
Playing ALFWorld with LLM-controlling by high-level RL policy

Jia Hong Chen

Department of Electrical Engineering
National Taiwan University
r11921051@ntu.edu.tw

Yu Hsiu Chou

Department of Electrical Engineering
National Taiwan University
r11942098@ntu.edu.tw

Chin Chieh Chao

Department of Electrical Engineering
National Taiwan University
r11942146@ntu.edu.tw

Tzu Hsu Wang

Department of Electrical Engineering
National Taiwan University
r11921a09@ntu.edu.tw

Abstract

In recent years, large language model (LLM) overwhelming internet. It could generate human-like text, translating language, answer professional question due to training data. Meanwhile, reinforcement learning (RL) is available to solve long-term task or task without ground truth answer. We aim to use many LLMs to cooperate with each other to solve tasks from ALFWorld, a text-based game. All of the LLMs received current state information and let RL agent choose the appropriate one to execute in the step.

1 Introduction

As numerous major corporations continue to introduce more abundant and sophisticated large language models (LLMs), users are increasingly integrating these models into their daily work. Prior researches, including Liu et al. [2], Park et al. [3], Wang et al. [5] and Zhu et al. [10], have explored the capabilities of these LLMs across diverse real-world tasks, revealing variations in performance across different model architectures. This raises an intriguing question: Can these LLMs, through collaborative efforts, achieve better advancements in tasks where individual models may exhibit comparatively weaker performance? Exploring the potential cooperative capabilities of these models across a spectrum of tasks becomes a key area of interest in understanding their collective impact on problem-solving scenarios.

In recent years, in addition to LLM, reinforcement learning has emerged as a focal point in contemporary research, situated within the realm of decision science. As a form of decision-making intelligence, reinforcement learning involves the study of optimal behaviors within a given environment to maximize rewards. This pursuit of optimal behavior is cultivated through interactive experiences, where the learning agent engages with and observes the responses of its environment. Analogous to the way children explore and learn actions that aid in accomplishing their objectives, reinforcement learning strives to navigate and adapt to its surroundings effectively. In light of this, our objective is to use the power of reinforcement learning to instruct an agent in the collaborative utilization of LLMs for task-solving.

Within the framework of Liu et al. [2], a notable trend has been identified—most LLMs tend to exhibit suboptimal performance when tasked with house-holding responsibilities. Recognizing this challenge, we have strategically decided to train on a specialized dataset designed to address such tasks. We decide to use Alfworld, proposed by Shridhar et al. [4], as our environment. By training on

Alfworld dataset, we aim to equip agents with a more tailored and contextually relevant knowledge base, fostering their ability to navigate, interpret, and respond effectively to house-holding challenges.

2 Related work

2.1 Prompting in Large language model

Wei et al. [6] explores the use of chain-of-thought prompting in large language models to improve their ability to perform complex reasoning tasks. The authors show that generating a chain of thought significantly improves the performance of language models on arithmetic, commonsense, and symbolic reasoning tasks.

The ReAct approach from Yao et al. [8] revolutionizes language models by combining reasoning and acting to enhance performance, interpretability, and trustworthiness across various domains. Unlike traditional models, ReAct addresses issues such as fact hallucination and error propagation by grounding reasoning in the external world and enabling reactive reasoning and knowledge updates. This innovative paradigm has demonstrated its superiority in tasks such as multi-hop question answering and fact verification, showcasing its potential for general task solving and few-shot learning setups.

2.2 Reinforcement learning and LLM

Cumulative Reasoning (CR) from Zhang et al. [9] is a method that leverages large language models to solve complex problems by decomposing tasks into smaller, manageable steps. CR involves three components: the proposer, verifier, and reporter, which work iteratively to propose, verify, and report potential propositions. This work presents empirical analyses demonstrating CR’s effectiveness, including its superior performance on logical inference tasks, the Game of 24, and the MATH dataset. Additionally, the document provides comparison results for various methods on LogiQA, ProofWriter, and FOLIO-val datasets, showcasing CR’s competitive accuracy and efficiency.

3 Problem formulation

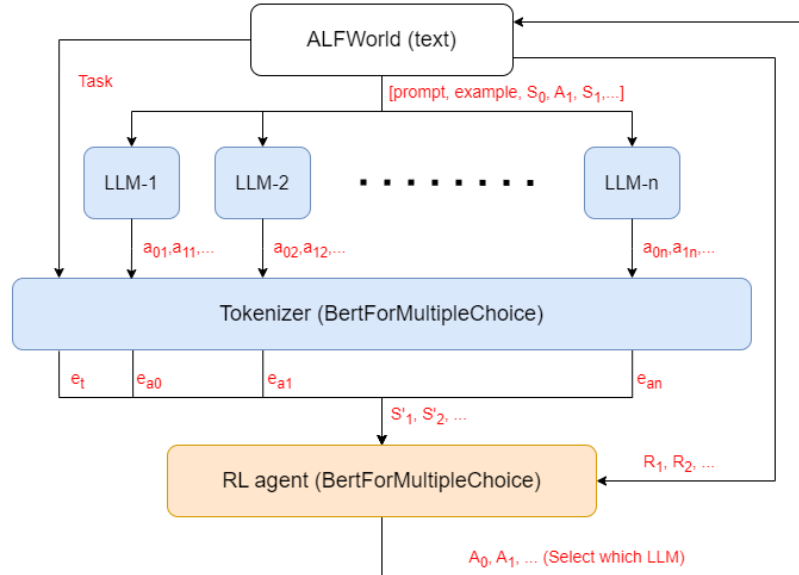


Figure 1: Overview of our model structure

At each step, ALFWorld outputs the environment information ($S_t, t = 0, 1, \dots, T$), feeding into each LLM with prompt and example. Collecting the actions output from each LLM ($a_{ti}, i = 0, 1, \dots, n$) and concatenating with task, encoding by the tokenizer as the RL agent input state (S'_t). RL agent will choose the most suitable action (A_t) to interact with the environment. Our main purpose is to maximize the success rate under the constraint of retry times.

4 Method

In Figure 1, we present a comprehensive overview of our system architecture. The observations from Alfworld undergo distinct preprocessing process for both Language Models (LLMs) and the reinforcement learning (RL) agent, ensuring that the input is appropriately tailored for each component.

Before reaching the RL agent, the preprocessed observations and the output from LLMs are strategically combined, establishing a symbiotic relationship between language understanding and decision-making capabilities. This combination empowers the RL agent to make informed choices based on a holistic comprehension of the environment.

Finally, the RL agent’s decisions trigger change within Alfworld. The rewards provided by Alfworld create a closed loop where the RL agent continually refines its decision-making process in response to the environment’s feedback.

The following provides a detailed breakdown of each component’s specific functionalities and information.

4.1 Environment

ALFWorld is an cross-modal framework featuring a diversity of embodied tasks with analogous text-based counterparts. Since both components are fully interactive, agents may be trained in either the language or embodied world and evaluated on test tasks in either modality. However, we only focus on the text-based part in this work.

4.1.1 Observation

After the environment is reset, a comprehensive overview is provided, including tables, beds, cabinets, and more. Additionally, the environment’s goal is reveal, such as ‘examine the alarm clock with the desk lamp’. Subsequent observation are driven by various actions, each triggering different result. For instance, executing the action ‘go to desk 1’ prompts the environment to output the items on that desk, like a desk lamp, pen, or book.

4.1.2 Action

Choose one of LLM output to interact with the environment.

4.1.3 Reward

There will be a positive reward R_{goal} if RL agent complete task before reaching the maximum number of steps. Otherwise, a negative penalty R_{fail} is given. If RL agent chooses an action to execute, it may be valid or invalid. Both wrong format and failed executed command are invalid. A negative penalty $R_{invalid}$ is given. If a valid action is chosen. We will match task string and ALFWorld output string and return reward as [7]. Giving a reward R_{action} . Last thing is about thinking. Since thinking is time-consuming, sometimes is not necessary, too much thought should be avoided. There will be a thought penalty $W_{thought} * N_{thought}$. Multiply the weight of thought and current number of thought. If the thought is useful, the match score of task string and thought is high, thought reward $R_{thought}$ is given. $R_{thought}$ is calculated in a same way as action reward.

$$R_{action} = \frac{|T_{object} \cup O_{object}|}{|T_{object}|} \quad (1)$$

Where:

- T_{object} is the objects mentioned in task.
- O_{object} is the objects mentioned in observation after execute action.

Parameter	Value
R_{goal}	1
R_{fail}	-1
$R_{invalid}$	-1
$W_{thought}$	-0.05

4.2 Large language model

We select recently released large language models for our collaborative framework. Due to budget constraints, we opted not to include GPT-4 or GPT-3.5 as our candidates. Instead, our candidates consist of large language models released by Google, including text-bison@001, text-bison@002, and Gemini-pro. Notably, text-bison@001 is included twice in our considerations due to the availability of two distinct API versions released by Google. In addition to the Google models, we’ve also incorporated LLaMA2, a large language model released by Meta. In total, we have five prominent language models serving as candidates for our collaborative reinforcement learning approach. This diverse ensemble allows us to explore the unique capabilities offered by each model, contributing to the robustness and adaptability of our collaborative agent within the defined environment.

4.2.1 Prompt

At the beginning of the game, all the LLMs are given same initial prompt and an example matching the type of current task. The purpose of prompt is telling the LLMs about the process, including introducing ALFWorld, available output, meaning of ALFWorld output and the existence of other LLMs. Coming afterward is one example in the same type. Letting LLMs know the environment description, thinking at first and learning available action format. See supplementary material for details.

4.2.2 Output

LLMs output either thought or action. The difference between them is that thought will not be sent to ALFWorld but action will be sent to ALFWorld and get a response.

4.3 Deep reinforcement learning

As highlighted earlier, our strategy centers around using reinforcement learning to make collaboration among large language models. This collaborative approach aims to unite the strengths of these models, enabling them to work together to solve complex tasks.

4.3.1 Preprocess

Since the constraints related to token count, we’ve strategically chosen to capture essential information in the observation. This includes preserving the environment’s goal and combining the outputs from all LLMs. This informative observation serves as a concise representation of critical aspects of the ongoing interaction, empowering the agent to make informed decisions within the constrained token space. Furthermore, to match to the input specifications of BERT, we add the tokenization step in our pipeline. The combined results are tokenized using the BERT tokenizer. This crucial preprocessing step ensures that the input data is appropriately formatted to meet the token-level requirements of the BERT model.

4.3.2 Model

As our environment is text-based, we need an agent capable of language comprehension. We have selected Bidirectional Encoder Representations from Transformers (BERT), proposed by Devlin et al. [1], to fulfill the roles of both actor and critic within our framework. The choice of BERT is its remarkable capacity to capture intricate linguistic nuances and contextual dependencies, aligning

with the demands of our language-centric tasks. Especially, we finetune bert-base-uncased with BertForMultipleChoice interface, which is a pretrain model loaded from Huggingface¹.

4.3.3 Algorithm

In the training phase, we leverage the stable-baselines3 library², employing the A2C algorithm to fine-tune our language-aware agent. The stable-baselines3 library provides a robust and versatile set of tools for reinforcement learning, allowing us to seamlessly integrate the A2C algorithm into our training pipeline. The A2C algorithm, known for its efficiency and effectiveness in training reinforcement learning agents, serves as the backbone for enhancing the decision-making capabilities of our language model. By combining the power of stable-baselines3 with the A2C algorithm, we aim to achieve optimal convergence and performance, ensuring that our language-aware agent is well-equipped to collaboratively solve tasks within the defined environment.

5 Experiment

We evaluate our models on unseen tasks of Alfworld. To avoid any syntactic errors, we provide a domain-specific trajectory to the agent. The examples of these trajectories can be found in the supplementary material.

5.1 Environment Selection

We selected 12 environments across 6 task types that preliminary testing indicated had lower success rates with the Bard model. The task types included Pick & Place, Examine in Light, Clean & Place, Heat & Place, Cool & Place, and Pick Two & Place. The specific environments within each type were chosen to represent diversity in topics, step horizons, and other factors that could impact the model’s performance.

5.2 Reinforcement Learning Agents

These are the LLMs available for us.

- Bardfree: A version of PaLM 2 without model updates since its release using the Google AI Studio API.
- Bard: Same model as Bardfree but using Google Cloud Vertex AI API.
- Bard2: An updated version of PaLM 2 with enhanced quality using Google Cloud Vertex AI API.
- LLaMA2 13B: 13B fine-tuned LLaMA2 model from Meta.
- Gemini pro: The Gemini Pro, a model recently released by Google.

We trained reinforcement learning agents with 3 LLMs, 4 LLMs and 5 LLMs:

- 3 LLMs: Bardfree, Bard2, Gemini pro
- 4 LLMs: LLaMA2 13B, Bardfree, Bard, Bard2
- 5 LLMs: LLaMA2 13B, Bardfree, Bard, Bard2, Gemini pro

5.3 Results

As for the baselines, we interacted with Alfworld using a single LLM or randomly selected the action from one of the LLMs at each step. Across the 12 environments, the performance of 4 LLMs and 5 LLMs RL agents exceeded that of all baseline models, as shown in Fig. 2. Consistent with our expectations, the API-based LLMs achieved higher results compared to the open-sourced LLMs. However, our models only moderately outperformed the top-performing baselines. We speculate that this is because the reinforcement learning agent simply learns to choose the best LLM option without

¹<https://huggingface.co/bert-base-uncased>

²<https://stable-baselines.readthedocs.io/en/master>

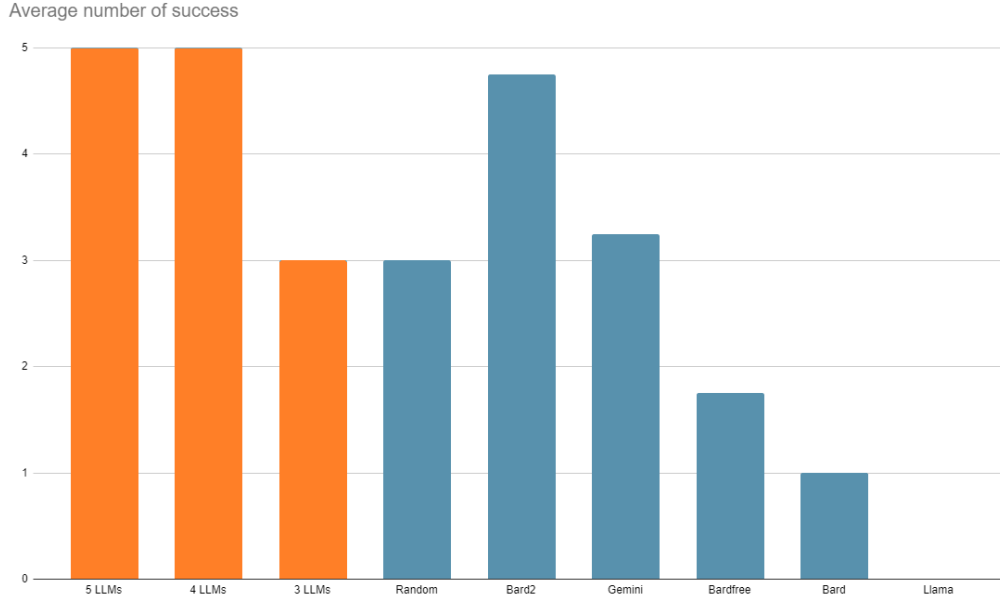


Figure 2: Average number of success of our models and baseline models

fine-tuning the LLMs. Consequently, the agent does not enhance each model’s specific skill, and the weakest options may impede overall performance.

6 Conclusion

We propose a high-level RL policy to ensemble LLMs to play ALFWorld. Although the performance didn’t match our expectation, to choose different LLMs in different stage, we still came out with some results. First thing is training with LLMs is very time-consuming, it could take days to figure out if the setting is useful. API-based LLMs are stronger but the stability depends on the remote server, it could be unstable for some reason. Second thing is that RL agent learns to continuously choose the same LLM, since one of them performs much better than others. Last but not least, many researches put a great effort into prompting LLMs with different ways, hope to get better result. We thought that more researches on prompt design should be done.

7 Supplementary Material

7.1 Prompt

Here’s the initial prompt:

Interact with a household to solve a task. Imagine you are an intelligent agent in a household environment and your target is to perform actions to complete the task goal. At the beginning of your interactions, you will be given the detailed description of the current environment and your goal to accomplish. For each of your turn, you will be given a list of actions which you can choose one to perform in this turn. You have two choice: 1. Directly output the action in this turn. Output format: Your next action. 2. You should first think about the current condition and plan for your future actions, and then output your action in this turn. Output format: THOUGHT: Your thoughts. After each turn, the environment will give you immediate feedback based on which you plan your next few steps. If the environment output "Nothing happened.", that means the previous action is invalid and you should try more options; if the environment output "OK.", that means you did not do anything to the environment. You have better do action in next step. Last but not least, if you think too many times, the reward you acquire will be gradually decreased. So, try to react to the environment after you think.

7.2 Example

Here's one example:

Alfworld: You are in the middle of a room. Looking quickly around you, you see a cabinet 9, a cabinet 8, a cabinet 7, 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 9, a drawer 8, a drawer 7, a drawer 6, a drawer 5, a drawer 4, 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: cool some cup and put it in shelf.

Agent: THOUGHT: To solve the task, I need to find and take a cup, then cool it with fridge, then put it in shelf.

Alfworld: OK.

Agent: THOUGHT: First I need to find a cup. A cup is more likely to appear in cabinet (1-9), coffeemachine (1), countertop (1-3), drawer (1-9), sinkbasin (1), diningtable (1), fridge (1), garbagecan (1), microwave (1), shelf (1-3), stoveburner (1-4), toaster (1). I can check one by one randomly.

Alfworld: OK.

Agent: go to drawer 3

Alfworld: The drawer 3 is closed.

Agent: open drawer 3

Alfworld: You open the drawer 3. The drawer 3 is open. In it, you see a fork 1.

Agent: go to stoveburner 1

Alfworld: On the stoveburner 1, you see a pot 2.

Agent: go to stoveburner 4

Alfworld: On the stoveburner 4, you see nothing.

Agent: go to cabinet 8

Alfworld: The cabinet 8 is closed.

Agent: open cabinet 8

Alfworld: You open the cabinet 8. The cabinet 8 is open. In it, you see a kettle 2, and a vase 2.

Agent: go to drawer 5

Alfworld: The drawer 5 is closed.

Agent: open drawer 5

Alfworld: You open the drawer 5. The drawer 5 is open. In it, you see a saltshaker 1.

Agent: go to shelf 1

Alfworld: On the shelf 1, you see a cup 1.

Agent: THOUGHT: Now I find a cup (1). Next, I need to take it.

Alfworld: OK.

Agent: take cup 1 from shelf 1

Alfworld: You pick up the cup 1 from the shelf 1.

Agent: THOUGHT: Now I take a cup (1). Next, I need to go to a fridge 1 and cool it.

Alfworld: OK.

Agent: go to fridge 1

Alfworld: The fridge 1 is closed.

Agent: cool cup 1 with fridge 1
 Alfworld: You cool the cup 1 using the fridge 1.
 Agent: THOUGHT: Now I cool a cup (1). Next, I need to put it in/on shelf 1.
 Alfworld: OK.
 Agent: go to shelf 1
 Alfworld: On the shelf 1, you see nothing.
 Agent: put cup 1 in/on shelf 1
 Alfworld: You put the cup 1 in/on the shelf 1.
 You (Agent) win!!!

References

- [1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.
- [2] X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men, K. Yang, S. Zhang, X. Deng, A. Zeng, Z. Du, C. Zhang, S. Shen, T. Zhang, Y. Su, H. Sun, M. Huang, Y. Dong, and J. Tang. Agentbench: Evaluating llms as agents, 2023.
- [3] J. S. Park, J. C. O’Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein. Generative agents: Interactive simulacra of human behavior, 2023.
- [4] M. Shridhar, X. Yuan, M.-A. Côté, Y. Bisk, A. Trischler, and M. Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning, 2021.
- [5] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023.
- [6] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.
- [7] S. Yao, H. Chen, J. Yang, and K. Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents, 2023.
- [8] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models, 2023.
- [9] Y. Zhang, J. Yang, Y. Yuan, and A. C.-C. Yao. Cumulative reasoning with large language models, 2023.
- [10] X. Zhu, Y. Chen, H. Tian, C. Tao, W. Su, C. Yang, G. Huang, B. Li, L. Lu, X. Wang, Y. Qiao, Z. Zhang, and J. Dai. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory, 2023.