Teaching Embodied Reinforcement Learning Agents: Informativeness and Diversity of Language Use

Jiajun Xi* **Yinong He*** **Jianing Yang** **Yinpei Dai** **Joyce Chai**

University of Michigan

{jiajunxi, heyinong, jianingy, daiyp, chaijy}@umich.edu

Abstract

In real-world scenarios, it is desirable for embodied agents to have the ability to leverage human language to gain explicit or implicit knowledge for learning tasks. Despite recent progress, most previous approaches adopt simple low-level instructions as language inputs, which may not reflect natural human communication. It's not clear how to incorporate rich language use to facilitate task learning. To address this question, this paper studies different types of language inputs in facilitating reinforcement learning (RL) embodied agents. More specifically, we examine how different levels of language **informativeness** (i.e., feedback on past behaviors and future guidance) and **diversity** (i.e., variation of language expressions) impact agent learning and inference. Our empirical results based on four RL benchmarks demonstrate that agents trained with diverse and informative language feedback can achieve enhanced generalization and fast adaptation to new tasks. These findings highlight the pivotal role of language use in teaching embodied agents new tasks in an open world. [1]

1 Introduction

Developing embodied agents that can understand and communicate with humans in natural language to learn and accomplish tasks is a long-standing goal in artificial intelligence. In recent years, the integration of human language and reinforcement learning (RL) has seen significant advancements. Unlike traditional RL methods that typically rely on numerical reward signals to guide agent learning, recent works (Cheng et al., 2023; Lin et al., 2023) explore using language as an intuitive and useful signal to shape an agent's behaviors. For example, when the agent is making mistakes during the task completion, providing language feedback can largely improve the instantaneous performance thus enhancing the overall agent learning efficiency and effectiveness (McCallum et al., 2023).

However, existing methods generally employ simple instructions, such as *"turn left"* and *"put the apple to the table"* to teach/control an agent (Hanjie et al., 2021; Zhang and Chai, 2021; Lin et al., 2023; McCallum et al., 2023; Shridhar et al., 2021). While useful, these instructions may not fully reflect the flexibility of language use in task learning and collaboration (Chai et al., 2018, 2019; Zhang et al., 2022, 2023; Dai et al., 2024a). In the real world, humans often express complex language instructions that are more ***informative***. For instance, when a student makes a mistake, a teacher may help them to retrospect on what went wrong (i.e., *hindsight instructions*) and then guide them on what should be done next to finish the goal (*i.e., foresight instructions*). In addition, humans are likely to engage in conversations with more ***diverse*** language patterns, describing the same goal with different expressions and styles. Therefore, we ask the following question:

How do the informativeness and diversity of natural language used during RL training affect an agent's ability to learn tasks?

We take a popular offline RL model - decision transformer (DT) (Chen et al., 2021) - as a backbone architecture and conduct a comprehensive study to examine how informativeness and diversity of language use may impact agents' learning ability. To control informativeness, we leverage expert agents' actions as a reference to generate *hindsight* reflection and *foresight* guidance, using hand-crafted language templates. To increase diversity, we construct a GPT-augmented language pool, where GPT-4 (OpenAI, 2024) is used to augment hand-crafted templates into much more natural and richer expressions. We further extended DT into a multi-modal Language-Teachable DT

*Equal contribution.

[1] Source code available at https://github.com/sled-group/Teachable_RL.

(LTDT) and demonstrated that LTDT agents that are trained with diverse and informative language significantly outperform the counterpart agents that are trained either with simple language alone or with no language inputs. Notably, we found that even with just one language template, combining hindsight and foresight feedback together improves agents' performance by an average of 9.86 points (from 37.95% to 47.81%) on four popular offline RL benchmarks compared to agents trained without language. When more language diversity is incorporated into training, an additional 10.14 points (from 47.81% to 57.95%) are obtained.

The contributions of this paper can be summarized as follows:

- We investigate in detail, for the first time, how language informativeness and diversity affect offline RL agents in task learning, and demonstrate their important roles in improving agents' performance, adaptability, and robustness.
- We show that training agents with informative and diverse instructions can intrinsically improve the agent's understanding of the task and lead to better performance.
- We propose a simple framework to generate both hindsight and foresight language feedback and enrich language variation without any human annotators.

2 Related Work

Offline Reinforcement Learning Offline reinforcement learning (RL) has become a focal point of research due to its ability to utilize pre-existing datasets for training agents without real-time interactions. Several algorithms address the unique challenges of offline RL, such as mitigating extrapolation errors and ensuring robust policy evaluation. A survey by Prudencio et al. (2023) outlines the field's taxonomy and open problems. Benchmarking efforts by Fujimoto et al. (2019) assess various batch deep RL algorithms. Key approaches include Conservative Q-Learning (CQL) (Kumar et al., 2020), Implicit Q-Learning (IQL) (Kostrikov et al., 2021), and the Decision Transformer (DT) (Chen et al., 2021), which treats RL as a sequence modeling problem (Janner et al., 2021). Recent work also explores generalization across tasks (Lee et al., 2022; Reed et al., 2022; Schubert et al., 2023), the use of exploratory data (Yarats et al., 2022), and integrating large language models (LLMs) (Mirchandani et al., 2023). Efficient online RL leveraging offline data is also a focus (Ball et al., 2023; Modhe et al., 2023). Our research builds on the Decision Transformer (DT) by integrating language feedback, creating the Language-Teachable Decision Transformer (LTDT). This novel approach incorporates rich, human-like language instructions, improving agent learning through enhanced informativeness and diversity of language inputs.

Language in Reinforcement Learning The intersection of natural language and RL offers new ways to develop intuitive and effective learning paradigms for embodied agents. Initial works utilized language for feedback and task instructions (She and Chai, 2017; Nguyen et al., 2017; Shridhar et al., 2020). Recent studies have explored various methods for incorporating language feedback in RL, such as the LTC paradigm (Wang et al., 2023), lifelong robot learning with human-assisted language planners (Parakh et al., 2023), and frameworks for rich information requests (Dai et al., 2020; Tseng et al., 2021; Nguyen et al., 2022). Language for corrections (Sharma et al., 2022; Liu et al., 2023) and as reward signals (Xie et al., 2023; Goyal et al., 2019; Yu et al., 2023) has shown to enhance agent performance. Vision-language joint training approaches, like CLIP (Radford et al., 2021), BLIP-2 (Li et al., 2023), and InstructBLIP (Dai et al., 2023), demonstrate the potential of combining visual and language modalities for RL tasks (Ma et al., 2023; Nguyen et al., 2019; Khandelwal et al., 2022). Further, multimodal prompts for robotic manipulation (Jiang et al., 2023; Fan et al., 2022) and LLMs for planning in robotics (Ahn et al., 2022; Huang et al., 2022; Singh et al., 2023; Yao et al., 2022; Dai et al., 2024b) highlight the evolving role of language in RL. Other works, like (Mehta et al., 2023), focus on generating problem-specific language feedback templates. In contrast, our work focuses on the informativeness and diversity of language instructions, two problem-agnostic yet easy-to-implement properties. By using both hindsight and foresight language templates and enhancing diversity through GPT-4, we demonstrate notable improvements in agent performance and generalizability, showcasing the impact of complex language inputs in offline RL training.

3 Problem Setting

In this section, we outline the problem setting by defining the offline reinforcement learning problem

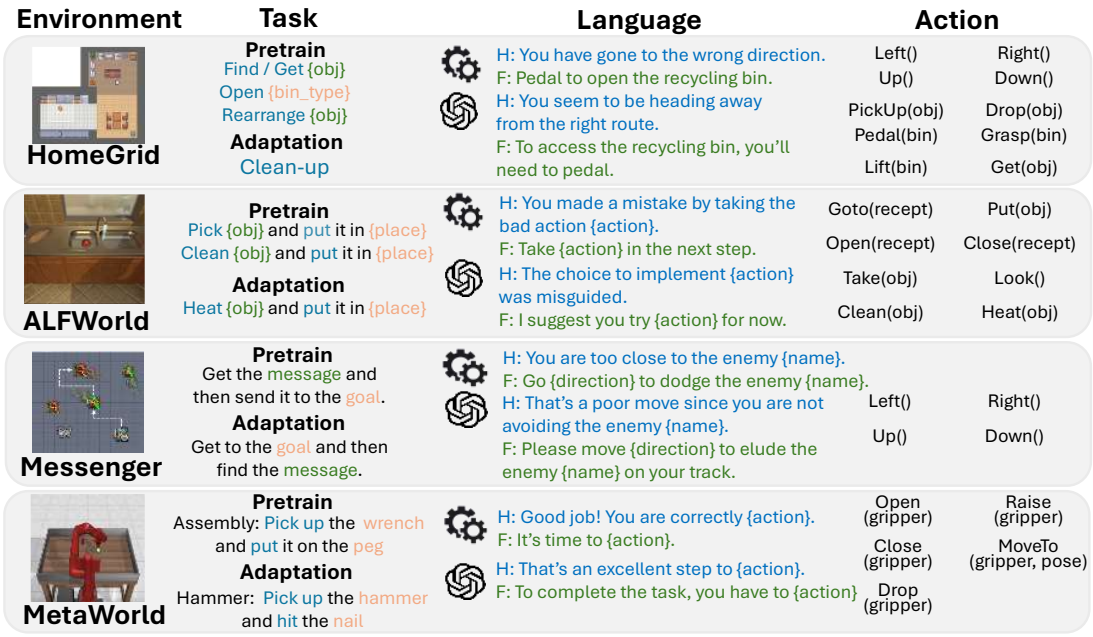

Figure 1: An overview of four environments used for experiments. It shows tasks to be learned in each environment; examples of hindsight (marked H) and foresight (F) language feedback (next to the gear icon are hand-crafted templates and next to the GPT icon are GPT-4 generated feedback); as well as low-level actions in each environment.

(Sec. 3.1), and a taxonomy of language feedback (Sec. 3.2). Then we describe the instantiation of such definitions in four different RL environments we used for experiments (Sec. 3.3).

3.1 Offline Reinforcement Learning

To support a systematic study of language use, we formulate the problem in the offline reinforcement learning (RL) setting. At each time step t, the agent receives an observation o_t, a reward r_t, and a language feedback l_t for its previous action. The agent then executes an action a_t according to a policy π, which is conditioned on the entire interaction history h_t up to time t, i.e., $\pi(a_t \mid h_t)$, where $h_t = \{o_{\leq t}, r_{\leq t}, l_{\leq t}, a_{<t}\}$ represents the history of observations, rewards, language feedback, and past actions up to time t. The agent's goal is to complete the task by maximizing the expected discounted sum of rewards $\mathbb{E}[\sum_{t=1}^{T} \gamma^t r_t]$ where T is the episode length, and γ is the discount factor. In offline RL, the training trajectories are pre-collected with an expert agent (a well-trained agent or a planner-based expert with privileged information). The trained agents are evaluated interactively with the environment.

3.2 Language Feedback: Informativeness and Diversity

We aim to investigate how the ***informativeness*** and ***diversity*** of language instructions used during the training of an offline RL agent affect the agent's performance on seen tasks and adaptation to unseen tasks.

3.2.1 Informativeness

Informativeness refers to the richness of information content in language feedback. Following Cheng et al. (2023), we categorize feedback into two types: ***hindsight*** and ***foresight***. Hindsight feedback involves comments or critiques about the agent's past actions. For example, *"Excellent, you are moving towards the goal!"* encourages the agent to continue its current path, while *"You are getting too close to the enemy."* alerts the agent about a mistake. Hindsight feedback reflects on incorrect actions taken in previous steps, which can guide agents toward success by narrowing down the search space for correct actions (See Appendix E for more analysis). Conversely, foresight feedback guides potential future actions. For instance, *"You should go right to get closer to the target."* directs the agent towards the goal, and *"You should go left to avoid the enemy on the right."* helps the agent make strategic decisions to avoid threats. Language feedback is considered most informative when it includes both hindsight and foresight elements, and least informative when neither is present.

3.2.2 Diversity

Diversity in language feedback refers to the variety of ways the same information is conveyed. If feedback is provided using only one template, it

is less diverse. It becomes more diverse when the same information is expressed in many different ways. The goal is to expose the RL agent to various expressions of the same feedback to enhance its ability to generalize.

3.3 Environments

As shown in Figure 1, we conduct experiments across four environments—HomeGrid, ALFWorld, Messenger, and MetaWorld—each featuring discrete action spaces, with hand-crafted hindsight and foresight language instructions. More information and examples of languages for each environment can be found in Appendix A.

HomeGrid (Lin et al., 2023) is a multitask grid world designed to evaluate how well agents can understand and use various types of language to complete tasks. It includes five task types (FIND, GET, CLEAN UP, REARRANGE, OPEN), involving interaction with objects and trash bins with a total of 38 tasks. The agent receives a reward of 1 when the task is completed and receives a reward of 0.5 if a subgoal is completed.

ALFWorld (Shridhar et al., 2021) is a text-game environment that aligns with the embodied ALFRED benchmark (Shridhar et al., 2020) and provides simulation for household tasks. It includes six types of tasks which require the agent to navigate and interact with household objects by following language instructions. The agent gets a reward of 1 when the task is completed. We adopt the hindsight and foresight language templates from LLF-ALFWorld introduced in (Cheng et al., 2023), which adds an extra language wrapper to the original ALFWorld environment.

Messenger (Hanjie et al., 2021) is a grid world with several entities. The agent's task is to retrieve a message from one entity and deliver it to another goal entity, while avoiding enemies. At the start of each episode, the agent is provided with a manual describing the randomized roles of the entities and their movement dynamics. The agent receives a reward of 1 when the task is completed.

MetaWorld (Yu et al., 2019) is a benchmark that consists of a variety of manipulation tasks performed by a simulated Sawyer robot arm. It includes 50 types of common robot manipulation tasks. We select two of them in our experiments: ASSEMBLY and HAMMER. The agent receives a reward of 1 when completing a task.

Algorithm 1 Offline Data Collection

Initialize $\mathcal{D} \leftarrow \emptyset$
for each episode with $seed_i$ **do**
  Initialize $\mathcal{D}_i \leftarrow \emptyset$
  Initialize environment env with $seed_i$.
  Append task description Td to $\mathcal{D}_i$
  Initialize the non-expert agent with a sub-optimal policy π.
  Initialize the expert agent with policy π^*.
  for each time step **do**
    $a_t \leftarrow \pi(h_t)$
    $a_t^* \leftarrow \pi^*(h_t)$
    $r_t, s_t, l_t^{hind}, l_t^{fore} \leftarrow \text{env}(a_t, a_t^* | h_t)$.
    if Use GPT-augmented Pool **then**
      $l_t^{hind} = \text{GPT-augmented}(l_t^{hind})$
      $l_t^{fore} = \text{GPT-augmented}(l_t^{fore})$
    end if
    $l_t \leftarrow \begin{cases} l_t^{\text{hind}} + l_t^{\text{fore}} & \text{if H + F} \\ l_t^{\text{hind}} & \text{if only H} \\ l_t^{\text{fore}} & \text{if only F} \\ \text{<empty>} & \text{if No Lang} \end{cases}$
    Append (r_t, s_t, a_t, l_t) to $\mathcal{D}_i$
  end for
  Aggregate Datasets $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$
end for

4 Data Generation

To train an agent that can understand language feedback in an offline reinforcement learning manner, we construct an offline dataset $\mathcal{D}$ consisting of two parts:

- Agent trajectory consisting of task description Td and the tuples $(\hat{R}_t, s_t, a_t)$, where $\hat{R}_t$ represents the reward, s_t is the state, and a_t is the action.
- language feedback l_t conveying hindsight and foresight information at each time step.

Algorithm 1 outlines the data generation process, and we explain the algorithm in detail in the following sections.

4.1 Trajectory Generation

To improve model generalization and avoid overfitting, it is essential to train on diverse, sub-optimal trajectories rather than relying solely on optimal ones generated by an expert agent (Kumar et al., 2020; Chen et al., 2021). We achieve this by introducing perturbations to an expert planner (see Appendix B), allowing the non-expert agent to produce sub-optimal trajectories. This promotes broader exploration of the state-action space, enhancing the model's ability to generalize to unseen scenarios (Kumar et al., 2020; Chen et al., 2021).

During data collection, we begin by appending the task description Td to the trajectory sequence and initializing the environment with a fixed seed.

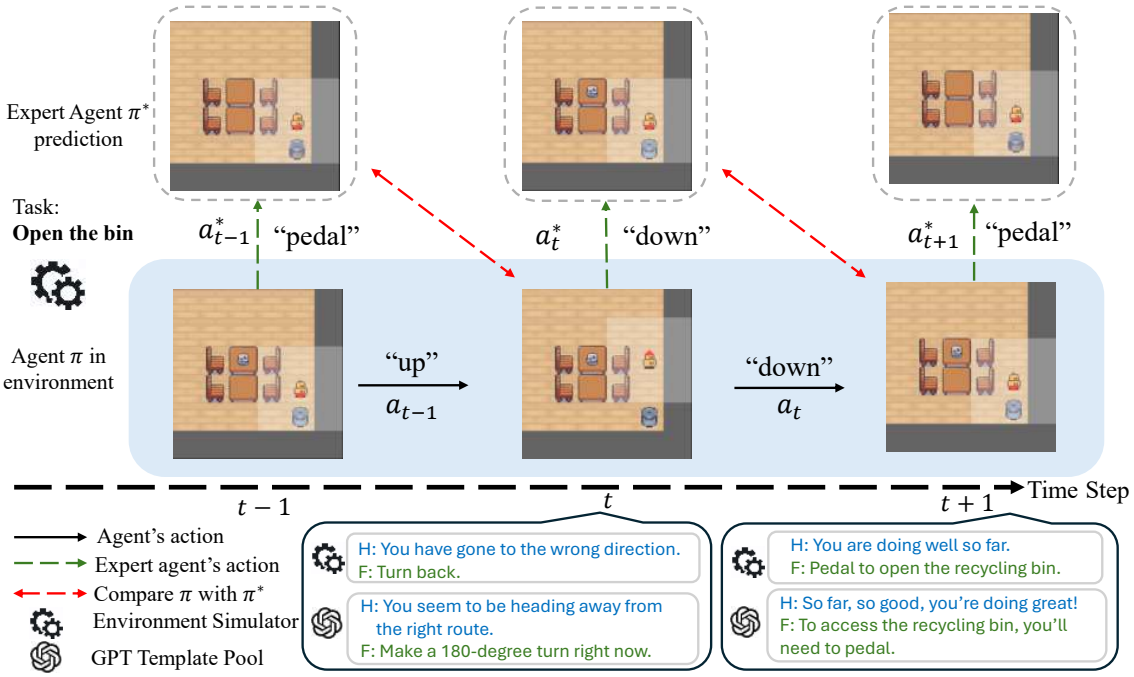

Figure 2: A demonstration of hindsight and foresight language feedback generation. In our framework, the agent π executes the trajectory, while the expert agent π^*, with access to privileged ground truth knowledge, is used solely to provide information for generating language feedback to π. At time step t, hindsight language is generated by comparing the agent's action a_{t-1} with the expert agent's action a^*_{t-1}, whereas foresight language is generated by referring to the expert agent's action a^*_t to guide the agent on the next step. To increase the diversity of language feedback, we construct a pool of language templates comprising GPT-augmented languages, and sample candidate instructions as online language feedback.

A non-expert agent, using a sub-optimal policy π derived from the expert agent's optimal policy π^*, interacts with the environment. At each time step, the environment state o_t, reward $\hat{R}_t$, and the non-expert agent's action a_t are recorded to form the trajectory sequence: $(Td, \hat{R}_1, s_1, a_1, \ldots, \hat{R}_t, s_t, a_t)$.

4.2 Language Feedback Generation

For the second part of the dataset $\mathcal{D}$, we collect the language feedback along the non-expert agent's trajectory. As shown in Figure 2, we follow a structured process to generate diverse and informative language feedback. For the state at time step t, the expert agent π^* proposes an expert action a^*_t (e.g. "down") at this state, which is further transformed into a foresight template l^{fore}_t (e.g. "Turn back.") by the environment simulator, guiding the agent on what should be done at this state. After the non-expert agent π steps the environment (into time step $t+1$) with its generated action a_t (e.g. "down"), the environment simulator generates a hindsight template l^{hind}_{t+1} (e.g. "You are doing well so far.") based on the comparison between agent action a_t and expert agent action a^*_t at the last time step t, reflecting on whether the agent is on the right track.

For each foresight/hindsight template, we use GPT-4 to augment it into more natural and varied expressions. (e.g. We can augment "You are doing well so far." into *"Up until now, you're doing wonderfully."* or *"So far, so good, you're doing great!"*.) We compile all the rewritten sentences into a set called the ***GPT-augmented language pool***. At each step of the non-expert agent, we randomly select one candidate from the pool as the language instruction. This process ensures the feedback provided to the agent has high level of diversity and enriches the learning experience.

The level of informativeness and diversity of the language feedback depends on the inclusion of hindsight and foresight (e.g. concatenated when both are required) and the use of GPT-augmented language pool. The language feedback at each time step will finally get concatenated with the trajectory sequence into $(Td, \hat{R}_1, s_1, a_1, l_1, \ldots \hat{R}_t, s_t, a_t, l_t)$. Algorithm 1 summarizes the data collection process.

5 Model

Architecture. We extend the Decision Transformer (DT) architecture (Chen et al., 2021) to create the Language-Teachable Decision Transformer (LTDT) by augmenting the input to include language feedback. This architecture is a decoder-only transformer, similar to GPT-2 (Rad-

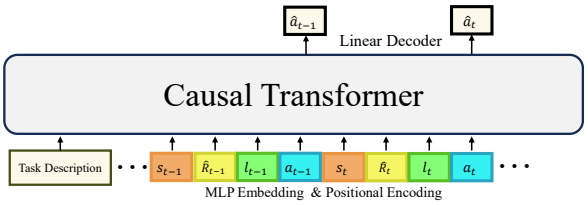

Figure 3: Language-Teachable Decision Transformer.

ford et al., 2019), and models a trajectory sequence $(Td, \hat{R}_1, s_1, a_1, l_1, \ldots, \hat{R}_t, s_t, a_t, l_t)$, with the language feedback input appended at each step and a task description (TD) input prefixed at the beginning of the sequence. Like the original DT, the embeddings of these inputs are passed through the Causal Transformer, which encodes positional information to maintain sequence order. The transformer's output is used to predict the next action in the sequence, conditioned on the state, return-to-go, action, and language feedback in the last K time steps, with the task description as the prefix ($4K + 1$ tokens in total), as shown in Figure 3.

Training. Similar to the original DT training, given an offline dataset of trajectory sequences, we sample a sub-sequence of length K (with $4K + 1$ tokens), and the prediction head is trained to predict discrete actions with the cross-entropy loss or continuous actions with the MSE loss. More training details can be found in Appendix G.

Language Embeddings. We use language embeddings from a frozen Sentence-BERT model (Reimers and Gurevych, 2019) in all environments. We find Sentence-BERT more sensitive to language feedback changes, capturing nuanced semantic differences better.

6 Experiment

In this section, we design experiments to answer the following two research questions (RQs):

- **RQ 1**: How do the ***informativeness*** and ***diversity*** of language affect agents' performance on seen tasks?
- **RQ 2**: How does the ***informativeness*** of the language feedback affect pre-trained agents' adaptability on ***unseen*** tasks?

For RQ1, we control agents trained with hindsight information, foresight information, or both to investigate the function of informativeness. We compare agents trained with language from both hand-crafted templates and the GPT-augmented language pool to examine the function of language diversity.

For RQ2, agents are taught in languages from the GPT-augmented language pool and tested on unseen tasks after fine-tuning with few-shot samples.

6.1 Experimental Setup

Setup for RQ 1. We compare performance on seen tasks between agents trained with varying levels of language informativeness and diversity: 1) the `No Language` agent is trained without any language instructions; 2) the `Template Foresight` agent is trained with hand-crafted foresight language templates; 3) the `Template Hindsight` agent is trained with hand-crafted hindsight language templates; 4) the `Template Hindsight + Foresight` agent is trained with hand-crafted foresight and hindsight language templates; and 5) the `GPT-augmented Hindsight + Foresight` agent is trained with hindsight and foresight languages from the GPT-augmented language pool. We train on 100, 1,000, 20,000, and 10,000 trajectories for HomeGrid, ALFWorld, Messenger, and Meta-World environments, respectively. Evaluation is performed over 5 runs, with 100 random seeds for each run.

Setup for RQ 2. We pre-train different agents on seen tasks and then compare adaptability (how well an agent performs after few-shot learning) on unseen tasks: 1) the `No Language pre-trained` agent is pre-trained without any language instructions; 2) the `GPT-augmented hindsight pre-trained` agent is pre-trained with hindsight language from the GPT-augmented language pool; 3) the `GPT-augmented foresight pre-trained` agent is pre-trained with foresight language from the GPT-augmented language pool; 4) the `GPT-augmented hindsight + foresight pre-trained` agent is pre-trained with both hindsight and foresight language from the GPT-augmented language pool. During the few-shot adaptation stage, we choose to fine-tune the pre-trained agents with both hindsight + foresight language from the GPT-augmented language pool for all settings, since this mimics a real-world few-shot learning scenario, where humans likely provide diverse feedback, including both hindsight and foresight, to guide the agent in new tasks. We pretrain on 6,432, 1,000, 20,000, and 10,000 trajectories for HomeGrid, ALFWorld, Messenger, and Meta-World, respectively. For all environments, we adapt on 5, 10, and 20 trajectories to 1 new task. Evaluation is performed over 5 runs, with 100 seeds per run. Further details on task setup of RQ 1 and RQ 2 can be found in Appendix C. Additional re-

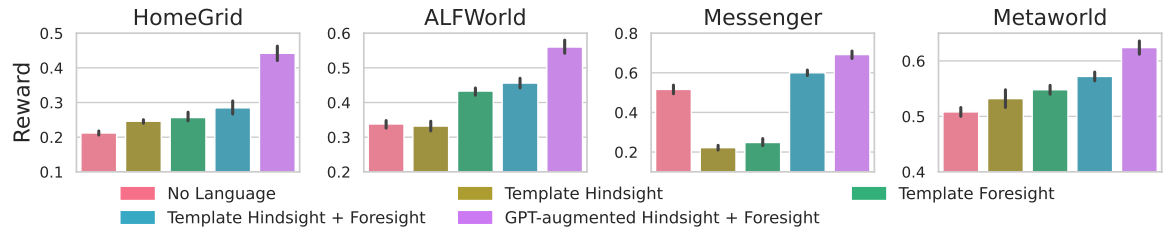

Figure 4: Comparison of agent performance in four environments (averaged across 100 seeds in each environment) under varying levels of language feedback informativeness and diversity. Agents trained with more informative language feedback exhibit progressively higher performance. Furthermore, given the same informativeness (Hindsight + Foresight), increasing diversity with the GPT-augmented language pool leads to the highest performance.

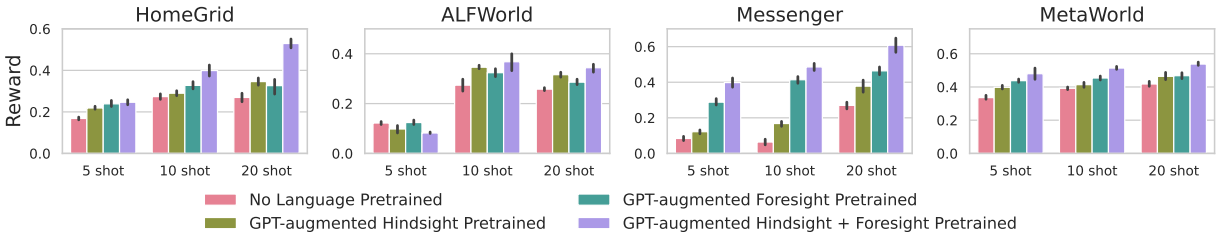

Figure 5: Comparison of agent performance on **unseen tasks** in four environments (averaged across 100 seeds in each environment) under varying language informativeness in agent pre-training. Agent trained with more informative language adapts to new tasks faster and better.

sults when training and adapting on same types of language can be found in Appendix D.

Evaluation. At inference time, an agent is given a short task description before it starts to act, and language feedback along its execution. The language feedback should ideally come from real humans, who provide feedback varying in informativeness, diversity, and frequency (how often feedback is provided). However, recruiting and moderating real humans to generate online feedback is expensive and difficult to scale. Therefore, we employ GPT-4 to provide online language feedback to mimic real humans. Specifically, at each time step, we provide all necessary context information to GPT-4 in its prompt and let it decide "whether to speak" (frequency), "what to speak" (informativeness), and "how to speak" (diversity). The context information, in this case, consists of the ground-truth environment states, action/state history, and template-based hindsight and foresight short text description generated by comparing the actions of the expert agent and the trained agent. GPT-4 then has the freedom to rephrase, combine, shorten, and discard such context information to utter diverse, coherent, and natural language feedback, mimicking a real human. See Appendix H for an example of such GPT-generated online feedback.

Metric. We use the reward value as our main metric. Agents receive a reward of 1 upon task completion for all environments and receive additional rewards for achieving specific sub-goals for the HomeGrid and ALFWorld environments.

6.2 Experimental Results

Results for RQ 1. As we can see in Figure 4, agents trained with both diverse and informative language feedback (`GPT-augmented Hindsight + Foresight`) consistently achieve the highest performance across all environments. The varied and paraphrased instructions generated from GPT provide a richer set of linguistic inputs, enabling the agents to develop a more robust language understanding for task execution during evaluation.

When examining the impact of informativeness, we observe that agents trained with both hindsight and foresight information (Template Hindsight + Foresight) consistently achieve higher performance across all environments compared to those trained with only hindsight or foresight information. This indicates that integrating both types of feedback enhances the informativeness of the language, enabling the agents to develop a more comprehensive understanding and leading to better decision-making and overall performance. The only exception is in the Messenger environment, where the no-language agent exhibits a surprisingly strong performance. However, upon further investigation of this exception, we find that if the hindsight-only or foresight-only feedback is from the GPT-augmented pool, the agent can still outperform the `No Language` agent (refer to Appendix F).

In terms of diversity, the results show that agents trained with diverse language feedback, as indicated by the 'GPT-augmented' bars, consistently outperform those trained with less varied language

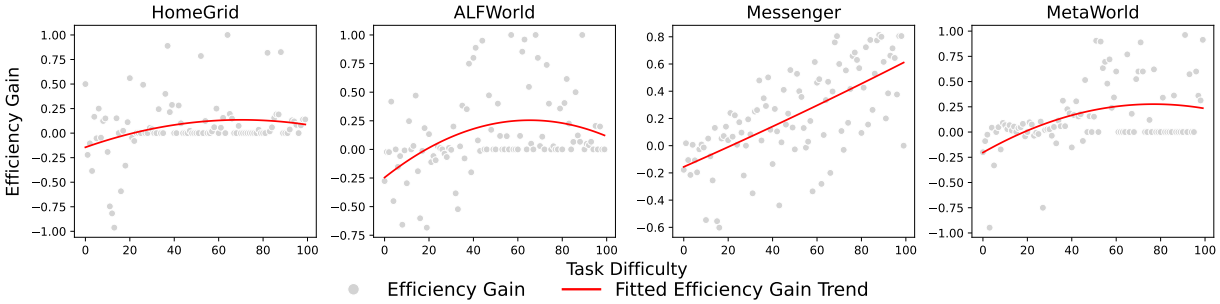

Figure 6: Efficiency gain vs. task difficulty. We fit the scatter plots with a second-degree polynomial to visualize the overall trend. As task difficulty increases, the general trend of the efficiency gain is to rise initially and then decline, suggesting: (1) for tasks that are too easy or too hard, language feedback does not improve efficiency; (2) language feedback is most helpful in increasing efficiency for moderate tasks.

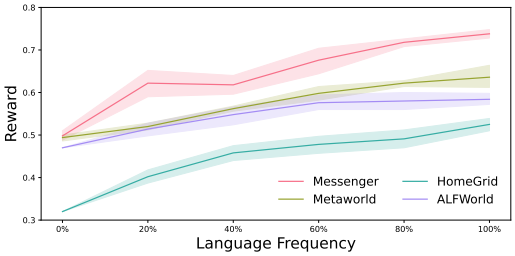

Figure 7: Performance vs. language frequency. Agents perform better with more frequent language feedback across four environments.

input. The rich set of augmented instructions generated by GPT helps agents develop a more flexible and nuanced understanding of task instructions, which translates to better performance during evaluation. This highlights the critical role of linguistic diversity in enhancing the robustness and adaptability of the agents' language comprehension, ultimately leading to improved task execution across different environments.

Results for RQ 2. The results in Figure 5 reveal that agents pre-trained with more informative language can adapt to unseen tasks *faster* and *better*. "Adapting faster" is evident by the fact that agents pre-trained with GPT-augmented Hindsight + Foresight language in 5 or 10 shots can already achieve a similar performance 20-shot performance of agents trained with less informative language. "Adapting better" is evident by the fact that, at a given number of shots available for adaptation, the agent trained with the most informative language performs the best compared to its less informatively-pretrained counterparts. These results indicate that agents pre-trained with more informative language can adapt and generalize to new tasks faster and better.

6.3 Ablation Study

Efficiency Gain vs. Task Difficulty. Can language feedback help the agent to achieve more different tasks? To answer this question, we define **efficiency gain** as the difference in efficiency between an agent trained with informative and diverse GPT languages, and an agent trained without any languages. Efficiency is measured by a path-weighted reward, as introduced in ALFRED (Shridhar et al., 2020). This reward, r_p, is calculated as $r_p = r \times \frac{L^*}{\max(L,L^*)}$, where r is the total reward, L is the agent's trajectory length, and L^* is the expert agent's trajectory length. Higher r_p indicates successful task completion with fewer steps. We define **task difficulty** for each configuration by calculating the average success rates of agents trained without language feedback, ranking these from lowest to highest. Configurations with lower success rates are considered more difficult, indicating greater challenges for agents learning from these configurations without language assistance. As shown in Figure 6, the efficiency gain generally rises with increasing learning difficulty, then declines. This suggests that: (1) for tasks that are too easy or too hard, language feedback does not improve efficiency; (2) language feedback is most helpful in increasing efficiency for moderate tasks.

Performance vs. Language Frequency. In the main experiments, we utilize an online GPT model to determine whether to provide language feedback at each time step. However, it is important to explore how varying the frequency of language feedback influences agent performance. To investigate this, we control the feedback frequency by sampling according to pre-defined probabilities (e.g., 20%, 40%). The language feedback is extracted from the GPT-augmented language pool; if no language is sampled, an empty string is provided to the agent. The evaluation is conducted on agents trained with both hindsight and foresight feedback derived from the GPT-augmented language pool. As illustrated in Figure 7, agents' performance improves steadily across all environments with more frequent language feedback during evaluation. This

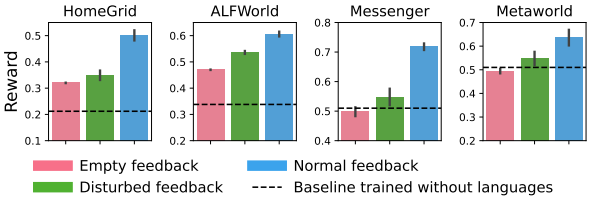

Figure 8: We investigate two special evaluation settings: (1) no language feedback is provided during evaluation and (2) disturbed language feedback is given at every step. Results show that agents trained with the GPT-augmented language still outperform the no-language agent (the black dotted line) in the disturbed setting, and also achieve better performance in some environments while no language is given.

finding suggests that agents trained with informative and diverse language feedback can continually absorb and leverage new information when additional feedback is provided, leading to enhanced performance.

Performance under Corrupted Language. This ablation aims to evaluate how agents perform when provided with incorrect instructions. We assess the performance of an agent trained with GPT-4-augmented informative and diverse language under two conditions: (1) Empty Feedback: the absence of language feedback during testing, and (2) Disturbed Feedback: the provision of disturbed language at each step. The disturbed language consists of redundant, irrelevant, or misleading information (e.g., incorrect actions or objects) and is generated using GPT-augmented templates with disrupted content. The results in Figure 8 reveal two interesting findings: (1) When tested without any language feedback, the agent trained with informative and diverse language performs comparably or even exceeds the performance of the agent trained without any language (represented by the black dotted line). This indicates that the agent develops a robust intrinsic understanding of the task, demonstrating that it does not overly rely on language feedback; (2) When exposed to disturbed feedback, the agent trained with informative and diverse language maintains performance levels comparable to the no-language agent. This showcases the agent's ability to withstand misleading information, a critical trait for real-world applications where human feedback may be unreliable.

7 Conclusion

In this paper, we investigate how the informativeness and diversity of language feedback affect embodied agents. We introduce the Language-Teachable Decision Transformer (LTDT), which makes decisions based on human language feedback. To facilitate the training of LTDT agents, we propose an easy-to-use pipeline for collecting offline hindsight and foresight GPT templates. We compare the performance of agents by varying the informativeness and diversity of the training languages across four reinforcement learning environments and evaluate the agents' ability to understand real-world human language using online GPT as a proxy. Our results demonstrate that training with more informative and diverse language feedback significantly enhances agent performance and enables fast adaptation to unseen tasks.

Limitations

Our study has several limitations. First, the investigated environments are primarily game-based and do not test the agents' ability to incorporate real-life visual inputs. Future work will focus on evaluating agents in more realistic and complex environments that involve real-world visual inputs and challenges. Second, while GPT language outputs can produce diverse and contextually relevant language, they may not fully cover all human language styles and nuances. Specifically, GPT models might miss certain idioms, dialects, or culturally specific references that are prevalent in human communication. Future work will aim to incorporate a broader spectrum of language variations and test agents in scenarios involving more diverse linguistic inputs.

Ethical Impacts

Our study, conducted entirely within simulated environments, does not present immediate ethical concerns. The teachable nature of our Language-Teachable Decision Transformer (LTDT) method is designed to make AI agents more controllable and better aligned with human values, promoting safer and more ethical interactions. By enhancing agent performance through informative and diverse language instructions, we aim to foster AI systems that are more transparent and responsive to human guidance, addressing ethical considerations in the deployment of artificial intelligence. As AI becomes more mainstream, these considerations are increasingly pertinent, and our work strives to advance AI technology responsibly.

Acknowledgements

This work was supported by NSF IIS-1949634 and has benefited from the Microsoft Accelerate Foundation Models Research (AFMR) grant program.

We would like to thank the anonymous reviewers for their valuable comments and suggestions.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. 2022. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*.

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. 2023. Efficient online reinforcement learning with offline data. *arXiv preprint arXiv:2302.02948*.

Joyce Chai, Maya Cakmak, and Candy Sidner. 2019. Teaching robots new tasks through natural interaction. In K. A. Cluck and J. E. Laird, editors, *Interactive Task Learning: Agents, Robots, and Humans Acquiring New Tasks through Natural Interactions*. MIT Press.

Joyce Chai, Qiaozi Gao, Lanbo She, Shaohua Yang, Sari Saba-Sadiya, and Guangyue Xu. 2018. Language to action: Towards interactive task learning with physical agents. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 2–9. ijcai.org.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097.

Ching-An Cheng, Andrey Kolobov, Dipendra Misra, Allen Nie, and Adith Swaminathan. 2023. Llf-bench: Benchmark for interactive learning from language feedback. *arXiv preprint arXiv:2312.06853*.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. 2023. Instructblip: Towards general-purpose vision-language models with instruction tuning. *Preprint*, arXiv:2305.06500.

Yinpei Dai, Jayjun Lee, Nima Fazeli, and Joyce Chai. 2024a. Racer: Rich language-guided failure recovery policies for imitation learning. *arXiv preprint arXiv:2409.14674*.

Yinpei Dai, Hangyu Li, Chengguang Tang, Yongbin Li, Jian Sun, and Xiaodan Zhu. 2020. Learning low-resource end-to-end goal-oriented dialog for fast and reliable system deployment. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 609–618.

Yinpei Dai, Run Peng, Sikai Li, and Joyce Chai. 2024b. Think, act, and ask: Open-world interactive personalized robot navigation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3296–3303. IEEE.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. 2022. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35:18343–18362.

Scott Fujimoto, Edoardo Conti, Mohammad Ghavamzadeh, and Joelle Pineau. 2019. Benchmarking batch deep reinforcement learning algorithms. *arXiv preprint arXiv:1910.01708*.

Prasoon Goyal, Scott Niekum, and Raymond J Mooney. 2019. Using natural language for reward shaping in reinforcement learning. *arXiv preprint arXiv:1903.02020*.

Austin W. Hanjie, Victor Zhong, and Karthik Narasimhan. 2021. Grounding language to entities and dynamics for generalization in reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 4051–4062. PMLR.

Peter E Hart, Nils J Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107.

Wanwei He, Yinpei Dai, Binyuan Hui, Min Yang, Zheng Cao, Jianbo Dong, Fei Huang, Luo Si, and Yongbin Li. 2022a. Space-2: Tree-structured semi-supervised contrastive pre-training for task-oriented dialog understanding. *arXiv preprint arXiv:2209.06638*.

Wanwei He, Yinpei Dai, Min Yang, Jian Sun, Fei Huang, Luo Si, and Yongbin Li. 2022b. Unified dialog model pre-training for task-oriented dialog understanding and generation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 187–200.

Wanwei He, Yinpei Dai, Yinhe Zheng, Yuchuan Wu, Zheng Cao, Dermot Liu, Peng Jiang, Min Yang, Fei Huang, Luo Si, et al. 2022c. Galaxy: A generative pre-trained model for task-oriented dialog with semi-supervised learning and explicit policy injection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 10749–10757.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. 2022. Inner monologue: Embodied reasoning through planning with language models. In *arXiv preprint arXiv:2207.05608*.

Michael Janner, Qiyang Li, and Sergey Levine. 2021. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. 2023. Vima: General robot manipulation with multimodal prompts. In *Fortieth International Conference on Machine Learning*.

Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. 2022. Simple but effective: Clip embeddings for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14829–14838.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2021. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191.

Kuang-Huei Lee, Ofir Nachum, Mengjiao Sherry Yang, Lisa Lee, Daniel Freeman, Sergio Guadarrama, Ian Fischer, Winnie Xu, Eric Jang, Henryk Michalewski, et al. 2022. Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35:27921–27936.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *Preprint*, arXiv:2301.12597.

Jessy Lin, Yuqing Du, Olivia Watkins, Danijar Hafner, Pieter Abbeel, Dan Klein, and Anca Dragan. 2023. Learning to model the world with language. *arXiv preprint arXiv:2308.01399*.

Zeyi Liu, Arpit Bahety, and Shuran Song. 2023. Reflect: Summarizing robot experiences for failure explanation and correction. *arXiv preprint arXiv:2306.15724*.

Yecheng Jason Ma, William Liang, Vaidehi Som, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. 2023. Liv: Language-image representations and rewards for robotic control. *Preprint*, arXiv:2306.00958.

Sabrina McCallum, Max Taylor-Davies, Stefano Albrecht, and Alessandro Suglia. 2023. Is feedback all you need? leveraging natural language feedback in goal-conditioned rl. In *NeurIPS 2023 Workshop on Goal-Conditioned Reinforcement Learning*.

Nikhil Mehta, Milagro Teruel, Patricio Figueroa Sanz, Xin Deng, Ahmed Hassan Awadallah, and Julia Kiseleva. 2023. Improving grounded language understanding in a collaborative environment by interacting with agents through help feedback. *arXiv preprint arXiv:2304.10750*.

Suvir Mirchandani, Fei Xia, Pete Florence, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, Andy Zeng, et al. 2023. Large language models as general pattern machines. In *7th Annual Conference on Robot Learning*.

Nirbhay Modhe, Qiaozi Gao, Ashwin Kalyan, Dhruv Batra, Govind Thattai, and Gaurav Sukhatme. 2023. Exploiting generalization in offline reinforcement learning via unseen state augmentations. *arXiv preprint arXiv:2308.03882*.

Khanh Nguyen, Hal Daumé III, and Jordan Boyd-Graber. 2017. Reinforcement learning for bandit neural machine translation with simulated human feedback. *arXiv preprint arXiv:1707.07402*.

Khanh Nguyen, Debadeepta Dey, Chris Brockett, and Bill Dolan. 2019. Vision-based navigation with language-based assistance via imitation learning with indirect intervention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12527–12537.

Khanh X Nguyen, Yonatan Bisk, and Hal Daumé Iii. 2022. A framework for learning to request rich and contextually useful information from humans. In *International Conference on Machine Learning*, pages 16553–16568. PMLR.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Meenal Parakh, Alisha Fong, Anthony Simeonov, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. 2023. Lifelong robot learning with human assisted language planners. In *CoRL 2023 Workshop on Learning Effective Abstractions for Planning (LEAP)*.

Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. 2023. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. 2022. A generalist agent. *Transactions on Machine Learning Research*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Ingmar Schubert, Jingwei Zhang, Jake Bruce, Sarah Bechtle, Emilio Parisotto, Martin Riedmiller, Jost Tobias Springenberg, Arunkumar Byravan, Leonard Hasenclever, and Nicolas Heess. 2023. A generalist dynamics model for control. *arXiv preprint arXiv:2305.10912*.

Pratyusha Sharma, Balakumar Sundaralingam, Valts Blukis, Chris Paxton, Tucker Hermans, Antonio Torralba, Jacob Andreas, and Dieter Fox. 2022. Correcting robot plans with natural language feedback. *Preprint*, arXiv:2204.05186.

Lanbo She and Joyce Chai. 2017. Interactive learning of grounded verb semantics towards human-robot communication. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1634–1644. Association for Computational Linguistics.

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. *Preprint*, arXiv:1912.01734.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2023. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE.

Bo-Hsiang Tseng, Yinpei Dai, Florian Kreyssig, and Bill Byrne. 2021. Transferable dialogue systems and user simulators. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 152–166, Online. Association for Computational Linguistics.

Kuan Wang, Yadong Lu, Michael Santacroce, Yeyun Gong, Chao Zhang, and Yelong Shen. 2023. Adapting llm agents through communication. *Preprint*, arXiv:2310.01444.

Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. 2023. Text2reward: Automated dense reward function generation for reinforcement learning. *arXiv preprint arXiv:2309.11489*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, Pieter Abbeel, Alessandro Lazaric, and Lerrel Pinto. 2022. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. In *ICLR 2022 Workshop on Generalizable Policy Learning in Physical World*.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. 2019. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*.

Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, Brian Ichter, Ted Xiao, Peng Xu, Andy Zeng, Tingnan Zhang, Nicolas Heess, Dorsa Sadigh, Jie Tan, Yuval Tassa, and Fei Xia. 2023. Language to rewards for robotic skill synthesis. *Arxiv preprint arXiv:2306.08647*.

Yichi Zhang and Joyce Chai. 2021. Hierarchical task learning from language instructions with unified transformers and self-monitoring. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4202–4213, Online. Association for Computational Linguistics.

Yichi Zhang, Jianing Yang, Jiayi Pan, Shane Storks, Nikhil Devraj, Ziqiao Ma, Keunwoo Yu, Yuwei Bao, and Joyce Chai. 2022. DANLI: Deliberative agent for following natural language instructions. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1280–1298, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yichi Zhang, Jianing Yang, Keunwoo Yu, Yinpei Dai, Shane Storks, Yuwei Bao, Jiayi Pan, Nikhil Devraj, Ziqiao Ma, and Joyce Chai. 2023. Seagull: An embodied agent for instruction following through situated dialog. In *Alexa Prize SimBot Challenge Proceedings*.

A Environments and Language Feedback

A.1 Environments Overview

The Appendix Table 1 lists the information that is inherently available within the environment. All models, regardless of whether they are trained with language input or not, will have access to this environmental information.

Env	Image Observation	Instruction Manual	Text State Description
HomeGrid	Yes	No	No
AlfWorld	No	No	Yes
Messenger	No	Yes	No
MetaWorld	No	No	No

Table 1: Information provided by each environment.

A.2 Language Feedback for Different Environments

For each environment, we design multiple templates conveying different meanings, and then applied GPT-4 to augment the languages into a GPT-augmented language pool. The number of templates and the corresponding GPT-augmented sentences for each template are shown in Appendix Table 2.

Env	# Hind Templates	# Fore Templates	# AUG
HomeGrid	20	9	70
AlfWorld	4	4	200
Messenger	4	4	80
MetaWorld	2	6	180

Table 2: Number of templates and augmented sentences for each environment, where '# Hind Templates' refers to the number of hindsight templates, '# Fore Templates' refers to the number of foresight templates, and '# AUG' refers to the number of GPT-augmented sentences per template.

A.2.1 HomeGrid

HomeGrid is a multitask grid world designed to evaluate how well agents can understand and use various types of language to complete tasks. Agents will receive both task specifications and language hints, providing prior knowledge about world dynamics, information about world states, or corrections to assist the agents. We adopt the language hints in HomeGrid as foresight and further extend the environment to provide hindsight that provides comments on agents' past performance. Agents are expected to ground both hindsight and foresight to the environment to achieve higher performance. It includes five task types involving interaction with objects and bins (find, get, clean up, rearrange, open), with a total of 38 tasks. Object locations, bin locations, and bin dynamics are randomized. The agent receives a reward of 1 when the task is completed, and receives a reward of 0.5 if a subgoal exists (e.g., get the object in the clean-up task) and gets completed. Each template language is augmented to 70 sentences in the GPT template pool. Examples of hindsight and foresight languages are as follows:

- Hindsight Examples:
  - Template:
    - "You have gone to the wrong direction."
    - "You are doing well so far."
  - GPT Template:
    - "You seem to be heading away from the right route."
    - "So far, so good, you are doing great!"
- Foresight Examples:
  - Template:
    - "Turn back."
    - "Pedal to open the recycling bin."
  - GPT Template:
    - "Make a 180-degree turn right now."
    - "To access the recycling bin, you'll need to pedal."

Language instructions are generated based on the comparison of agent's action and expert planer action, considering distance, relative location, and interaction between the agent and target objects.

A.2.2 ALFWorld

ALFWorld is a text-game environment that aligns with the embodied ALFRED benchmark (Shridhar et al., 2020) and provides simulation for household tasks. It includes six types of tasks where agents need to navigate and interact with household objects through text actions. The location of the task objects is randomly located among 50 locations in each episode, making the task challenging for the agent to plan and for the subgoals. For the experiment, we adopt LLF-ALFWorld (Cheng et al., 2023), which provides an extra language wrapper for hindsight and foresight language generation over the original ALFWorld. The languages are generated based on both agents' past actions and the optimal trajectory for the current episode. Agent gets a reward of 1 when the task is completed. Each template is augmented to 200 sentences in

GPT template pool. Examples of hindsight and foresight languages are as follows:

- Hindsight Examples:
  - Template:
    - "You made a mistake by taking the bad action {action}."
    - "It was a right decision to not take the bad action {action}."
  - GPT Template:
    - "The choice to implement {action} was misguided."
    - "You made a sensible choice by not committing to the {avoid action}."
- Foresight Examples:
  - Template:
    - "You should now take the {action} action."
    - "Take {action} in the next step."
  - GPT Template:
    - "Consider taking the {action} as your next step."
    - "Moving on, consider the {action} action."

Language instructions are generated based on experts' next action and whether agent's past actions are aligned with expert past actions, considering whether agents have moved to the target position and conducted correct interaction with the objects.

A.2.3 Messenger

Messenger is a grid world with several entities. The agent's primary task is to retrieve a message from one entity and deliver it to another goal entity, all while avoiding enemies. At the start of each episode, the agent is provided with a manual describing the randomized roles of the entities and their movement dynamics. The challenge lies in the fact that the agent does not have access to the true identity of each entity and must ground the text manual to the dynamics, necessitating multi-hop reasoning. (For example, grounding the "an approaching queen is a deadly enemy" to the observations of dynamics.) (Lin et al., 2023) The agent receives a sparse reward of 1 when the task is completed. Each template language is augmented to 80 sentences in the GPT template pool. Examples of hindsight and foresight languages are as follows:

- Hindsight Examples:
  - Template:
    - "It's good that you are getting close to the {target} at {target direction} by moving {direction}!"
    - "Stepping {action direction}, yet you ran into {enemy name}. Be more cautious."
  - GPT Template:
    - "Good job on approaching the {target} to the {target direction} by moving {direction}! "
    - "Stepping {action direction} directly met {enemy name}. Needs strategic thinking."
- Foresight Examples:
  - Template:
    - "Move {optimal direction} to approach the {target name} located at the {target direction}. "
    - "Rest assured, there are no enemies around."
  - GPT Template:
    - "To get to the {target name} at {target direction}, go {optimal direction}. "
    - "Not detecting any danger, it's safe."

When generating the language instructions, we compare the agent's actions and the expert's actions, considering the locations of the target and nearest enemy, calculating the distance and generate the hindsight reflections based on some engineered rules.

A.2.4 MetaWorld

MetaWorld is a simulated benchmark that includes a variety of manipulation tasks performed using a Sawyer robot arm. It includes 50 types of robot manipulation tasks common in daily life. Since our main goal is not meta-learning, we select the "assembly" and "hammer" tasks for pretraining and adaptation in our experiments. This requires the agent to pick up the tool and aim at the specific target with high precision. To increase the challenge of the tasks, we introduce random disturbances at random steps. This requires the robot to actively recover and return to its normal trajectory whenever it deviates. The agent receives a sparse reward of 1 when completing the task. Each template language is augmented to 180 template languages in the GPT template pool. Examples of hindsight and foresight languages are shown in the following:

- Hindsight Examples:

- Template:
  - "It's excellent to raise the gripper."
  - "You are making mistakes for not opening your gripper."
- GPT Template:
  - "Good job for raising your gripper."
  - "You make a regrettable mistake since your gripper is closing."

- Foresight Examples:
  - Template:
    - "It's time to grasp the wrench now."
    - "Please raise the hammer."
  - GPT Template:
    - "Can you grab the wrench with your gripper?"
    - "I think the hammer should be raised now."

We compare the agent's actions with the expert's actions, and tell the agent's whether their decisions at the previous step matches with the expert's actions, and inform them of what an expert will do at the next step.

B Agent for Offline Data Collection and Language Feedback Generation

We use an expert agent and a non-expert agent with sub-optimal policies during the data collection. The sub-optimal policy is used for introducing some errors or perturbations in the training data, and letting the expert policy continue to recover. This helps agents learn to recover from potential failures using hindsight reflections and foresight instructions. In our experiments, we introduced 10-20% random noise in each trajectory as a sub-optimal policy. We found that this level of perturbation aids learning, but excessive disturbance (e.g., >50% per trajectory) significantly degrades performance as agents start learning suboptimal behaviors.

B.1 HomeGrid

For the HomeGrid environment, we design an expert planer to work as the expert agent. We first divide the task into several sub-tasks (i.e. divide "open the recycling bin" into 1. "navigate to the bin", 2. "open the bin"). For navigation (move to some place) sub-tasks, we implement breadth-first search to find the optimal path; for interaction sub-task (interact with object), we output the corresponding action. We implement the non-expert agent by adding "perturbation" into the expert planer. For example, we randomly reverse the next step of expert action and let the expert planner recover from the error.

B.2 ALFWorld

For the ALFWorld environment, we use a pre-built expert planer from LLF-Bench (Cheng et al., 2023) to work as both the expert agent and the agent for the data collection.

B.3 Messenger

As for the Messenger environment, we implement an expert agent using the A* algorithm (Hart et al., 1968). We define the cost by the distance to the target and the distance to the nearest enemies, and then heuristically search in the grid environment. The non-expert agent in the data collection is implemented by adding random disturbance to the expert agent.

B.4 MetaWorld

We build the expert agent on the pre-defined policy from the original MetaWorld codebase (Yu et al., 2019) and adapt the policy to random disturbance so that the expert planner can recover to a normal trajectory in any situation.

C Task Settings for RQ 1 and 2

Task Setting for RQ 1. We evaluate the agents' performance using the *same tasks* as in the training phase (but with different initialization of the agents and object layout for different episodes). Concretely, 1) in HomeGrid, we train and evaluate on multi-tasks, including FIND, GET, REARRANGE and OPEN; 2) in ALFWorld, we train and evaluate on multi-tasks including PICK&PLACE, CLEAN&PLACE and HEAT&PLACE tasks; 3) in Messenger, we train and evaluate on the task goal *"first retrieve the message and then deliver to target entity"*; and 4) in MetaWorld, we train and evaluate on the ASSEMBLY task, in which the robot arm needs to pick up the wrench and put it on the peg.
Task Setting for RQ 2. We evaluate agents' performance on *unseen tasks* by first pre-training agents on certain tasks and then adapting agents to unseen tasks with few-shot episodes. Specifically, 1) in HomeGrid, we take FIND, GET, REARRANGE, OPEN tasks for pre-training and the CLEAN-UP task for adaptation and evaluation; 2) in ALFWorld, we take PICK&PLACE and CLEAN&PLACE for pre-training and HEAT&PLACE tasks for adaptation and evaluation; 3) in Messenger, we take *"first retrieve the message and then deliver to target entity"*

as the pretraining task and *"first get to the target entity and then retrieve the message"* (where the order of the goal is reversed compared to the pretraining tasks) for adaptation and evaluation; 4) in MetaWorld, we take the ASSEMBLY task for pretraining, and the HAMMER task for adaptation and evaluation.

D Performance under aligned language type with training.

As stated in Section 6.1, we use online GPT for all evaluations in RQ 1 and 2 to mimic real-life human language environments. In this section, we align the evaluation language type (and adaptation language type in RQ 2) with each agent's corresponding training language type for further investigation (e.g. `No Language` Agent is evaluated with empty language; `Template Hindsight` Agent is evaluated with Template Hindsight). Experiments on RQ 1 and 2 are conducted on HomeGrid and Messenger respectively, with the results presented in Table 3.

HomeGrid Env on RQ 1		
Training Language	**Aligned Eval**	**Online GPT Eval**
No Lang	0.235	0.212
Template H	0.260	0.246
Template F	0.305	0.262
Template H + F	0.325	0.285
GPT-augmented H + F	0.472	0.442
Messenger Env on RQ 2 (20 Shots)		
Training Language	**Aligned Adapt & Eval**	**Online GPT Eval**
No Lang	0.323	0.270
GPT-augmented H	0.450	0.378
GPT-augmented F	0.512	0.464
GPT-augmented H + F	0.623	0.608

Table 3: Comparison of agents' performance adapted (for RQ 2) and evaluated with aligned language type in HomeGrid environment on RQ 1 and Messenger environment on RQ 2. 'Aligned (Adapt &) Eval' refers to (adaptation &) evaluation with same type of language in training and 'Online GPT Eval' refers to online GPT evaluation (results in Section 6.2). The results show that GPT-augmented Hindsight + Foresight evaluated with online GPT still outperforms other training settings even with aligned language evaluation, indicating higher language informativeness and diversity enhance intrinsic task understanding.

The results Table 3 show that: (1) aligning the informativeness and diversity levels between training, adaptation and evaluation improves the final performance for all types; (2) more importantly, even with aligned evaluation and adaptation language, **no other settings have outperformed GPT-augmented Hindsight + Foresight evaluated with online GPT**. This further demonstrates that high informativeness and diversity in training language help agents intrinsically understand tasks to achieve better performance.

E Impact of hindsight on future steps

Compared to foresight feedback, which provides instructions for the correct action in the next step, hindsight feedback reflects on incorrect actions taken in previous steps. This retrospective analysis can still guide agents toward success by narrowing down the search space for corrective actions. To demonstrate the effectiveness of hindsight feedback, we conduct a quick comparative study between the `No Language` agent and the `Template Hindsight` agent in HomeGrid. The study was designed as follows:

1. Both agents are driven to the same state using an expert policy.
2. A deliberate mistake is introduced for both agents. Three types of mistakes are designed:
   - **Navigation Mistake**: The agent moves in the opposite direction compared to the expert action.
   - **Object Pick/Drop Mistake**: The agent picks or drops an object when the expert action is to drop or pick, respectively.
   - **Bin Manipulation Mistake**: The agent chooses the wrong action among pedal/lift/grasp to open a specific trash bin.
3. We use expert actions as the ground truth (GT) actions and compare the performance of both agents over 500 runs.

The results are shown in Appendix Table 4: The

Mistake Type	**No Lang (%)**	**Template Hindsight (%)**
Navigation	37.6 ± 0.3	46.2 ± 0.2
Object Pick/Drop	37.4 ± 2.5	41.8 ± 1.6
Bin manipulation	23.5 ± 1.2	24.8 ± 0.9

Table 4: Comparison of performance between `No Language` Agent and `Template Hindsight` Agent on different Mistake Types.

results indicate that for the navigation and object pick/drop mistakes, hindsight feedback is highly beneficial. This is because identifying a wrong action usually directly implies the correct action for those mistakes (e.g., if "turn left" is wrong, "turn right" is correct; if "pick the object" is wrong, "drop the object" is correct). However, for the bin manipulation mistake, hindsight feedback is less helpful

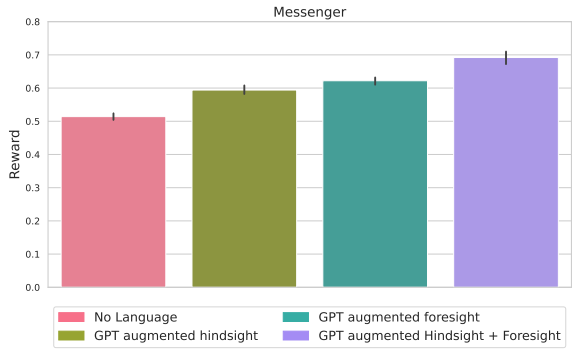

Figure 9: In the Messenger environment, when trained with more diverse foresight and hindsight languages, the agents can perform better than those trained without languages. Furthermore, agents trained with more informative languages demonstrate stronger performance.

since the action space grows larger (pedal/lift/grasp, compared to binary opposite actions in Navigation and Object Pick/Drop), and there are no clear implications for the correct action.

F More results on the Messenger environment

In the Messenger environment, models trained with only template foresight or hindsight languages struggle to generalize to diverse languages during testing. Without exposure to diverse languages during training, these models fail to extract the learned hindsight or foresight information from mixed and diverse languages. However, Figure 9 demonstrates that models trained with more diverse hindsight or foresight languages can overcome the generalization problem, and outperform those trained without language feedback, showcasing the importance of diversity in the training languages. Furthermore, the agents trained with both hindsight and foresight information still perform the best, aligning with results in other environments.

G Models and Training

We build our Language-Teachable Decision Transformer based on the code of the original Decision Transformer (Chen et al., 2021). In this section, we will show our training setup and model hyperparameters for each environment.

When selecting the data size, we prioritize the efficient use of a small-scale dataset and examine the impact of language feedback within the constraints of a limited budget and scarce data, as is common in the field of robotics.

G.1 HomeGrid

Estimated parameter size of the models: 12.191 MB. For research question 1, we train the model with 100 trajectories. For research question 2, the pretraining stages use 6432 trajectories. The models are trained on one Nvidia RTX A6000. For research question 1, training takes 3 GPU hours. For research question 2, pretraining takes 4 GPU hours and adaptation takes 3 GPU hours. Hyperparameters shown in Appendix Table 5.

G.2 ALFWorld

Estimated parameter size of the models: 6.5 MB. For research question 1, we train the model with 1000 trajectories. For research question 2, the pretraining stages use 10000 trajectories. The models are trained in one Nvidia RTX A6000. For research question 1, training takes 3 GPU hours. For research question 2, pretraining takes 4 GPU hours and adaptation takes 3 GPU hours. Hyperparameters shown in Appendix Table 6.

G.3 Messenger

Estimated parameters size of the models: 289.681 MB. We train the models with 10000 data trajectories during the pretraining stage for seen tasks. The pretraining stage for seen tasks takes 5 GPU hours on one Nvidia RTX A6000. The adaptation stage for unseen tasks takes 1 GPU hour. Hyperparameters are shown in Appendix Table 7.

G.4 MetaWorld

Estimated parameters size of the models: 289.681 MB. We train the models with 20000 data trajectories during the pretraining stage for seen tasks. The pretraining stage for seen tasks takes 2.5 GPU hours on one Nvidia RTX A6000. The adaptation stage for unseen tasks takes 1 GPU hour. Hyperparameters are shown in Appendix Table 8.

H Examples for Language Feedback in Evaluation

As discussed in section 6.1, we feed template hindsight (l^{hind}) and template foresight (l^{fore}) into an online GPT to generate language feedback as a proxy for real-world human feedback, which can be further extended into multi-turn human-machine dialogue systems in task-oriented settings (He et al., 2022a,b,c). In Figure 10, we demonstrate three examples of the GPT outcome. In example 1, we find GPT can concatenate both hindsight and foresight

Hyperparameters	Value
Number of transformer layers	3
Number of attention heads	1
Embedding dimension	128
Nonlinearity function	ReLU
Batch size	64
Context length K	10
Return-to-go conditioning	1.5
Dropout	0.1
Optimizer	AdamW
Learning Rate	$1e^{-4}$
Grad norm clip	0.25
Weight decay	$1e^{-4}$
Learning rate decay	Linear warmup for first $1e^5$ training steps

Table 5: Hyperparameters of Language-Teachable Decision Transformer for HomeGrid experiments.

Hyperparameters	Value
Number of transformer layers	3
Number of attention heads	1
Embedding dimension	128
Nonlinearity function	ReLU
Batch size	64
Context length K	10
Return-to-go conditioning	1.5
Dropout	0.1
Optimizer	AdamW
Learning Rate	$1e^{-3}$
Grad norm clip	0.25
Weight decay	$1e^{-4}$
Learning rate decay	Consine Annealing with minimum $lr = 1e^{-5}$

Table 6: Hyperparameters of Language-Teachable Decision Transformer for ALFWorld experiments.

and integrate them into a new fluent sentence. In the second example, we observe that GPT decides to discard the hind sight as the outcoı not to respond w doesn't need help.

Hyperparameters	Value
Number of transformer layers	5
Number of attention heads	2
Embedding dimension	128
Nonlinearity function	ReLU
Batch size	128 for pertaining and 1 for adaptation
Context length K	10
Return-to-go conditioning	1.5
Dropout	0.1
Optimizer	AdamW
Learning Rate	$1e^{-3}$ for pretraining and $1e^{-4}$ for adaptation
Grad norm clip	0.25
Weight decay	$1e^{-4}$
Learning rate decay	Linear warmup for first $1e^5$ training steps

Table 7: Hyperparameters of Language-Teachable Decision Transformer for Messenger experiments.

Hyperparameters	Value
Number of transformer layers	5
Number of attention heads	2
Embedding dimension	256
Nonlinearity function	ReLU
Batch size	128 for pertaining and 5 for adaptation
Context length K	12
Return-to-go conditioning	20
Return scale	10
Dropout	0.1
Optimizer	AdamW
Learning Rate	$1e^{-5}$ for pertaining and $1e^{-6}$ for adaptation
Weight decay	$1e^{-4}$
Learning rate decay	Linear warmup for first $1e^5$ training steps

Table 8: Hyperparameters of Language-Teachable Decision Transformer for MetaWorld experiments.

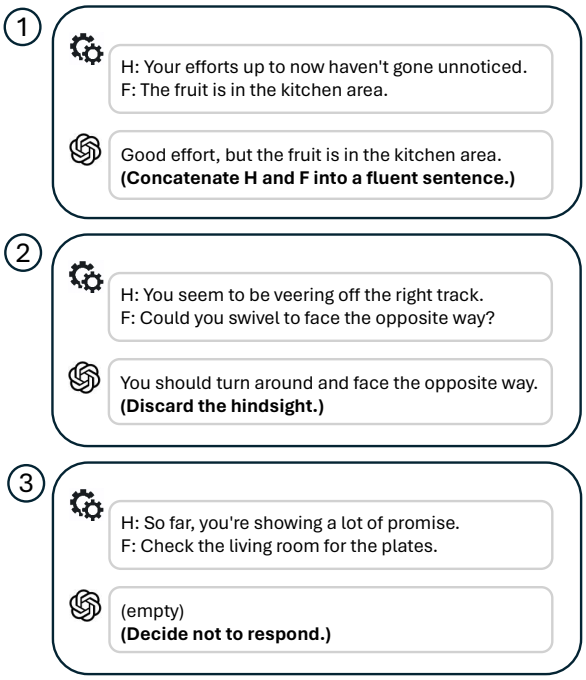

Figure 10: Examples for language feedback generated by online GPT in evaluation.