OEA: Online Environmental Adaptation for Task-Oriented Language Agents

Anonymous ACL submission

Abstract

Recent advancements in the field of intelligent agents, especially those leveraging large language models, have been impressively substantial. However, these models still encounter significant challenges in interactive and dynamic scenarios, such as online shopping, mainly due to their lack of knowledge of the current environment. In this paper, we propose an innovative online method for environmental adaptation. The trajectories generated by large language models during task execution are utilized to update a global action-observation tree. When encountering new tasks, our method transforms the action-observation tree into text and integrates this information into the context to aid the model in solving the task. This iterative process enables the model to progressively enhance its understanding of the environment, resulting in steadily improved performance over time. Our approach obviates the need for offline fine-tuning and serves as a versatile plug-and-play solution applicable to various scenarios. In two widely-used environments, Webshop and Alfworld, our method has significantly improved performance beyond ReAct and Reflection, achieving higher accuracy and reducing the required token expenditure.

1 Introduction

In the constantly evolving field of Natural Language Processing (NLP), the issue of distribution discrepancies between the downstream tasks and the pretraining corpora has always been a critical area of research. In the past, researchers have proposed various simple yet effective adaptation methods to address inconsistencies at both the domain and task levels, as detailed in Table 1. However, with the rapid advancement of large language model agents and their real-world applications, a new form of distribution discrepancy has become increasingly prominent: the environmental level. This discrepancy is particularly evident in real-world scenarios that involve interactive decision-making, such as online shopping, where each action can modify the web page environment. Since models were not pretrained with knowledge of various environmental interactions, and given that the real-world environment is constantly changing, achieving efficient planning and rational operation by the language model remains a challenge.

Method	Level		
	Domain	Task	Environment
Gururangan et al. (2020)	✓	✓	
Wu et al. (2021)	✓		
Li et al. (2021)		✓	
Ours			✓

Table 1: Comparison between our method and previous adaptive methods across different problem settings.

To address this issue, we propose an online environment adaptation method. Specifically, for each environment, we explicitly iterate and update a global action-observation tree. Initially, trajectories generated from the interaction between the language agent and the environment are transformed into abstract action-observation sequences through the in-context learning approach. These action-observation sequences are then integrated into the global tree. Subsequently, the sequences of nodes from the global action-observation tree are transformed into textual information through an in-context learning approach and integrated into the context. This enables the language model to utilize this information for more accurate and effective resolution of the subsequent task. The newly generated trajectories during the task resolution are then used to further update the global tree. Through this iterative process, the model progressively enhances its understanding of the environment, thereby increasing the success rate of task resolution. This entire process does not require additional human annotation of environmental information, facilitating the language model's adaptation to the environment. Compared to offline fine-tuning methods, our approach fully leverages the language model's capability for in-context learning, achieving online adaptation. So our approach more conveniently and swiftly addresses the challenges posed by the dynamic updates often seen in real environments, such as web page updates or changes in the placement of objects in embodied scenarios, among others.

The contributions of our study are threefold:

1. To tackle the challenge of large language models lacking environmental knowledge in real-world sequential decision-making scenarios, we propose an online, dynamic method for environmental adaptation. This approach allows the model to increasingly understand its environment as it completes more tasks, leading to progressively better performance.

2. Leveraging the in-context learning of large language models, our method requires no offline fine-tuning. This plug-and-play approach is well-suited for the ever-changing and updating environments of the real world.

3. In two typical environments, Webshop and Alfworld, our method has significantly enhanced the performance of both ReAct and Reflection. Not only is the accuracy higher, but the number of required steps and trials is also reduced.

2 Related Work

2.1 Language Agent

The field of research concerning intelligent agents based on language models is experiencing rapid advancements. This encompasses a spectrum from the classical enhancement of reasoning capabilities like the Chain of Thought (Wei et al., 2022) to the representative tool-utilizing approaches such as AutoGPT and HuggingGPT (Shen et al., 2023) in planning and solving paradigm. Researchers have taken a step further by exploiting the feedback and assessment capabilities inherent in language models, introducing innovations like the Tree of Thought (Yao et al., 2023) and Graph of Thoughts (Besta et al., 2023), aiming to solve increasingly complex problems. The issues addressed by these methodologies, such as solving mathematical problems or planning how to schedule APIs given specific tools and problems, inherently contain comprehensive information, so language models can leverage their intrinsic knowledge to achieve global planning.

However, in more realistic scenarios (Liu et al., 2023; Ma et al., 2024), such as online shopping on websites (Yao et al., 2022a) or benchmarks like TextWorld (Shridhar et al., 2021), every action incurs varying changes in the environment, necessitating models to make sequential decisions. ReAct (Yao et al., 2022b) incorporates environmental feedback to support reasoning, and Reflexion (Shinn et al., 2023) combines internal and external feedback. The method we propose assists these agents in explicitly providing and adaptively updating environmental information, thereby enabling them to make more reasonable plans and decisions.

2.2 Adaption on Language Model

In the constantly evolving field of Natural Language Processing (NLP), the issue of distribution discrepancies between the downstream tasks and the pretraining corpora has always been a critical area of research. In the past, researchers have proposed various simple yet effective unsupervised adaptation methods, such as domain-adaptive pretraining (Gururangan et al., 2020; Wu et al., 2021) and task-adaptive pretraining (Li et al., 2021). Unlike previous work, our focus is on the adaptability of language models within environments. Furthermore, instead of relying on continuous pretraining methods, we employ in-context learning to incorporate continuously updated environmental information into the context. This strategy facilitates the online adaptability of large language models to their environments.

2.3 Environmental Exploration and Modeling in Reinforcement Learning

In the domain of Reinforcement Learning (RL), several classic exploration strategies have been proposed, including the Epsilon-Greedy method (Sutton and Barto, 2018), the incorporation of curiosity-driven learning models (Pathak et al., 2017; Burda et al., 2018a), rewarding the agent for encountering unpredictable states and actions (Burda et al., 2018b), and the combination of count-based exploration with environments that provide sparse rewards (Ostrovski et al., 2017). Our approach also explores the environment during the pre-task phase, primarily utilizing the large language model's own knowledge and reasoning capabilities to explore key information about the environment as much as

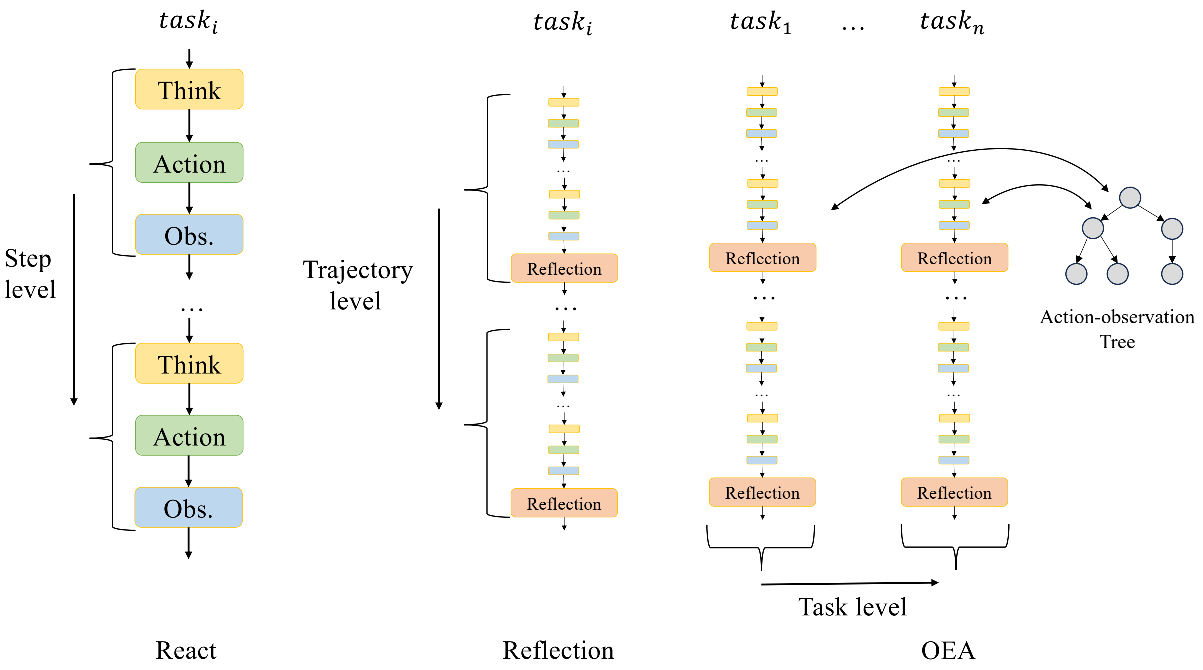

Figure 1: Comparison of previous methods with our 0EA approach in terms of acquiring environmental information. ReAct accumulates environmental information across different steps, while Reflection can accumulate this information across different trajectories. In contrast, 0EA enables language models to adapt to the environment across different tasks. Our approach explicitly maintains a global action-observation tree. The trajectories generated during task execution are used to update this global tree, which is then converted into textual form as environmental interaction information and incorporated into the context to further facilitate the execution of subsequent tasks.

possible.

At the same time, for reinforcement learning, environmental modeling stands as a pivotal component, aiding agents in the learning and decision-making processes. A widely-adopted approach to environmental modeling involves leveraging Markov Decision Processes (MDP) to formalize environments, as portrayed by Sutton and Barto (2018). Delving into deep learning, neural network models emerge as quintessential tools for approximating environmental dynamics. Studies by Mnih et al. (2015) explicate the integration of Convolutional Neural Networks (CNN) to approximate environmental states in RL. The works by Ha and Schmidhuber (2018) serve as exemplary instances of employing world models, allowing agents to predict future states and formulate optimal policies, thereby enhancing their efficacy across various tasks. Contrary to environmental modeling in reinforcement learning, our approach explicitly maintains a global action-observation tree. During task execution, this tree is converted into textual information and added to the context to help the agent better solve the task.

3 Method

As shown in Figure 1, previous frameworks for interaction between large language models and the environment primarily accumulate environmental information and learn interaction methods at different step levels or trajectory levels within a single task. However, within the same environment, language models should also share the environmental information and interaction methods acquired when executing different tasks.

Specifically, for each environment, we explicitly iterate and update a global action-observation tree. The entire algorithmic process is illustrated in Algorithm 1. In practice, to provide the language model with an initial understanding of the environment before task execution, we have designed *Pre-Task Environment Exploration* in addition to In-Task Environment Adaptation. In this phase, the Large Language Model (LLM) does not execute specific tasks but aims to survey the environment, exploiting the profound knowledge amassed during its pre-training phase to select actions that optimally explore environmental information. Both Pre-Task Environment Exploration and In-Task Environment Adaptation include the following core iterative mechanism:

1. Trajectories generated from the interaction between the language agent and the environment are transformed into abstract action-observation sequences through the in-context learning approach. For raw trajectories, we retain only the sequences of action and observation pairs. For instance, in the step involving the ReAct method, we remove 'think,' and for consecutive repetitive action-observation pairs, we merge them into a single entity. Once we obtain clean action-observation sequences, they will be abstracted if needed. For example, in the webshop environment, the action "search [3 ounce bright citrus deodorant sensitive skin]" is transformed into "search [Keywords based on given requirements and conditions]". Similarly, "click[B078GWRC1J]" is converted to "click[product]". Then we guide the language model to perform these conversions in an in-context learning manner through instructions and a few common examples of action and observation transformations. This language model for conversion can be the same as or different from the one executing the tasks.

2. These action-observation sequences are then integrated into the global tree. In practice, a sequence of action-observation pairs can be considered as a subtree, where each pair serves as the successor node to the previous one. In our method, nodes encapsulate both actions and observations, while edges represent the temporal sequence, indicating the progression from one to the next. Starting from the root node, identical nodes are merged, and their frequency is incremented by one. If the nodes are different, a new node is initialized on the global tree, with its frequency set to 1.

3. The sequences of nodes from the global action-observation tree are transformed into textual information through an in-context learning approach and integrated into the context. To achieve this, we utilize breadth-first search algorithms to identify the shortest paths from the root to each leaf node, subsequently concatenating the nodes encountered along these paths into sequences. Initially, these sequences may not form grammatically correct sentences. We guide the large language model to transform these sequences into grammatically correct environmental descriptions by providing task instructions and several examples of common transformations. These descriptions are then sampled in a non-repetitive manner, based on the frequency of the leaf nodes, using normalized frequencies as the probability distribution, until the cumulative token count of the sampled sentences aligns with the predetermined maximum context length allocated for environmental information.

4. Utilizing this refined environmental information, the language model is then able to more accurately and effectively tackle tasks, setting the stage for the commencement of the subsequent iteration cycle.

Algorithm 1 Online Environmental Adaptation Algorithm

```
Global: Action-Observation Tree $G_{\text{tree}}$, Environment $E$
Procedure PreTask ($E$, $LLM$,$trails$)
    for $i = 1$ to $trails$ do
        $text_{\text{env}} \leftarrow$ Convert($G_{tree}$)
        $traj \leftarrow$ Explore($E, LLM, text_{\text{env}}$)
        $tree \leftarrow$ Convert($traj$)
        $G_{tree} \leftarrow$ Merge($tree, G_{tree}$)

Procedure InTask ($E$, $LLM$, $Tasks$)
    for each $T$ in $Tasks$ do
        $text_{\text{env}} \leftarrow$ Convert($G_{tree}$)
        $traj \leftarrow$ Generate($E, LLM$,T$, text_{\text{env}}$)
        $tree \leftarrow$ Convert($traj$)
        $G_{tree} \leftarrow$ Merge($tree, G_{tree}$)
```

In practical applications, these two phases can either be interpreted within a unified framework or as distinct methodologies. The efficacy of the task performance is directly proportional to the extent of the pre-task environment exploration; however, this implies a reduced benefit from the in-task environment update, and vice versa. In real-world scenarios, if the objective is to ensure optimum and stable performance of the intelligent agent, consideration should be given to allocating additional computational resources for exploration. If resources are constrained, emphasis can be placed on in-task environment update to acquire environmental information through the trajectory of task execution, necessitating minimal additional resources.

Our method does not require additional human annotation of environmental information, achieving environmental adaptability for the language model. It is universally applicable to various interaction environments due to its general nature. Furthermore, our approach fully leverages the in-context learning capabilities of language models, facilitating online adaptation. Compared to offline fine-tuning methods, it more conveniently and rapidly addresses issues associated with the dynamic updating of real-world environments, such as updates to web pages or or changes in the placement of objects in embodied scenarios, among others.

4 Experiment

4.1 Baselines Setting

We utilize two popular methods for intelligent agent-environment interactions as our baselines: ReAct (Yao et al., 2022b) and Reflection (Shinn et al., 2023). For a detailed introduction, please see the appendix. It is noteworthy that the original papers adopted the text-davinci-003 as the core language model; however, as OpenAI will soon cease to offer this API, we opted for more prevalent language models. Consequently, the core language models available for the agent are GPT-3.5-turbo and GPT-4, respectively. The reported experimental results are the average of multiple runs.

4.2 Benchmark

We conducted evaluations on two benchmarks:

WebShop (Yao et al., 2022a) is a sophisticated web-based problem-solving benchmark WebShop designed to assess agents' ability to adeptly navigate through an e-commerce website, with the objective to accurately locate and secure products in response to client requests. This environment is enriched with a diverse array of 1.18M real-world products accompanied by 12k articulate human instructions. WebShop encapsulates an extensive variety of both structured and unstructured texts, including product titles, descriptions, and diverse options meticulously crawled from Amazon. The evaluation of this intricate task is orchestrated through an average score, representing the percentage of desired attributes covered by the selected product across all episodes. Our evaluations are conducted meticulously on the first 500 distinct test instructions, ensuring comprehensive assessment and validation of the agents' proficiency in managing complex, real-world e-commerce navigations and transactions.

ALFWorld-Eco . We have developed ALFWorld-Eco, a new environment based on ALFWorld. Shridhar et al. (2021) created the ALFWorld to align with the embodied ALFRED benchmark (Shridhar et al., 2020), is a sophisticated, synthetic text-based game designed to challenge agents to navigate and perform multi-step tasks within a range of interactive environments. It incorporates six diverse types of tasks, each requiring the agent to accomplish a high-level goal, like examining a paper under a desk lamp, by navigating and interacting with a simulated household through textual actions (e.g., go to coffee table 1, take paper 2, use desk lamp 1). The original ALFWorld dataset is constructed with a task-centric focus, containing 134 different tasks in the unseen dataset, with nearly every task corresponding to a distinct environment. This approach does not align with our objective, as we aspire to build a new dataset centered around environments, allowing the execution of multiple tasks within a single environment. Consequently, we have developed ALFWorld-Eco, a new environment based on ALFWorld, where agents can complete 60 different tasks, categorized into six types, with each category comprising ten specific tasks. We employ the success rate as our metric for evaluation. The agent is assigned a score of 1 upon the successful completion of the task, and conversely, it receives a score of 0 if it fails to complete the task. We will release this new benchmark alongside the corresponding code. Our ALFWorld-Eco focuses on fostering a more diversified interaction within each environment, enabling agents to achieve a wider range of tasks and goals. This shift from task-centric to environment-centric design empowers agents to better understand and adapt to varying contextual circumstances within the same environment, enhancing their flexibility and applicability in real-world scenarios.

4.3 Parameter Setting

For GPT-3.5-turbo and GPT-4, we specify the parameters for related API calls. The temperature is meticulously set to zero, ensuring the output is strictly limited to a maximum token count of one hundred. The top-p value is securely fixed at one,

LLM	Method	Environment	
		Webshop	ALFWorld-Eco
GPT-3.5	ReAct	65.0	11.7
	ReAct + OEA	70.2	18.3
	Reflection	68.6	45.8
	Reflection + OEA	73.5	54.2
GPT-4	ReAct	59.7	83.3
	ReAct + OEA	67.4	89.6
	Reflection	69.5	94.4
	Reflection + OEA	73.8	96.1

Table 2: Experimental Results of OEA on Webshop and ALFWorld-Eco

with both frequency and presence penalties effectively set to zero.

During the pre-task exploration phase, we establish the number of exploration trials as 5. When integrating our method with ReAct and Reflection, it is only necessary to add an additional module for environment interaction information in the context. For this component, we set its maximum length to 512 tokens. The code, along with the newly constructed environments and tasks, will be made open source.

4.4 Results

Table 2 demonstrates that our OEA method consistently yielded significant improvements across various baseline methods (ReAct and Reflection), environments (WebShop and ALFWorld-Eco), and language models (GPT-3.5-turbo and GPT-4), underscoring its effectiveness and broad applicability. In a detailed comparison, several conclusions can be drawn:

- Environmental Comparison: Our method exhibited a more pronounced improvement in ALFWorld-Eco compared to WebShop. This distinction suggests that tasks in ALFWorld-Eco require a deeper reliance on environmental knowledge, underlining the importance of our adaptation approach in scenarios where understanding the environment is crucial for task success.

- Methodological Comparison: When comparing the improvements provided by our OEA method to the baseline approaches, a greater enhancement was observed with ReAct than with Reflection. This discrepancy may be attributed to Reflection's inherent inclusion of some environmental interaction information within its reflective process, thus presenting a smaller scope for our method to offer additional improvements.

- Language Model Impact: The performance impact varied significantly with the size of the language model. Larger models, such as GPT-4, demonstrated more substantial performance gains, indicating that our adaptation method effectively leverages the increased capacity of larger models to enhance task performance.

We further conducted a bad case analysis on ReAct and ReAct+ OEA, identifying three types of errors related to Webshop environmental interaction information:

1. Executing the search action directly on the product page instead of returning to the search page;

2. Encountering an error due to the agent selecting the [Next] button in the environment;

3. Continuously browsing products without clicking 'buy now' until reaching the maximum number of steps.

Large language models equipped with environmental adaptation largely avoided the aforementioned errors. The main cases where they failed were when they reached the maximum number of steps during multiple refinements of search keywords for products.

4.4.1 Average Success Rate across Different Stage

While the main results from the previous section highlight an increase in the final average success rate, they do not showcase the gradual im-

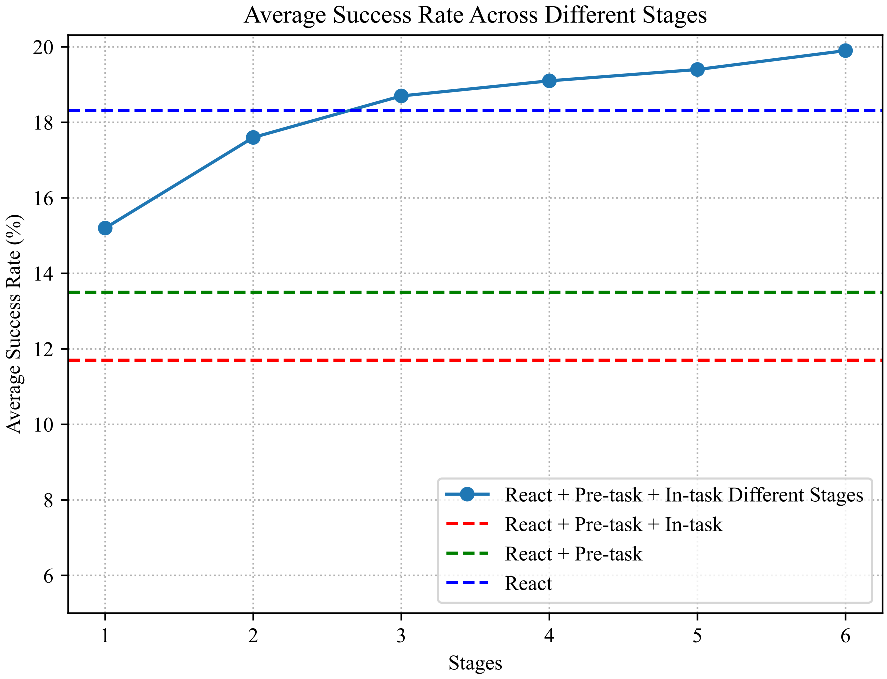

Figure 2: Average Accuracy across Different Stages on ALFWorld-Eco

provement in success rate during the adaptive process. Therefore, in this section, we delve even further into analyzing the advantages yielded by environmental adaptation during the in-task phase within the ALFWorld-Eco environment. Specifically, 60 tasks are randomly assorted, into six distinct groups, each encompassing 10 tasks. The average accuracy is then calculated within each group, representing the accuracy for varying stages during the in-task phase. To mitigate the potential bias introduced by varying difficulties of different tasks, we conduct a total of five rounds of repeated experiments. Subsequently, the average accuracy of the six groups at diverse stages is once more averaged per round to denote the final accuracy. As shown in Figure 2, there is a noticeable increase in accuracy in the later stages, illustrating that, during the in-task phase, the agent's continuous adaptation to the environment systematically amplifies its capability to resolve tasks. The figure presents the ablation performance of the method, where "ReAct" indicates the exclusive application of the ReAct method without any environmental adaptation. "ReAct+Pre-task" denotes that the examination only occurs in the pre-task phase, with no updates occurring in the in-task phase. A comparison of these results reveals that both exploration and updating are crucial for the agent's performance in the ALFWorld-Eco environment, underscoring the importance of the two components.

Ablations	Webshop
Ours	70.2
w/o Frequencies	68.5
w/o Abstract Tree	68.8
w/o Grammatically Correct	66.9
w/o Pre-task Exploration	69.3
w/o In-task Adaptation	67.1
w/o all (React only)	65.0

Table 3: Summary of Ablation Studies in the Webshop Environment Using GPT-3.5-turbo

4.4.2 Ablation Studies

In our ablation studies conducted within the Webshop environment using the GPT-3.5-turbo model, we explored a variety of ablation experiments. These experiments included ablations on strategies such as using random frequencies instead of normalized frequencies as the probability distribution, constructing and merging trees directly from raw text without converting them into an abstract action-observation tree, and adding node sequences to the context without converting them into grammatically correct text. Additionally, we performed ablations on phases by eliminating the pre-task exploration phase and the in-task exploration phase. These ablations helped us understand the impact of

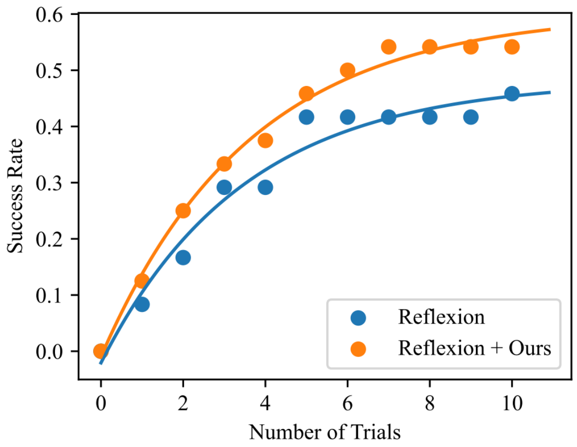

Figure 3: Success Rate across Different Trials in the ALFWorld-Eco

different components and stages on our method's overall performance.

Table 3 shows that at the strategy level, the failure to convert sequences into grammatically correct sentences significantly affects model performance. When trees are constructed and merged directly from raw text, without being converted into an abstract action-observation tree, the resulting global tree is large, with most nodes' frequencies essentially being 1. This situation is almost the same as not using normalized frequencies as the probability distribution, which explains why the results of the two ablations in the table are similar. Moreover, from a phase perspective, In-task Adaptation exerts a greater influence on the model than Pre-task Exploration.

4.4.3 Success Rate across Different Trials

Using the GPT-3.5-turbo model in the ALFWorld-Eco environment, we further explored the model's performance across various trials. Figure 3 demonstrates that our method consistently improves the model's performance across different trials. In other words, to achieve the same success rate, our method allows the model to require fewer trials.

4.4.4 Token Usage Across Different Methods

Although the integration of environmental information modules in the context consumes additional tokens, the results from the previous section have shown a reduction in the average number of trials. This reduction prompts a further investigation into whether the overall token usage has increased or decreased. Using the same GPT-3.5-turbo model within the ALFWorld-Eco environment, we analyzed the token usage for two methods (Reflection and Reflection + OEA) with the maximum number of trials set to 10. The results reveal that our method often succeeds in fewer trials and concludes earlier. Therefore, despite the addition of extra environmental information in the context, the total number of tokens used is, in fact, lower by 8%. This outcome suggests that our approach, by efficiently reducing the number of trials needed for success, compensates for the additional token expenditure on environmental information, ultimately leading to a decrease in overall token usage.

5 Conclusion

Our approach aims to address the lack of knowledge about real-world environmental interactions in pretrained large language models. We propose an online Environmental Adaptation method to augment their decision-making processes both before and during task execution, proving to be pivotal in enhancing the effectiveness and adaptability of language agents in various applications. Moving forward, we plan to conduct evaluations using a wider array of agents and more authentic datasets. Additionally, we hope to explore how the integration of environmental information can enable the amalgamation of various global planning methodologies, further elevating the performance of sequential decision-making tasks that require interaction with the environment.

6 Limitation

Because our method utilizes text to describe environmental interactions, it is still subject to the limitations of limited context. If the environmental space is extremely large or in an open-domain setting, where it is challenging to describe the environment with limited text, the effectiveness of our approach would theoretically be impacted.

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, et al. 2023. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. 2018a. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. 2018b. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

David Ha and Jürgen Schmidhuber. 2018. World models. *arXiv preprint arXiv:1803.10122*.

Shiyang Li, Semih Yavuz, Wenhu Chen, and Xifeng Yan. 2021. Task-adaptive pre-training and self-training are complementary for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1006–1015, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Yuxian Gu, Hangliang Ding, Kai Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Shengqi Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. 2023. Agentbench: Evaluating llms as agents. *ArXiv*, abs/2308.03688.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. Agentboard: An analytical evaluation board of multi-turn llm agents.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. 2017. Count-based exploration with neural density models. In *International conference on machine learning*, pages 2721–2730. PMLR.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR.

Yongliang Shen, Kaitao Song, Xu Tan, Dong Sheng Li, Weiming Lu, and Yue Ting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *ArXiv*, abs/2303.17580.

Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. {ALFW}orld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Han Wu, Kun Xu, Linfeng Song, Lifeng Jin, Haisong Zhang, and Linqi Song. 2021. Domain-adaptive pretraining methods for dialogue understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 665–669, Online. Association for Computational Linguistics.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022a. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *ArXiv*, abs/2305.10601.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022b. React: Synergizing reasoning and acting in language models. *ArXiv*, abs/2210.03629.

Baselines Setting

We utilize two popular methods for intelligent agent-environment interactions as our baselines: ReAct (Yao et al., 2022b) and Reflection (Shinn et al., 2023).

- ReAct (Yao et al., 2022b) is a method that capitalizes on Large Language Models (LLMs) to generate reasoning traces and task-specific actions in an interleaved manner, thereby improving the interaction between reasoning and action. Reasoning traces assist the model in deducing, monitoring, and updating action plans, as well as in managing exceptions. Simultaneously, actions allow the model to interact with external sources, such as knowledge bases or environments, to acquire additional information. We utilized the original codebase, setting the maximum steps for ReAct to 50.

- Reflection (Shinn et al., 2023) introduces an innovative framework designed to enhance the learning of goal-driven language agents that interact with external environments by leveraging linguistic feedback rather than traditional weight updates. This method permits agents to verbally process feedback signals, storing these reflections in an episodic memory buffer to facilitate better decision-making in future attempts. It has shown considerable progress over baseline agents in tasks requiring sequential decision-making and coding, all without the necessity for extensive training samples or model fine-tuning. We utilized the original codebase, setting the maximum trials for Reflection to 10.