Player–Coach Teamwork: Multi-agent Collaboration for Improving LLM Reasoning

Heewon Park
Soongsil University
Seoul, South Korea
heewon012@soongsil.ac.kr

Minhae Kwon *
Soongsil University
Seoul, South Korea
minhae@ssu.ac.kr

Abstract

Large language models (LLMs) are increasingly deployed as autonomous agents in interactive environments. These settings require multi-step reasoning, adaptation to context, and long-horizon decision making. Methods that incorporate reasoning traces, along with reflection-based approaches, have improved robustness. However, single-agent designs still reinforce internal biases and often correct only after errors accumulate, limiting their effectiveness in real-time interactions. Multi-agent frameworks demonstrate the potential of role separation and collaboration, but typically require fixed protocols or additional training overhead. We propose a *Player–Coach* collaboration system that achieves conditional, real-time correction without additional training. The Player engages directly with the environment through a reasoning–acting loop, while the Coach is selectively invoked under high uncertainty to provide metacognitive feedback such as clarifying objectives, surfacing overlooked observations, or resolving confusions. A composite uncertainty trigger, combining normalized entropy and margin signals, ensures interventions occur only when needed. Experiments on the ALFWorld benchmark show that this architecture improves success rates and reduces trajectory lengths compared to single-agent baselines, highlighting the promise of uncertainty-aware multi-agent design for scaling LLMs in interactive environments.

1 Introduction

The rise of LLMs has accelerated their use as autonomous agents in interactive environments, where tasks naturally unfold over multiple steps. In such settings, agents must reason about ambiguous observations, adapt to evolving contexts, and avoid cascading errors that can accumulate across long sequences of actions [1, 2, 3]. These requirements highlight the importance of designing systems that remain robust, reliable, and efficient when deployed in dynamic and long-horizon environments.

Recent studies have shown that augmenting LLMs with explicit reasoning traces and reflective mechanisms can significantly improve their performance in sequential decision making [4, 5, 6, 7, 8, 9, 10, 11]. Such methods leverage structured reasoning, action-feedback loops, and post-hoc reflection to enhance task completion. However, most existing approaches rely on a single-agent paradigm, where the model must evaluate and correct its own behavior. This design often reinforces internal biases and provides corrections only after errors have already accumulated, limiting its effectiveness in real-time multi-step interactions.

Multi-agent collaboration has emerged as a promising approach to externalize reasoning and introduce role specialization [12, 13, 14, 15]. These frameworks demonstrate that interaction between agents,

*Corresponding author: Minhae Kwon (minhae@ssu.ac.kr)

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Scaling Environments for Agents (SEA).

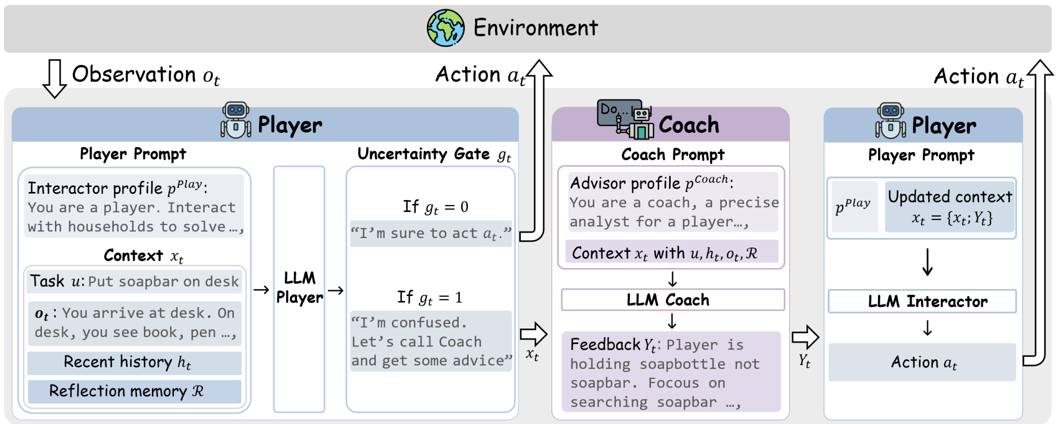

Figure 1: Overview of the proposed Player–Coach system. At each step t, the environment provides an observation o_t, which the Player encodes into a contextual prompt x_t including the task u, recent history h_t, and reflection memory $\mathcal{R}$. The Player proposes an action a_t with reasoning as ReAct, while an uncertainty gate g_t monitors confidence. If the Player is confident ($g_t = 0$), the action is executed directly. If uncertainty is high ($g_t = 1$), the Player queries the Coach, which analyzes x_t and produces feedback Y_t containing rationale, reminders, or corrective suggestions. This feedback is appended to the Player's context, prompting re-scoring of admissible actions before executing the final action. By separating fast interaction from conditional metacognitive intervention, the framework enables real-time error correction and yields more reliable trajectories.

whether through role assignment, debate, or actor–critic structures, can improve robustness and generalization. However, many of these methods still depend on fixed collaboration protocols or additional training overhead, which makes them less practical for lightweight deployment in interactive environments.

To address these limitations, we propose a Player–Coach multi-agent collaboration system that enables conditional, real-time correction in multi-turn environments. Figure 1 shows our proposed Player-Coach multi-agent system. The Player engages with the environment through a reasoning and acting loop as in ReAct [5], while the Coach is selectively invoked under uncertainty to analyze the Player's trajectory from an external vantage point. The Coach provides structured metacognitive feedback by clarifying objectives, surfacing forgotten observations, and resolving confusions. This feedback is immediately incorporated into the Player's decision process, ensuring that fast interaction and deliberate correction remain explicitly separated. As a result, the architecture mitigates self-reinforcing loops and yields more accurate, stable, and efficient multi-step behavior.

Our contributions are summarized as follows.

- We propose a Player–Coach multi-agent collaboration system that enables real-time, uncertainty-triggered self-correction in multi-step environments.
- We introduce a composite uncertainty measure that combines normalized entropy and top-two margin to trigger interventions precisely when needed.
- Through experiments on ALFWorld, a challenging text-based household benchmark, we show that our approach improves task success rates and reduces trajectory length compared to strong single-agent baselines, all without requiring additional training or fine-tuning.

The remainder of this paper is organized as follows. Section 2 reviews related works. Section 3 presents the proposed method. Section 4 reports simulation results. Section 5 concludes the paper with a discussion of key findings.

2 Related Works

LLM-Based Sequential Reasoning and Acting LLMs have been increasingly applied as agents that perform sequential reasoning and acting in interactive environments [4, 5, 6, 7, 9, 10, 16, 17, 18, 19, 20, 21]. Early work, such as Chain-of-Thought [4] shows that prompting with explicit intermediate reasoning steps significantly improves problem solving. Tree-of-Thought [6] extends this idea by enabling structured search over reasoning paths, while ReAct [5] integrates reasoning and acting in a loop to guide decision making. Beyond prompting strategies, LAC [7] introduces an Actor–Critic style framework for sequential decision making without gradient-based optimization. These approaches highlight the promise of augmenting LLMs with explicit reasoning traces and interaction loops, but they typically rely on a single-agent paradigm. This limitation becomes more evident as tasks grow longer and more open-ended, where accumulated biases and missing context can lead to cascading errors that single-agent reasoning alone cannot resolve.

Reflection and Collaborative Correction A substantial body of work enhances LLM robustness through either self-reflection or multi-agent collaboration [8, 10, 11, 9, 12, 13, 15, 14]. Reflection-based methods focus on enabling agents to revise their own trajectories: Reflexion [8] introduces verbal reinforcement learning to improve future behavior, Constitutional AI [10] leverages normative principles for self-feedback, and Self-Refine [11] and RAFA [9] iteratively refine outputs using self-generated or adaptive signals. While these methods improve robustness, they generally operate post hoc, intervening only after errors have already occurred. Multi-agent frameworks externalize correction by introducing role separation and inter-agent communication. AutoGen [12] and CAMEL [13] employ structured role-based workflows, MAD [15] fosters divergent thinking through debate, and MAPLE [14] adopts an actor–critic formulation to coordinate multiple agents. These studies highlight the promise of reflective or collaborative correction, yet many rely on fixed protocols, extensive training, or additional supervision. Our Player–Coach multi-agent framework advances this direction by providing real-time, uncertainty-triggered metacognitive correction without extra training overhead, showing that even lightweight role separation can yield significant benefits in multi-step environments.

3 Method

We propose a *Player–Coach* multi-agent collaboration LLM system that performs conditional, real-time intervention during sequential decision making. The system comprises two distinct agents, each instantiated by a role-specific prompt (profile): a *Player* that directly engages with the environment by alternating reasoning and acting, and a *Coach* that is invoked only under high uncertainty to supply metacognitive feedback. By separating interaction from externalized analysis, the Coach can scrutinize the Player's trajectory from an independent vantage point, mitigating self-reinforcing errors and producing trajectories that are both more accurate and more efficient. The overall algorithmic flow of the proposed system is provided in Algorithm 1.

3.1 Player for Reasoning and Acting

The Player is the agent that directly engages with the environment. At each step t, the Player alternates between a short natural-language *reason* and a concrete *act* as ReAct [5]. Given the task instruction and a bounded history of recent steps, the Player constructs a contextual prompt and scores the admissible actions to produce a fast proposal. The context x_t is serialized into a textual prompt following a fixed template, where the task instruction precedes the history and reflection memory, ensuring consistent conditioning of the LLM.

Formally, the environment $\mathcal{E}$ provides the current observation o_t and the set of admissible actions $\mathcal{A}_t = \{a^{(1)}, \ldots, a^{(K_t)}\}$, where $K_t = |\mathcal{A}_t|$ is the number of admissible actions at step t. The Player maintains the most recent n steps of history

$$h_t = \big((o_{t-n}, a_{t-n}), \ldots, (o_{t-1}, a_{t-1})\big),$$

and builds a textual context

$$x_t = \mathrm{Concat}(u, h_t, o_t, \mathcal{R}),$$

where u is the task instruction and $\mathcal{R}$ is a short memory of prior reflections.

Each admissible action string $a = (w_1, \ldots, w_{|a|})$ is assigned a sequence log-likelihood under the Player profile p^{Play} as follows.

$$\ell(a \mid x_t) = \sum_{j=1}^{|a|} \log f_\theta(p^{Play}, x_t, w_{<j})[w_j] \tag{1}$$

Algorithm 1 Player–Coach LLM Agent System

Input: Task instruction u, environment $\mathcal{E}$, Player profile p^{Play}, Coach profile p^{Coach}
Output: Action sequence $\{a_1, a_2, \ldots\}$
for each step $t = 1, 2, \ldots$ until termination **do**
  Receive observation o_t and admissible actions $\mathcal{A}_t$ from $\mathcal{E}$
  Build context $x_t = \text{Concat}(u, h_t, o_t, \mathcal{R})$
  Compute log-likelihood scores $\ell(a \mid x_t)$ for $a \in \mathcal{A}_t$ under p^{Play}
  Normalize into preference distribution $q(a \mid x_t)$
  Evaluate uncertainty signals: entropy $\bar{H}_t$ and margin m_t
  if $\bar{H}_t \geq \tau_H$ **or** $m_t \leq \tau_m$ **then**
    Query Coach: $Y_t = f_\theta(p^{Coach}, x_t)$
    Update context: $x_t \leftarrow \text{Concat}(x_t, Y_t)$
  end if
  Select action $a_t = \arg\max_{a \in \mathcal{A}_t} q(a \mid x_t)$
  Execute a_t in $\mathcal{E}$ and update history h_{t+1}
end for
Receive final trial-level reflection R^{trial} from Coach
Update reflection memory $\mathcal{R} \leftarrow \mathcal{R} \cup \{R^{\text{trial}}\}$

Here, f_θ denotes the base LLM and $[\cdot]$ indexes the probability of token w_j. The likelihood $\ell(a \mid x_t)$ captures both linguistic plausibility and contextual appropriateness, meaning that actions more consistent with the task and history naturally receive higher scores. These scores are normalized into a preference distribution as follows.

$$q(a \mid x_t) = \frac{\exp(\ell(a \mid x_t))}{\sum_{a' \in \mathcal{A}_t} \exp(\ell(a' \mid x_t))}, \qquad a \in \mathcal{A}_t. \tag{2}$$

The Player then outputs a fast proposal via greedy selection $a_t \in \arg\max_{a \in \mathcal{A}_t} q(a \mid x_t)$, which provides a deterministic choice that balances efficiency and stability without requiring more costly beam search or sampling strategies.

3.2 Uncertainty-Guided Intervention Trigger

The Player invokes the Coach only when its own action preference distribution $q(a \mid x_t)$ in (2) indicates high uncertainty. Specifically, we monitor two complementary signals: (i) normalized entropy, which captures diffuse uncertainty spread over many actions, and (ii) the top-two margin, which captures local ambiguity between the most likely candidates. By combining these signals, the system reduces both spurious interventions and missed opportunities.

First, the normalized entropy is defined as follows.

$$\bar{H}_t = -\frac{1}{\log K_t} \sum_{a \in \mathcal{A}_t} q(a \mid x_t)\, \log q(a \mid x_t) \tag{3}$$

Here, high $\bar{H}_t$ indicates uncertainty dispersed across many admissible actions, making it difficult for the Player to commit to a single option.

Second, we define the top-two margin

$$m_t = q_{(1)} - q_{(2)}, \tag{4}$$

where $q_{(1)} \geq q_{(2)}$ are the two largest probabilities. A small m_t signals indecision between the most likely actions, even when overall entropy may not be high.

The uncertainty gate g_t is activated whenever either condition holds

$$g_t = \mathbb{I}\big[\, \bar{H}_t \geq \tau_H \;\vee\; m_t \leq \tau_m \,\big], \tag{5}$$

where τ_H and τ_m are hyperparameters chosen in $(0, 1)$. By combining entropy and margin, the gate avoids pathological cases where one metric alone would fail (e.g., low entropy but high local ambiguity). This composite rule balances coverage and precision, ensuring the Coach intervenes only when the Player is genuinely uncertain.

3.3 Coach as Externalized Metacognition and Fusion

The Coach complements the Player by providing targeted metacognitive feedback from an externalized perspective. Unlike the Player, which focuses on rapid reasoning and acting, the Coach is invoked only when the uncertainty gate g_t in (5) is triggered. This allows the Coach to critically examine the Player's context and trajectory, surfacing forgotten facts, resolving object confusions, or re-stating the task objective. The feedback Y_t is generated in a structured format, typically including (i) a rationale for the suggested correction, (ii) reminders of overlooked facts, and (iii) a concrete corrective suggestion or minimal probe.

Formally, when $g_t = 1$, the Player queries the Coach feedback $Y - t$ with the current context under Coach profile p^{Coach} as follows.

$$Y_t \;=\; f_\theta\big(p^{Coach}, x_t\big) \tag{6}$$

The final action at step t is chosen as

$$a_t \;=\; \begin{cases} \arg\max\limits_{a \in \mathcal{A}_t} q(a \mid x_t) & g_t = 0 \\ \arg\max\limits_{a \in \mathcal{A}_t} q(a \mid \mathrm{Concat}(x_t, Y_t)) & g_t = 1, \end{cases} \tag{7}$$

ensuring that Coach feedback directly influences the Player's decision while remaining within the admissible action set $\mathcal{A}_t$. Instead of overriding the Player's choice, the Coach augments the context, allowing the Player to rescore all admissible actions with enriched information.

At the end of each trajectory, the Coach also provides a trial-level reflection R^{trial}, which summarizes lessons or common mistakes (e.g., forgotten observations, object confusions). This reflection is appended to the memory as $\mathcal{R} \leftarrow \mathcal{R} \cup \{R^{\text{trial}}\}$, so that subsequent trials can benefit from accumulated experience, effectively integrating episodic learning into the Player's ongoing decision process.

4 Simulation Results

In this section, we present empirical evaluations of the proposed Player–Coach multi-agent LLM system. We assess its effectiveness on long-horizon, multi-step decision making tasks and compare against strong single-agent baselines. Example prompts are provided in Appendix A.

4.1 Experimental Setup

We evaluate our approach on ALFWorld benchmark [22], a widely used text-based household environment that requires agents to complete everyday tasks through sequences of high-level actions. The benchmark contains 140 seen and 134 unseen evaluation tasks. The main challenge of ALFWorld lies in locating target objects and executing household tasks that demand commonsense reasoning over long action sequences.

As baselines, we include ReAct [5], which alternates reasoning and acting under a single-agent design, and LAC [7], an actor–critic style single-agent framework. Both represent strong single-agent paradigms. Our method is implemented with the LLaMA-3-8B model [23] as the pretrained backbone, and trajectories are capped at a maximum of 50 steps.

We also compare two variants of our Player–Coach system: (i) Fixed-Step Coach, where the Coach is invoked deterministically every 20 steps, and (ii) Uncertainty-Gated Coach, where interventions are adaptively triggered by the proposed uncertainty gate g_t in (5). This ablation highlights the effectiveness of uncertainty-aware conditional intervention.

Table 1: Performance comparison on ALFWorld with LLaMA-3-8B. We report success rate (%) and average trajectory length for successful episodes (mean $\pm$ std). Best results are **bolded**.

	Seen		Unseen	
Method	Success rate (%)	Trajectory length	Success rate (%)	Trajectory length
ReAct [5]	64.9	13.4 ± 6.5	59.0	15.1 ± 7.2
LAC [7]	67.1	14.2 ± 6.7	61.1	15.3 ± 6.9
Ours (Fixed-Step)	72.4	12.7 ± 5.2	62.4	13.3 ± 6.8
Ours (Uncertainty-Gated)	**77.6**	$\mathbf{11.3 \pm 4.1}$	**66.9**	$\mathbf{12.5 \pm 5.8}$

4.2 Performance Comparison

Table 1 shows success rates and average trajectory lengths on the ALFWorld benchmark. Both variants of our Player–Coach framework consistently outperform the single-agent baselines, demonstrating the advantage of externalized metacognitive intervention.

Player with fixed-step Coach yields noticeable gains over ReAct and LAC, with higher success rates and shorter trajectories. This suggests that even simple periodic feedback helps mitigate self-reinforcing loops by forcing the Player to reconsider its state and plan. Nevertheless, this variant also shows inefficiencies: it consumes interventions even when the Player is confident and accurate, and it may fail to provide corrections at the most critical moments when errors first arise. These limitations explain why its improvements, though consistent, are modest relative to the uncertainty-based variant.

Player with uncertainty-gated Coach achieves the best overall performance, surpassing all baselines by a large margin. In the seen split, it improves success rate by over 10% compared to ReAct, while also reducing trajectory length by more than two steps on average. In the unseen split, it yields a 5.8% gain in success rate alongside a 2.6-step reduction, highlighting that the uncertainty-driven trigger generalizes effectively to new tasks. We attribute these improvements to two factors: (i) adaptive interventions prevent wasted calls to the Coach, ensuring that feedback is provided only when the Player shows genuine hesitation or confusion, and (ii) feedback arrives exactly at decision points with the greatest potential for cascading errors, preventing early mistakes from propagating through the trajectory.

Taken together, these results emphasize that metacognitive oversight is most effective when it is conditional and demand-driven, rather than periodic or fixed. By aligning interventions with the Player's uncertainty, our approach balances efficiency with robustness, leading to both higher success and shorter, more stable trajectories. This demonstrates the potential of role-separated, uncertainty-aware agents for scaling to complex multi-turn environments.

5 Conclusion and Broader Impact

We presented a Player–Coach multi-agent framework that enables conditional, real-time self-correction for LLM agents operating in multi-step environments. The Player executes reasoning and acting steps efficiently, while the Coach is invoked only under high uncertainty to provide metacognitive feedback. By combining normalized entropy and margin-based signals, our uncertainty gate triggers interventions precisely when they are most needed. Experiments on ALFWorld show that this design improves success rates and reduces trajectory lengths compared to strong single-agent baselines, without requiring any additional training.

Our framework illustrates how lightweight, training-free intervention mechanisms can improve the reliability and efficiency of autonomous agents. This has implications for safer deployment of LLM agents in interactive applications, where minimizing cascading errors is critical. At the same time, the approach promotes modularity, enabling extensions to settings with multiple specialized Coaches or human-in-the-loop systems. While promising, our method has several limitations. The current design relies on handcrafted uncertainty thresholds that may require environment-specific tuning. Moreover, our evaluation is restricted to text-based household tasks in ALFWorld; extensions to embodied, multimodal, or real-world environments may expose new challenges such as perception noise or delayed feedback. Future work will explore adaptive thresholding, richer feedback modalities, and validation in more diverse domains.

References

[1] Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, et al. Mt-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. In *Association for Computational Linguistics*, 2024.

[2] Ma Chang, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An analytical evaluation board of multi-turn LLM agents. In *Advances in Neural Information Processing Systems*, volume 37, pages 74325–74362, 2024.

[3] Kaustubh Deshpande, Ved Sirdeshmukh, Johannes Baptist Mols, Lifeng Jin, Ed-Yeremai Hernandez-Cardona, Dean Lee, Jeremy Kritz, Willow E Primack, Summer Yue, and Chen Xing. Multichallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier LLMs. In *Findings of the Association for Computational Linguistics*, pages 18632–18702, 2025.

[4] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in neural information processing systems*, volume 35, pages 24824–24837, 2022.

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

[6] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822, 2023.

[7] Heng Dong, Kefei Duan, and Chongjie Zhang. Enhancing decision-making of large language models via actor-critic. In *International Conference on Machine Learning*, 2025.

[8] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652, 2023.

[9] Zhihan Liu, Hao Hu, Shenao Zhang, Hongyi Guo, Shuqi Ke, Boyi Liu, and Zhaoran Wang. Reason for future, act for now: A principled framework for autonomous llm agents with provable sample efficiency. In *International Conference on Machine Learning*, 2023.

[10] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[11] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594, 2023.

[12] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation framework. In *Conference on Language Modeling*, 2024.

[13] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for “mind" exploration of large language model society. In *Advances in Neural Information Processing Systems*, volume 36, pages 51991–52008, 2023.

[14] Ye Bai, Minghan Wang, and Thuy-Trang Vu. MAPLE: Multi-agent adaptive planning with long-term memory for table reasoning. *arXiv preprint arXiv:2506.05813*, 2025.

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

[16] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213, 2022.

[17] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *Advances in neural information processing systems*, 2023.

[18] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. VOYAGER: An open-ended embodied agent with large language models. *arXiv preprint arXiv: Arxiv-2305.16291*, 2023.

[19] Ruiwen Zhou, Yingxuan Yang, Muning Wen, Ying Wen, Wenhao Wang, Chunling Xi, Guoqiang Xu, Yong Yu, and Weinan Zhang. TRAD: Enhancing llm agents with step-wise thought retrieval and aligned decision. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024.

[20] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, volume 36, pages 28091–28114, 2023.

[21] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 17682–17690, 2024.

[22] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

[23] AI Meta. Llama 3: Open foundation and instruction models, 2024.

A Prompts

Table 1: An example of trajectory for ALFWorld `pick_two_obj_and_place`.

```
You are in the middle of a room. Looking quickly around you, you see
an armchair 1, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a
drawer 5, a drawer 4, a drawer 3, a drawer 2, a drawer 1, a dresser 1,
a garbagecan 1, a safe 1, a shelf 12, a shelf 11, a shelf 10, a shelf
9, a shelf 8, a shelf 7, a shelf 6, a shelf 5, a shelf 4, a shelf 3,
a shelf 2, a shelf 1, a sidetable 1, and a sofa 1. Your task is to:
put two pillow in sofa.
> [Player] Act 1: think: To solve the task, I need to find and take
the first pillow, then put it in sofa, then find and take the second
pillow, then put it in sofa.
OK.
> [Player] Act 2: think: First I need to find the first pillow. A
pillow is more likely to appear in armchair (1), cabinet (1-4),
drawer (1-5), dresser (1), sidetable (1), sofa (1). I can check one
by one, starting with armchair 1.
OK.
> [Player] Act3: go to armchair 1
You arrive at armchair 1. On the armchair 1, you see a cellphone 2,
a pillow 2, and a pillow 1.
> [Player] Act 4: take pillow 1 from armchair 1
You pick up the pillow 1 from the armchair 1.
> [Player] Act 5: think: Now I find the first pillow (1). Next, I
need to put it in sofa 1.
OK.
> [Player] Act 6: go to sofa 1
You arrive at sofa 1. On the sofa 1, you see a newspaper 3.
```

> [Uncertainty] H=2.054, margin=0.060, $g_t = 1 \rightarrow$ Coach invoked.
> [Coach] Rationale: You are close, but you forgot to put the first
pillow in sofa. You should put the first pillow in sofa before find-
ing the second pillow.
> [Coach] Plan: ['put pillow 1 in sofa 1', 'go to armchair 2', 'take
second pillow', 'put second pillow in sofa 1']
> [Coach] Advice: Put the first pillow in sofa before finding the
second one.

```
> [Player] Act 7: move pillow 1 to sofa 1
On the sofa 1, you see a newspaper 3, and a pillow 1.
> [Player] Act 8: go to armchair 1
You arrive at armchair 1. On the armchair 1, you see a cellphone 2,
and a pillow 2.
> [Player] Act 9: take pillow 2 from armchair 1
You pick up the pillow 2 from the armchair 1.
> [Player] Act 10: think: Now I find the second pillow (2). Next, I
need to put it in sofa 1.
OK.
> [Player] Act 11: go to sofa 1
You arrive at sofa 1. On the sofa 1, you see a newspaper 3, and a
pillow 1.
> [Player] Act 12: move pillow 2 to sofa 1
You move the pillow 2 to the sofa 1.
> [Coach] Trial reflection: Consider the possibility of multiple ob-
jects in a location before taking an action.
Outcome: success
```

Table 2: Example trajectory on ALFWorld `pick_clean_then_place_in_recep`.

```
You are in the middle of a room. Looking quickly around you, you see
a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a countertop 1, a
garbagecan 1, a handtowelholder 2, a handtowelholder 1, a sinkbasin 2,
a sinkbasin 1, a toilet 1, a toiletpaperhanger 1, and a towelholder 1.
Your task is to: put a clean cloth in countertop.
> [Player] Act 1: think: To solve the task, I need to find and take
a cloth, then clean it with sinkbasin, then put it in countertop.
OK.
> [Player] Act 2: think: First I need to find a cloth. I will check
the cabinets starting from cabinet 4.
OK.
> [Player] Act 3: go to cabinet 4
You arrive at cabinet 4. The cabinet 4 is closed.
> [Player] Act 4: open cabinet 4
You open the cabinet 4. The cabinet 4 is open. In it, you see a
soapbottle 2.
> [Player] Act 5: go to cabinet 3
You arrive at cabinet 3. The cabinet 3 is closed.
> [Player] Act 6: open cabinet 3
You open the cabinet 3. The cabinet 3 is open. In it, you see a
cloth 2.
> [Player] Act 7: take cloth 2 from cabinet 3
You pick up the cloth 2 from the cabinet 3.
> [Player] Act 8: go to sinkbasin 1
You arrive at sinkbasin 1. On the sinkbasin 1, you see a soapbar 1.
> [Player] Act 9: clean cloth 2 with sinkbasin 1
You clean the cloth 2 using the sinkbasin 1.
> [Player] Act 10: go to countertop 1
You arrive at countertop 1. On the countertop 1, you see a candle 1,
a cloth 3, and a spraybottle 1.
> [Player] Act 11: think: I might still need soap from cabinet 4 to
clean a cloth before placing it on the countertop. I should go back
and fetch soap first.
OK.
> [Uncertainty] H=1.87, margin=0.78, g_t = 1 → Coach invoked.
> [Coach] Rationale: You already cleaned cloth 2. The goal is to
place a clean cloth on the countertop, not to fetch soap now. Also
note there is an unclean cloth 3 on the countertop--do not switch tar-
gets.
> [Coach] Plan: (1) Verify the scene, (2) Place cloth 2 on countertop
1.
> [Coach] Advice: Skip fetching soap; execute the placement with the
cleaned cloth you hold.
> [Player] Act 12: examine countertop 1
On the countertop 1, you see a candle 1, a cloth 3, and a spraybottle
1.
> [Player] Act 13: move cloth 2 to countertop 1
You move the cloth 2 to the countertop 1.
> [Coach] Trial reflection: After cleaning an item, prioritize plac-
ing it to meet the goal rather than restarting the cleaning process.
Outcome: success
```

Table 3: Example trajectory on ALFWorld `pick_heat_then_place_in_recep`.

```
You are in the middle of a room.  Looking quickly around you, you see
a cabinet 6, a cabinet 5, a cabinet 4, a cabinet 3, a cabinet 2, a
cabinet 1, a coffeemachine 1, a countertop 3, a countertop 2, a
countertop 1, a drawer 3, a drawer 2, a drawer 1, a fridge 1, a
garbagecan 1, a microwave 1, a shelf 3, a shelf 2, a shelf 1, a
sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a
stoveburner 1, and a toaster 1.
Your task is to:  put a hot apple in fridge.
> [Player] Act 1:  think:  First I need to find an apple.  I will
start with the fridge.
OK.
> [Player] Act 2:  go to fridge 1
You arrive at fridge 1.  The fridge 1 is closed.
> [Player] Act 3:  open fridge 1
You open the fridge 1.  In it, you see a bowl 1, a cup 2, a cup 1, a
lettuce 1, a mug 2, a potato 1, and a tomato 1.
> [Player] Act 4:  think:  No apple here.  Maybe I should check
cabinets one by one.
OK.
> [Uncertainty] H=2.05, margin=0.09, g_t = 1 → Coach invoked.
> [Coach] Rationale:  Apples are more likely on countertops.
> [Coach] Plan:  (1) Go to countertop 1, (2) take apple, (3) heat it
with microwave, (4) place in fridge.
> [Coach] Advice:  Check the countertop before searching every cabi-
net.
> [Player] Act 5:  go to countertop 1
You arrive at countertop 1.  On the countertop 1, you see an apple 1,
an apple 2, and a mug 3.
> [Player] Act 6:  take apple 1 from countertop 1
You pick up the apple 1 from the countertop 1.
> [Player] Act 7:  go to microwave 1
You arrive at microwave 1.  The microwave 1 is closed.
> [Player] Act 8:  heat apple 1 with microwave 1
You heat the apple 1 using the microwave 1.
> [Player] Act 9:  think:  The apple is now hot.  Next, I should place
it in fridge 1.
OK.
> [Player] Act 10:  go to fridge 1
You arrive at fridge 1.  The fridge 1 is open.
> [Player] Act 11:  move apple 1 to fridge 1
You move the apple 1 to the fridge 1.
> [Coach] Trial reflection:  Prioritize likely locations (countertops)
before exploring less relevant places.
Outcome:  success
```