among agents plays a crucial role in enhancing coordination and improving the overall system performance. However, efficient communication remains a challenge, especially when dealing with high-dimensional state spaces or continuous action spaces.

Our approach integrates MARL with large language models to facilitate cooperative reasoning between agents. By leveraging the power of LLMs to process and generate natural language, agents in our system can communicate more effectively, share knowledge, and improve the transparency of the decision-making process.

2.4 Integrated Approaches

Recent advancements have explored the integration of LLMs, RL, and MAS to create more intelligent and adaptable systems. For instance, combining RL with LLMs allows agents to learn from textual feedback and improve decision-making based on natural language inputs. Similarly, integrating MAS with LLMs enables agents to collaborate more effectively by sharing reasoning steps and refining outcomes iteratively.

One of the key benefits of these integrated approaches is the ability to improve reasoning accuracy over time. By using RL to optimize reasoning policies and MAS to enable collaborative learning, the system can refine its output based on continuous feedback and improve its performance in complex reasoning tasks. Additionally, the use of LLMs adds an additional layer of abstraction, enabling the system to process and generate explanations in natural language, which enhances the interpretability and transparency of the reasoning process.

The combination of these approaches has the potential to revolutionize domains such as automated question answering, dialogue systems, and autonomous decision-making, where reasoning, communication, and decision optimization are crucial. Our work seeks to contribute to this growing area by proposing a multi-agent system that uses RL and LLMs to iteratively refine reasoning steps, ensuring both accuracy and interpretability.

3 METHOD

3.1 System Architecture

The proposed system leverages a multi-agent architecture designed to enhance the reasoning capabilities of large language models (LLMs) in handling complex, multi-step tasks. This architecture integrates three distinct yet collaborative agents: the *Reasoning Agent*, the *Evaluation Agent*, and the *Regeneration Agent*. Each agent serves a specialized function, but together they work iteratively to improve the overall quality of the reasoning process. The Reasoning Agent generates initial reasoning steps by processing the input question and producing a sequence of logically coherent facts or hypotheses. The Evaluation Agent then assesses the quality of this reasoning path, evaluating its correctness, coherence, and alignment with the input question. The Regeneration Agent utilizes feedback from the Evaluation Agent to refine and optimize the reasoning steps, ensuring that the final output is as accurate and logically sound as possible.

The multi-agent[8] design allows for a dynamic feedback loop, where each agent's output influences the next agent's process, creating a cycle of continuous improvement. This iterative process is particularly beneficial in tasks requiring complex and nuanced reasoning, where a single model might struggle to maintain consistency across multiple steps. Additionally, the system is equipped with an *External Memory* that facilitates knowledge sharing among agents, storing both intermediate results and evaluation feedback to enhance long-term reasoning capabilities. Through this design, the system

is capable of handling a broad spectrum of natural language reasoning tasks with greater accuracy and transparency than traditional single-agent approaches.

3.1.1 Reasoning Agent

The Reasoning Agent is responsible for generating an initial reasoning path in response to a given input question. It uses a deep learning model, such as a transformer-based architecture, to generate a sequence of reasoning steps or intermediate facts that are logically connected. The output from this agent is a sequence of reasoning steps $R = (r_1, r_2, \ldots, r_n)$, where each r_i represents a reasoning step in the process.

Formally, the reasoning agent produces the reasoning path R given an input question q:

$$R = \text{ReasoningAgent}(q)$$

The generated reasoning path R is evaluated by the Evaluation Agent to assess the correctness and relevance of the generated reasoning.

3.1.2 Evaluation Agent

The Evaluation Agent's primary function is to evaluate the quality of the reasoning path produced by the Reasoning Agent. It assesses the logical consistency, factual correctness, and coherence of the reasoning steps. This evaluation is carried out through a reward function r_{eval}, which quantifies the quality of the reasoning process. The Evaluation Agent provides feedback based on the reward function, which influences the subsequent optimization of the reasoning process.

Let Q denote the quality of the reasoning path, and let $r_{\text{eval}}(R)$ be the reward function applied to the reasoning path R. The Evaluation Agent computes the reward as:

$$r_{\text{eval}}(R) = f_{\text{eval}}(R, q)$$

where f_{eval} is a scoring function that evaluates the reasoning path R based on its coherence and correctness relative to the input question q.

The Evaluation Agent's feedback is used to guide the Regeneration Agent in refining the reasoning path, ensuring the output is more accurate and coherent.

3.1.3 Regeneration Agent

The Regeneration Agent is responsible for refining and improving the reasoning path generated by the Reasoning Agent. Upon receiving the feedback [9]from the Evaluation Agent, the Regeneration Agent updates the reasoning steps based on the provided reward signal. This process is essential for optimizing the quality of the reasoning steps and ensuring that the final output is more aligned with the correct answer.

Mathematically, the Regeneration Agent updates the reasoning path based on the feedback $r_{\text{eval}}(R)$ as follows:

$$R_{\text{new}} = \text{RegenerationAgent}(R, r_{\text{eval}}(R))$$

This update can be viewed as an optimization process, where the goal is to minimize the difference between the current reasoning path R and the ideal reasoning path that would lead to a correct and coherent answer.

3.1.4 External Memory System

The multi-agent system is supported by an *External Memory* system that acts as a shared storage between the Reasoning, Evaluation, and Regeneration Agents. The memory stores past reasoning steps, evaluations, feedback, and the resulting optimized paths. This memory system plays a crucial role in ensuring that the agents can learn from past experiences and improve over time[10].

At each iteration, the reasoning steps generated by the Reasoning Agent, the corresponding evaluation scores from the Evaluation Agent, and the updated reasoning path produced by the Regeneration Agent are stored in the External Memory. The memory allows the system to retain important contextual information, which is essential for ensuring continuity and context preservation during the reasoning process.

Let M denote the external memory, and at each time step t, the memory is updated with the latest reasoning steps and feedback:

$$M_t = M_{t-1} \cup \{R_t, r_{\text{eval}}(R_t), R_{\text{new},t}\}$$

where M_t represents the memory at time t, R_t is the reasoning path at time t, and $r_{\text{eval}}(R_t)$ is the evaluation score of the reasoning path.

The External Memory system is critical for enabling error correction and improving the system's ability to handle complex, multi-step reasoning tasks by storing historical interactions that can be referenced during subsequent learning stages.

3.1.5 Interaction Between Agents

The interaction between the Reasoning, Evaluation, and Regeneration Agents can be viewed as a feedback loop. The Reasoning Agent generates reasoning paths, which are evaluated by the Evaluation Agent. Based on the evaluation feedback, the Regeneration Agent refines the reasoning process, which is then reassessed in the next iteration.

This iterative process can be formalized as follows:

$$R_t = \text{ReasoningAgent}(q_t), \quad r_{\text{eval}}(R_t) = f_{\text{eval}}(R_t, q_t), \quad R_{\text{new},t} = \text{RegenerationAgent}(R_t, r_{\text{eval}}(R_t))$$

where R_t is the reasoning path at time step t, q_t is the input question at time t, and $R_{\text{new},t}$ is the updated reasoning path.

3.2 PPO Training Framework

The training of the proposed multi-agent system is primarily guided by the Proximal Policy Optimization (PPO) algorithm, a popular reinforcement learning (RL) [11]technique. PPO is used to fine-tune the policies of the Reasoning Agent, Evaluation Agent, and Regeneration Agent, optimizing their interactions and improving the overall system's reasoning capabilities. PPO has proven to be effective for complex, high-dimensional tasks, such as natural language reasoning, due to its stability and relatively low variance compared to other RL algorithms like TRPO.

At the core of PPO is the idea of optimizing the policy through surrogate objective functions. Let π_θ represent the policy of an agent, where θ denotes the policy parameters. The objective is to maximize the expected reward while preventing large updates to the policy that could destabilize the learning process. The surrogate objective function for PPO can be expressed as:

$$\hat{L}^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]$$

where $r_t(\theta)$ is the probability ratio between the new policy and the old policy at time step t, and $\hat{A}_t$ represents the advantage function, which estimates how much better an action taken at time step t is compared to the average action. The term ϵ is a small constant (typically set to 0.2) that controls the degree of policy clipping, ensuring that the updates do not significantly alter the policy, thus maintaining stability.

The advantage function $\hat{A}_t$ is typically computed using Generalized Advantage Estimation (GAE), which is defined as:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + (\gamma\lambda)^2\delta_{t+2} + \dots$$

where δ_t is the temporal difference (TD) error at time step t, and γ is the discount factor that weighs future rewards. The parameter λ controls the bias-variance trade-off in the advantage estimation process.

To ensure the training process is stable, PPO uses an importance sampling method that prevents excessively large updates to the policy. By clipping the ratio $r_t(\theta)$, PPO ensures that the updates are not too large, thus mitigating the risk of overfitting to the current data and preventing catastrophic forgetting.

3.2.1 Training Process

During training, each agent follows the process outlined below:

1. **Trajectory Collection**: Each agent interacts with the environment (or in this case, the reasoning task) by generating a sequence of actions based on the current policy. This trajectory consists of states, actions, rewards, and the resulting new states. The Reasoning Agent generates an initial reasoning path, which is evaluated by the Evaluation Agent. The Regeneration Agent then refines the reasoning process based on feedback.

2. **Policy Optimization**: Once a batch of trajectories is collected, the agent's policy is updated by optimizing the PPO objective function. The policy update aims to improve the quality of reasoning steps and enhance the accuracy of the final output. This step is performed by computing the gradient of the objective function with respect to the policy parameters θ:

$$\nabla_\theta \hat{L}^{\text{CLIP}}(\theta)$$

3. **Parameter Update**: After calculating the gradient, the policy parameters are updated using a standard stochastic gradient ascent algorithm. The learning rate α controls the size of each update:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta \hat{L}^{\text{CLIP}}(\theta)$$

4. **LoRA Fine-Tuning**: We leverage LoRA (Low-Rank Adaptation) to efficiently fine-tune the Llama-3.1-8B-Instruct model. LoRA introduces trainable low-rank adapters into specific attention modules (*q_proj* and *v_proj*), significantly reducing computational overhead. The LoRA parameters are:

- Rank (r): 8
- LoRA alpha: 32
- LoRA dropout: 0.05

The fine-tuning is conducted on a cluster of four nodes, each with eight NVIDIA H100 GPUs, enabling a total of 32 GPUs. Training is distributed via `torchrun` with the NCCL backend, ensuring efficient communication and gradient synchronization. The effective global batch size is 2048, achieved with a per-device batch size of 16 and gradient accumulation steps of 4. Training spans three epochs with a learning rate of 2×10^{-4} and utilizes FP16 precision for improved speed and memory efficiency.

Through this framework, each agent's policy is optimized iteratively, with feedback loops from the Evaluation and Regeneration Agents refining the reasoning process. The use of PPO ensures that the learning process remains stable and efficient, even in the complex, multi-step tasks required for natural language reasoning.

3.3 Dataset and Preprocessing

In this study, we utilize the HotpotQA dataset, which is designed to evaluate multi-hop reasoning in natural language question answering. The HotpotQA dataset contains questions that require reasoning over multiple documents, making it suitable for testing models that are tasked with handling complex, multi-step reasoning processes. Each question in the dataset is paired with the correct answer, which may require combining information from multiple passages. These passages, referred to as supporting facts, provide the necessary context to answer the questions. The dataset includes two

types of questions: those that require multiple supporting facts and those that require a single piece of information, which we refer to as "distractor" questions. The dataset also provides the exact passages that are considered necessary to answer each question, which are labeled as supporting facts.

For the purpose of this study, we split the HotpotQA dataset into three subsets: training, validation, and test sets. The training set is used to fine-tune the multi-agent reasoning system, the validation set is utilized for model selection and hyperparameter tuning, while the test set is reserved for final evaluation. The data preprocessing steps are essential to ensure that the data is properly formatted and ready for use by the reasoning agents.

The first preprocessing step involves text normalization, where all textual data, including questions, answers, and supporting facts, is transformed into a consistent format. This includes converting all text to lowercase, removing punctuation, and tokenizing the text into smaller units such as words or subwords. Tokenization is performed using a pre-trained tokenizer compatible with modern language models like BERT or GPT, which ensures that the text is processed in a way that aligns with the input requirements of these models.

Once the text is tokenized, the next step is to pair the questions with their corresponding supporting facts. The HotpotQA dataset provides multiple passages for each question, which may span across different documents. We concatenate these passages together into a single input sequence, which is then processed by the Reasoning Agent. The passages are selected based on their relevance to the question, ensuring that only the most pertinent information is provided to the agent. This step is critical in enabling the model to reason across multiple pieces of information and integrate them effectively.

Moreover, we perform feature engineering to extract relevant metadata from the dataset. This includes characteristics such as the length of the questions and passages, the number of entities present, and the complexity of the reasoning task. These features are concatenated with the textual embeddings and fed into the multi-agent reasoning system, providing the agents with additional contextual information that aids in the reasoning process.

Once the preprocessing steps are completed, the data is structured into a format suitable for training. Each training example consists of a tokenized question paired with its corresponding supporting facts, along with the correct answer. These components are stored as tensors, which are then organized into batches for efficient training. The training process involves feeding these preprocessed inputs into the multi-agent reasoning system, which learns to generate and refine answers through an iterative process of reasoning, evaluation, and regeneration.

The preprocessing pipeline ensures that the model is exposed to high-quality, relevant data, while also incorporating additional context such as entity information and metadata features. This comprehensive approach to data preprocessing is critical for enabling the multi-agent system to handle complex multi-hop reasoning tasks and generate accurate answers.

For fine-tuning, the O1-OPEN/OpenO1-SFT dataset is a structured collection of instruction-response pairs aimed at improving the alignment and reasoning capabilities of large language models. Each entry contains two main fields: `instruction` and `output`. The `instruction` provides a query or task description, while the `output` contains a detailed, step-by-step solution or response.

3.4 Experimental Setup

The experimental setup for this research involves several key components: the training of the multi-agent system, evaluation of reasoning quality, and analysis of model efficiency. The system is trained using the HotpotQA dataset, with specific attention paid to the interaction between the three agents and the effectiveness of the PPO algorithm in improving reasoning performance. The experiments are designed to test the system's ability to generate accurate and coherent reasoning steps, evaluate the correctness of these steps, and iteratively refine the answers based on feedback.

To evaluate the system's performance, we will assess the accuracy of the generated answers, the logical consistency of the reasoning steps, and the overall efficiency of the model in terms of training time and resource usage. The impact of using LoRA adapters on training efficiency and model convergence will also be analyzed. Additional experiments will be conducted to explore the system's scalability and its ability to generalize to other types of reasoning tasks.

Through these experiments, we aim to demonstrate the effectiveness of the proposed multi-agent reasoning framework and its ability to outperform traditional LLMs in complex, multi-step reasoning tasks.

4 Results and Analysis

In this section, we present the results of our multi-agent reasoning system and analyze the system's performance on the HotpotQA dataset. The results include an evaluation of the model's reasoning accuracy, its ability to handle multi-hop reasoning tasks, and a comparison with baseline models. We also provide a detailed analysis of the strengths and weaknesses of the proposed system, drawing insights into areas where the model excels and where further improvements can be made.

4.1 Performance on HotpotQA

The performance of the multi-agent reasoning system is evaluated on the HotpotQA dataset, which contains a diverse set of multi-hop question answering tasks. We use several metrics to assess the quality of the generated answers, including accuracy, F1 score, and exact match (EM) score. The accuracy metric measures the proportion of correct answers out of the total number of questions, while the F1 score provides a balance between precision and recall, and the EM score indicates the percentage of questions for which the model generates exactly the same answer as the ground truth.

Table 1 shows the performance of our model compared to several baseline models, including a standard BERT-based question answering model, a vanilla Transformer model, and a traditional multi-hop reasoning model. As seen in the table, our multi-agent system significantly outperforms the baseline models in terms of both accuracy and F1 score. The exact match score also demonstrates a notable improvement, indicating that the multi-agent system is able to generate answers that are highly consistent with the ground truth.

Table 1: Performance Comparison on HotpotQA

Model	Accuracy	F1 Score	Exact Match (EM)
BERT-based QA Model	71.2%	68.5%	56.1%
Vanilla Transformer	73.8%	71.4%	58.7%
Traditional Multi-hop Reasoning	75.1%	72.5%	60.2%
Multi-agent Reasoning System	**78.3%**	**75.6%**	**63.4%**

The results clearly demonstrate the effectiveness of the multi-agent architecture. By introducing a collaborative reasoning process where the Reasoning Agent generates hypotheses, the Evaluation Agent assesses their correctness, and the Regeneration Agent refines the output, the model is able to produce more accurate and contextually coherent answers. This collaborative approach enhances the model's ability to perform multi-hop reasoning, a task that requires combining information from multiple sources in a manner that traditional single-agent systems struggle with.

4.2 Error Analysis

Despite the overall improvements, some errors are still observed in the generated answers. A common source of error is the incorrect linking of entities between the question and the supporting facts. This issue often arises when the model is unable to correctly identify and associate relevant pieces of information, especially in cases where multiple entities with similar names or relationships are present. Additionally, some reasoning steps involve complex logic that the model fails to capture accurately, particularly when dealing with highly ambiguous questions or rare entities that do not appear frequently in the training data.

To better understand the types of errors the model makes, we perform a qualitative error analysis. We categorize the errors into two main types: 1) Entity Linking Errors, where the model incorrectly associates entities between the question and supporting facts, and 2) Reasoning Errors, where the model fails to follow the correct reasoning path or misinterprets the relationship between different facts. We find that entity linking errors are most prevalent in cases where the supporting facts contain complex or less frequent entities, while reasoning errors tend to occur in situations that require abstract or multi-step reasoning.

To address these errors, we propose two potential directions for future work. First, incorporating a more advanced entity linking mechanism could improve the model's ability to correctly identify and associate relevant entities, particularly in challenging scenarios involving rare or ambiguous entities. Second, enhancing the reasoning process by incorporating external knowledge sources or memory augmentation could help the model capture more complex reasoning patterns and reduce errors in multi-step logical inference.

4.3 Comparison with Existing Methods

In addition to comparing our model with baseline models, we also compare the performance of our multi-agent system with other state-of-the-art multi-hop reasoning models. These models include methods that leverage graph-based reasoning, such as Graph Neural Networks (GNNs), as well as other reinforcement learning-based systems. While some of these models achieve competitive performance on the HotpotQA dataset, our multi-agent system shows superior results in both accuracy and F1 score, especially in the context of handling multi-hop reasoning tasks.

A key differentiator of our approach is the use of multiple agents working in tandem, which allows the model to generate more accurate answers through iterative refinement. This stands in contrast to models that rely on a single agent to perform both reasoning and evaluation, which often results in less accurate or incoherent answers. The ability of our system to iteratively improve upon its reasoning process through collaboration between agents gives it a distinct advantage over other methods that do not employ such a collaborative framework.

4.4 Discussion

The results of our experiments demonstrate the effectiveness of the multi-agent reasoning framework in improving the performance of multi-hop question answering tasks. The collaborative approach, where each agent plays a specific role in generating, evaluating, and refining answers, has proven to be a powerful mechanism for enhancing reasoning accuracy. Our model not only outperforms baseline models but also shows significant improvements over existing multi-hop reasoning methods, especially in complex reasoning scenarios.

Despite these successes, there are still challenges to address. The error analysis reveals that the model struggles with certain types of reasoning tasks, particularly those involving ambiguous entities or complex logic. Future work should focus on addressing these weaknesses by incorporating better entity recognition and external knowledge sources to support reasoning. Additionally, improving the training process through more advanced techniques, such as curriculum learning or few-shot learning, could help the model generalize better to unseen questions and entities.

Overall, the results of this study highlight the potential of multi-agent systems in improving large language model reasoning, particularly in the context of complex, multi-hop question answering tasks. The insights gained from this research provide a solid foundation for further advancements in multi-agent reinforcement learning for natural language processing.

5 Conclusion and Future Work

5.1 Summary

This research presents a novel multi-agent collaboration framework aimed at enhancing the reasoning capabilities of large language models. The primary contributions of this work include the development of a multi-agent system composed of the *Reasoning Agent*, the *Evaluation Agent*, and the *Regeneration Agent*, each playing a crucial role in generating, evaluating, and refining reasoning processes. By adopting the Proximal Policy Optimization (PPO) algorithm, the system effectively updates and optimizes the agent parameters, ensuring stability and performance during training. The use of Low-Rank Adapters (LoRA) further enhances the training efficiency by enabling lightweight updates to the model parameters, thus minimizing computational overhead.

The proposed framework significantly improves the reasoning quality and efficiency of the model. By implementing an iterative reasoning process, the system can iteratively refine its outputs, resulting in more accurate and coherent responses. This collaborative approach, combined with the robustness of PPO optimization, addresses several limitations of traditional large language models, particularly

in handling multi-step reasoning and maintaining logical coherence throughout the process. Furthermore, the system's architecture allows for seamless scalability, providing a foundation for future improvements and the integration of additional agents or functionalities.

5.2 Future Work

While the current system demonstrates promising results, there are several potential avenues for future development. One possible direction is to incorporate more diverse evaluation metrics to assess reasoning quality from different perspectives, such as fairness, transparency, or reliability. Expanding the range of evaluation criteria would provide a more comprehensive understanding of the reasoning process and could lead to further refinement of the output generation process.

Another area for improvement is the optimization of the collaboration mechanism between agents. While the current agent structure is effective, exploring new methods to improve agent communication, synchronization, and conflict resolution could enhance the overall system's efficiency and robustness. Additionally, improvements to the reward computation process—by including more complex or domain-specific feedback—could further guide the agents in producing more aligned and contextually appropriate responses.

Looking forward, there is significant potential to extend the system's capabilities for real-world applications. In education, this framework could be applied to enhance tutoring systems, enabling adaptive learning experiences based on dynamic reasoning. In research, the system could aid in generating more accurate and well-supported hypotheses by integrating external knowledge sources or databases. Moreover, in industrial applications, the ability to perform multi-step reasoning in fields like customer service, automated decision-making, or healthcare could greatly benefit from such a system.

Furthermore, there are numerous opportunities for functional expansion, such as integrating multi-language support, where the system could perform reasoning tasks in various languages, making it applicable to a global user base. The incorporation of knowledge graphs and other structured knowledge representations could further strengthen the system's ability to perform reasoning tasks that require external factual knowledge. Overall, this research opens the door to a wide range of future applications and enhancements, paving the way for more intelligent and adaptive systems across various domains.

References

[1] Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*, 2023.

[2] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[3] Dom Huh and Prasant Mohapatra. Multi-agent reinforcement learning: A comprehensive survey, 2024.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[6] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, November 2017.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[8] Pouya Pezeshkpour, Eser Kandogan, Nikita Bhutani, Sajjadur Rahman, Tom Mitchell, and Estevam Hruschka. Reasoning capacity in multi-agent systems: Limitations, challenges and human-centered solutions, 2024.

[9] Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision, 2017.

[10] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[11] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks, 2015.