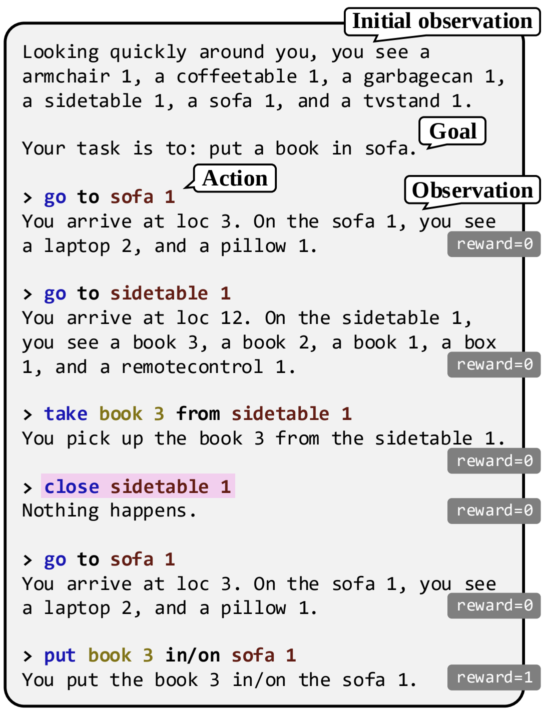
Teaching Text Agents to Learn Sequential Decision Making from Failure

Canasai Kruengkrai[1] **Koichiro Yoshino**[1,2,3]
[1]Guardian Robot Project, RIKEN
[2]Nara Institute of Science and Technology
[3]Institute of Science Tokyo
canasai.kruengkrai@riken.jp,yoshino.k.ai@m.titech.ac.jp

Abstract

Text-based reinforcement-learning agents improve their policies by interacting with their environments to collect more training data. However, these self-collected data inevitably contain *intermediate* failed actions caused by attempting physically infeasible behaviors and/or hallucinations. Directly learning a policy from such trajectories can reinforce incorrect behaviors and reduce task success rates. In this paper, we propose a failed action-aware objective that suppresses the negative impact of failed actions during training by assigning zero return based on textual feedback. Building on this objective, we introduce a perturbation method that leverages unsuccessful trajectories to construct new successful ones that share the same goal. This allows agents to benefit from diverse experiences without further interaction with the environment. Experiments in ALFWorld and ScienceWorld demonstrate that our method significantly outperforms strong baselines and generalizes across environments. Code is available at https://github.com/riken-grp/text-agent.


Initial observation
Goal
Action
Observation

```
Looking quickly around you, you see a
armchair 1, a coffeetable 1, a garbagecan 1,
a sidetable 1, a sofa 1, and a tvstand 1.

Your task is to: put a book in sofa.

> go to sofa 1
You arrive at loc 3. On the sofa 1, you see
a laptop 2, and a pillow 1.                 reward=0

> go to sidetable 1
You arrive at loc 12. On the sidetable 1,
you see a book 3, a book 2, a book 1, a box
1, and a remotecontrol 1.                   reward=0

> take book 3 from sidetable 1
You pick up the book 3 from the sidetable 1.
                                            reward=0
> close sidetable 1
Nothing happens.                            reward=0

> go to sofa 1
You arrive at loc 3. On the sofa 1, you see
a laptop 2, and a pillow 1.                 reward=0

> put book 3 in/on sofa 1
You put the book 3 in/on the sofa 1.        reward=1
```

Figure 1: Example of an episode in ALFWorld (Shridhar et al., 2021) where the agent interacts with the environment, starting from an initial observation and continuing until the goal is completed. The environment returns the failure message `Nothing happens` to indicate that the action **`close sidetable 1`** is invalid.

1 Introduction

Asking an agent such as a robot to do a simple task like "*put a book on the sofa*" requires a wide range of technologies. Moreover, training physical robots in real environments can be costly, time-consuming, and difficult to scale (Pinto and Gupta, 2016; Kalashnikov et al., 2018; Young et al., 2020; Jang et al., 2022; Ahn et al., 2023). Transfer of learning from simulated environments to the real world is a promising strategy for improving cost-time efficiency and scalability (Tobin et al., 2017; Pinto et al., 2018; James et al., 2019). Simulated environments at different levels of abstraction, ranging from *high-level* language instructions to *low-level* embodied executions, offer flexible training grounds for agents to develop their skills (Kolve et al., 2017; Puig et al., 2018; Chevalier-Boisvert et al., 2019; Shridhar et al., 2020; Deitke et al., 2022). Among these, text-based environments provide a particularly efficient way to train agents in high-level reasoning before grounding them in real-world or embodied settings.

Humans have the ability to reason abstractly. For example, to find a book, we typically look on a shelf or table rather than in a garbage can. Ideally, agents should learn to reason in a similar way to select appropriate actions. Text-based environments allow agents to practice abstract reasoning for long-horizon manipulation and navigation

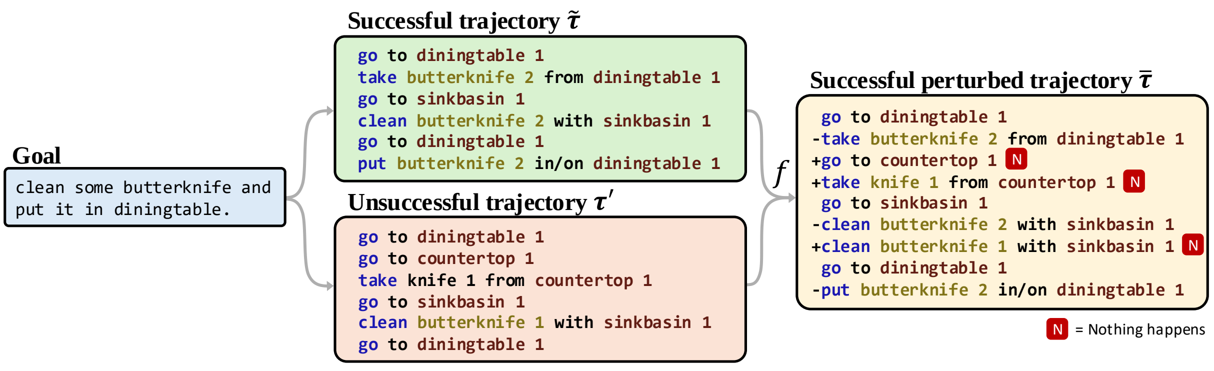

Goal
clean some butterknife and put it in diningtable.
Successful trajectory $\tilde{\tau}$
go to diningtable 1
take butterknife 2 from diningtable 1
go to sinkbasin 1
clean butterknife 2 with sinkbasin 1
go to diningtable 1
put butterknife 2 in/on diningtable 1
Unsuccessful trajectory τ'
go to diningtable 1
go to countertop 1
take knife 1 from countertop 1
go to sinkbasin 1
clean butterknife 1 with sinkbasin 1
go to diningtable 1
f
Successful perturbed trajectory $\bar{\tau}$
go to diningtable 1
-take butterknife 2 from diningtable 1
+go to countertop 1 N
+take knife 1 from countertop 1 N
go to sinkbasin 1
-clean butterknife 2 with sinkbasin 1
+clean butterknife 1 with sinkbasin 1 N
go to diningtable 1
-put butterknife 2 in/on diningtable 1
N = Nothing happens

Figure 2: Our perturbation function f creates a successful perturbed trajectory $\bar{\tau}$ by combining a successful trajectory $\tilde{\tau}$ and an unsuccessful trajectory τ', both of which share the same goal but may come from different scenes. For clarity, we simplify the trajectories by showing only action sequences. The "+" and "−" symbols follow the semantics of a `diff`-style output: in $\bar{\tau}$, unmarked actions are shared between the two input trajectories, actions prefixed with "−" are present in $\tilde{\tau}$ but not in τ', and actions prefixed with "+" are present in τ' but not in $\tilde{\tau}$; these "+" actions are assigned with the failure message `Nothing happens`.

tasks (Côté et al., 2018; Hausknecht et al., 2020; Urbanek et al., 2019; Shridhar et al., 2021; Wang et al., 2022; Yao et al., 2022; Carta et al., 2023; Zhao et al., 2024). Instead of being based on visual input, text-based environments describe the world through natural language, as shown in Figure 1. This enables agents to focus on high-level decision making while bypassing low-level embodied actions.[1] Here, reinforcement learning (RL) is a widely adopted approach to learn effective behaviors in these environments, because it optimizes decision making through interaction and feedback.

A major challenge in text-based environments is the open-ended action space, which makes exploration difficult (Osborne et al., 2022; Jansen and Cote, 2023). Agents using random policies rarely make progress toward task goals (Shridhar et al., 2021; Wang et al., 2022). Expert demonstrations are often needed to guide learning and help the agent identify meaningful actions. Previous work has shown that only a few demonstrations are sufficient in order to initialize a policy, and agents can then collect additional training data through interactions with the environment (Micheli and Fleuret, 2021). However, unlike expert demonstrations, self-collected data may include *intermediate* failed actions caused by attempting physically infeasible behaviors and/or hallucinations. Learning directly from such noisy data may reinforce incorrect behaviors and reduce overall task success.

[1]For instance, a high-level action like `go to sofa 1` may correspond to a sequence of low-level embodied actions such as `LookDown`, `RotateRight`, and `MoveAhead`.

In this paper, we address the problem of intermediate failed actions in self-collected trajectories and ask: *How can we reduce their negative impact during training while still allowing the agent to experience them?* To this end, we introduce a failed action-aware objective that works by inferring returns (i.e., cumulative rewards) from textual observations. The idea is simple: the agent should receive zero return at a particular step if its action causes a failure. This reduces the influence of failed actions during training and encourages the agent to focus on actions that contribute to task success.

Building on this objective, we propose a perturbation method that allows the agent to learn from unsuccessful trajectories. Recent work has shown that unsuccessful trajectories can be helpful, but their effectiveness depends on pairing them with expert demonstrations (Song et al., 2024) or using large language models to extract insights from both successes and failures (Zhao et al., 2024). Our perturbation method relies only on self-collected samples from online interactions. It takes a successful trajectory and an unsuccessful one that shares the same goal, possibly from different scenes, and produces a new successful perturbed trajectory (see Figure 2). Experiments in ALFWorld (Shridhar et al., 2021) and ScienceWorld (Wang et al., 2022) show that our approach outperforms existing methods that learn from failure and generalizes to another environment without tuning hyperparameters.

Our contributions. We address the problem of intermediate failed actions that appear in self-collected successful trajectories in text-based environments (Section 3.1). We propose a failed action-aware objective that infers returns directly from textual observations and reduces the influence of failed actions during training (Section 3.2). We introduce a trajectory perturbation method that augments training data by pairing successful and unsuccessful trajectories with shared goals to generate new successful perturbed trajectories (Section 3.3).

2 Preliminaries

2.1 Problem formulation

We formalize the problem as a discrete, undiscounted, partially observed Markov decision process (POMDP, Kaelbling et al., 1998). A POMDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, P, \Omega, O, R\rangle$, where $\mathcal{S}$ is a set of environment states, $\mathcal{A}$ is a set of actions, $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is a state transition function, Ω is a set of observations, $O : \mathcal{S} \times \mathcal{A} \to \Omega$ is an observation function, and $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function. The functions P, O, and R are hidden from the agent. In this setting, $\mathcal{S}$ contains complete information about the scene (e.g., object positions and states), while O selects from $\mathcal{S}$ what information to show to the agent based on the last action (Côté et al., 2018). We aim to learn a policy π that maximizes the cumulative undiscounted rewards.

The agent uses π to generate actions in order to complete a task within a fixed number of steps. Algorithm 1 outlines this interaction process. In Step 1, the agent receives an initial observation $\boldsymbol{o}_0$, which includes the task goal. In Step 5, the agent takes as input a trajectory history $\boldsymbol{\tau}_{<t} = (\boldsymbol{o}_0, \boldsymbol{o}_1, \boldsymbol{a}_1, r_1, \ldots, \boldsymbol{o}_{t-1}, \boldsymbol{a}_{t-1}, r_{t-1})$ and outputs an action $\boldsymbol{a}_t$. In Step 6, the environment returns a new observation $\boldsymbol{o}_t$, a scalar reward r_t, and a termination signal done, which is `true` when $r_t = 1$ or the step limit is reached (typically 50 steps).

Throughout the paper, we will focus on household tasks and use examples from ALFWorld (Shridhar et al., 2021). However, it is important to note that our approach is not tied to a particular environment. In Section 4.3, we show results from ScienceWorld (Wang et al., 2022), another text-based environment.

2.2 Behavior cloning and reinforcement learning

Our baseline follows the two-stage procedure of Micheli and Fleuret (2021):

(1) Initialize a policy π from expert demonstrations using imitation learning.

Algorithm 1 Agent-environment interaction protocol

```
1:  $\boldsymbol{o}_0 \leftarrow$ env.reset()
2:  $\boldsymbol{\tau} \leftarrow \{(\boldsymbol{o}_0)\}$
3:  $t \leftarrow 1$
4:  while not done do
5:      $\boldsymbol{a}_t \leftarrow$ agent.act($\boldsymbol{\tau}_{<t}$)
6:      $\boldsymbol{o}_t, r_t,$ done $\leftarrow$ env.step($\boldsymbol{a}_t$)
7:      $\boldsymbol{\tau} \leftarrow \boldsymbol{\tau} \cup \{(\boldsymbol{a}_t, \boldsymbol{o}_t, r_t)\}$
8:      $t \leftarrow t + 1$
9:  end while
10: return $\boldsymbol{\tau}$
```

(2) Continue improving π through online reinforcement learning.

In Stage (1), we frame imitation learning as supervised behavior cloning (Pomerleau, 1991). We distinguish between general trajectories $\boldsymbol{\tau}$, which include rewards, and expert trajectories $\boldsymbol{\tau}^*$, which do not. We omit rewards in expert trajectories because supervised behavior cloning does not require them. An expert trajectory is defined as:

$$\boldsymbol{\tau}^* = (\boldsymbol{o}_0, \boldsymbol{a}_1, \boldsymbol{o}_1, \boldsymbol{a}_2, \boldsymbol{o}_2, \ldots, \boldsymbol{a}_T, \boldsymbol{o}_T).$$

Here, T is the final time step of the episode.

We define a stochastic policy $\pi(\boldsymbol{a}_t|\boldsymbol{\tau}^*_{<t}) = p(\boldsymbol{a}_t|\boldsymbol{\tau}^*_{<t}; \theta)$ parameterized by θ. We express the loss on a trajectory $\boldsymbol{\tau}^*$ as the average negative log-likelihood over the actions:

$$L(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \log p(\boldsymbol{a}_t|\boldsymbol{\tau}^*_{<t}; \theta). \tag{1}$$

In other words, we learn a policy that predicts the expert's next action given the trajectory history. Each action $\boldsymbol{a}_t$ consists of a sequence of tokens, $\boldsymbol{a}_t = (a_t^1, \ldots, a_t^{|\boldsymbol{a}_t|})$. We factorize the probability of the action into token-level probabilities:

$$p(\boldsymbol{a}_t|\boldsymbol{\tau}^*_{<t}; \theta) = \prod_{i=1}^{|\boldsymbol{a}_t|} p(a_t^i|a_t^{<i}, \boldsymbol{\tau}^*_{<t}; \theta).$$

We compute each token probability using the output of a large language model (LLM):

$$p(a_t^i|a_t^{<i}, \boldsymbol{\tau}^*_{<t}; \theta) = \text{softmax}\big(\text{LLM}_\theta(a_t^{<i}, \boldsymbol{\tau}^*_{<t})\big).$$

The parameters θ are those in the LLM.

In Stage (2), we apply a Monte Carlo method that lets the agent learn from *complete* episodes of experience. We define a return g_t as the cumulative undiscounted reward r from time step t (Sutton and Barto, 2018):

$$g_t = r_{t+1} + r_{t+2} + \ldots + r_T.$$

The objective maximizes the expected return:

$$J(\theta) = \mathbb{E}[g_t] = \frac{1}{T}\sum_{t=1}^{T} \log p(\boldsymbol{a}_t|\tilde{\boldsymbol{\tau}}_{<t};\theta) g_t, \quad (2)$$

where $\tilde{\boldsymbol{\tau}}$ is a *successful* trajectory that the agent collects by interacting with the environment.

Assuming rewards are delayed until the episode ends, maximizing Eq. (2) becomes equivalent to minimizing Eq. (1) because $g_t = 0+0+\ldots+1 = 1$ for all $\boldsymbol{a}_t$ in $\tilde{\boldsymbol{\tau}}$. Thus, Stages (1) and (2) end up optimizing the same objective over two distinct sources of data, $\boldsymbol{\tau}^*$ and $\tilde{\boldsymbol{\tau}}$. Micheli and Fleuret (2021) refer to these stages as *action modeling* and *iterated action modeling*, respectively.

3 Approach

3.1 Motivating examples

Unlike expert trajectories, self-collected trajectories can contain intermediate failed actions. A common type of failure is an infeasible action. For example, in Figure 1, the agent generates the action `close sidetable 1`, but the verb `close` is incompatible with that object. Figure 3(a) shows another failure where the agent tries to `use cellphone 1`. These actions fail because the agent does not fully understand object properties and interactions.

Here, we do not intend to analyze all types of failed actions. Instead, we will focus on failures where the agent hallucinates the presence of objects. In Figure 3(b), the agent believes that `pencil 3` is on `sidetable 1` and tries to take it to `garbagecan 1`, resulting in two failed actions.[2] In Figure 3(c), the agent navigates to `sofa 1` and then attempts to go to `sofa 2`, which does not exist in the current scene.

In text-based environments like ALFWorld, hallucinations are difficult to avoid partly because of the way scenes are configured. Objects or receptacles of the same type are distinguished by numeric IDs (e.g., `sofa 1`, `sofa 2`, and so on). These ID tokens are reused across scenes, and the agent often starts with a correct object type but hallucinates non-existing IDs. For example, it may generate `go to sofa 2` even if only `sofa 1` exists in the scene.

[2]This failure is caused by a language model hallucination. However, similar issues can arise in real environments due to sensor noise or perceptual errors.

```
> take cellphone 1 from sidetable 2
You pick up the cellphone 1 from the
sidetable 2.

> use cellphone 1
Nothing happens.

> go to safe 1
You arrive at loc 14. The safe 1 is closed.
```

(a) Physical infeasibility

```
> go to sidetable 1
You arrive at loc 10. On the sidetable 1,
you see a cd 3, a creditcard 3, and a
keychain 1.

> take pencil 3 from sidetable 1
Nothing happens.

> go to garbagecan 1
You arrive at loc 3. On the garbagecan 1,
you see a cd 2.

> put pencil 3 in/on garbagecan 1
Nothing happens.
```

(b) Object hallucination

```
> go to sofa 1
You arrive at loc 45. On the sofa 1, you
see a remotecontrol 2.

> go to sofa 2
Nothing happens.

> go to shelf 1
You arrive at loc 23. On the shelf 1, you
see a statue 1.

> take statue 1 from shelf 1
You pick up the statue 1 from the shelf 1.
```

(c) Location hallucination

Figure 3: Examples of intermediate failed actions during online interaction.

3.2 Failed action-aware objective

The presence of failed actions becomes problematic for iterated action modeling because it treats all actions, including failed ones, equally when updating the policy from a successful trajectory $\tilde{\boldsymbol{\tau}}$. A direct approach is to remove failed actions from the training data before updating the policy. In the early stages of our research, we tested several filtering methods, but they hindered the agent's exploration and provided only minimal improvements (see Section 4.2).

Beyond filtering, another possible approach is to replace the return g_t in Eq. (2) with an advantage function (Schulman et al., 2016), which estimates how much better or worse an action performs compared with the average. However, this method requires additional model parameters. Instead, we propose a simple technique that directly infers returns from textual observations.

Text-based environments typically provide specific messages for failed actions.[3] Let $\mathcal{F}$ be a set of known failure messages. We define the *failed action-aware objective* as:

$$\hat{J}(\theta) = \frac{1}{T}\sum_{t=1}^{T} \log p(\boldsymbol{a}_t|\tilde{\boldsymbol{\tau}}_{<t};\theta)A(\boldsymbol{o}_t), \quad (3)$$

where

$$A(\boldsymbol{o}_t) = \begin{cases} 0 & \text{if } \boldsymbol{o}_t \in \mathcal{F}, \\ g_t & \text{otherwise.} \end{cases}$$

We can view A as a binary advantage function, suggesting that there is no advantage to taking a failed action.[4] Our failed action-aware objective $\hat{J}$ introduces minimal computational overhead compared with the vanilla objective in Eq. (2). Unlike filtering methods, we retain failed actions in $\tilde{\boldsymbol{\tau}}$ in order to allow the agent to learn from failure.

Our method relies on capturing the deterministic failure messages, which are sufficient for the environments we target. Generalization to environments with more diverse failure signals may require developing a signal classifier. We leave this for future work.

3.3 Trajectory perturbation from successful and unsuccessful pairs

As described in Section 2.2, the agent improves its policy by using only successful trajectories. We inspect the number of successful and unsuccessful trajectories the agent collects during online interaction. Figure 4 shows the counts across 10 trials under the failed action-aware objective $\hat{J}$. Like many online reinforcement learning methods, iterated action modeling discards unsuccessful trajectories. We address this limitation by generating new successful trajectories from pairs of successful and unsuccessful ones that share the same goal.

Let $\boldsymbol{\tau}'$ be an unsuccessful trajectory. We pair $\boldsymbol{\tau}'$ with a successful trajectory $\tilde{\boldsymbol{\tau}}$ (without fail actions) if they share the same goal. We define a perturbation function $f(\tilde{\boldsymbol{\tau}}, \boldsymbol{\tau}')$ that returns a successful perturbed trajectory $\bar{\boldsymbol{\tau}}$, and apply it to all pairs of goal-aligned $\tilde{\boldsymbol{\tau}}$ and $\boldsymbol{\tau}'$ found in the collected trajectories.[5] Figure 2 illustrates how the perturbation function f generates $\bar{\boldsymbol{\tau}}$ by comparing the action sequences of $\tilde{\boldsymbol{\tau}}$ and $\boldsymbol{\tau}'$.

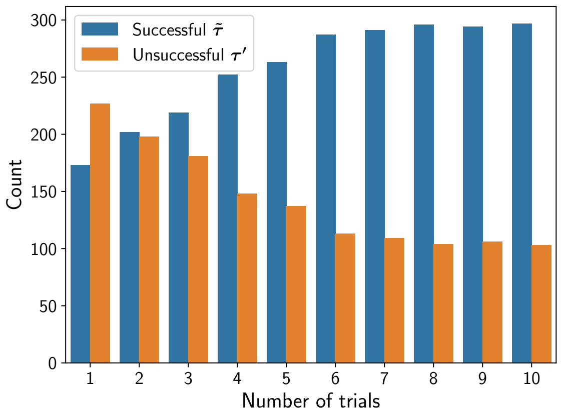


Figure 4: Number of successful and unsuccessful trajectories collected in 10 trials, each consisting of 400 episodes. As interaction progresses, the agent collects more successful trajectories than unsuccessful ones.

We implement the perturbation function f using `unified_diff` from Python's `difflib` library to identify differences between the action sequences of $\tilde{\boldsymbol{\tau}}$ and $\boldsymbol{\tau}'$. We annotate actions from $\boldsymbol{\tau}'$ that do not appear in $\tilde{\boldsymbol{\tau}}$ with the failure message and include them in $\bar{\boldsymbol{\tau}}$. We truncate $\bar{\boldsymbol{\tau}}$ at the point where $\tilde{\boldsymbol{\tau}}$ ends to ensure that the perturbed trajectory preserves a successful state. The process takes less than one second per pair.

Our augmented training data consist of successful trajectories and their perturbed versions. We use them to update the policy after online interaction. Since our objective $\hat{J}$ allows failed actions in the trajectory histories, we can learn from $\bar{\boldsymbol{\tau}}$ even when it includes failed segments. These perturbed trajectories provide diverse contexts and help to train a more robust policy without requiring additional environment interactions.

4 Experiments

Setup. We focused on six household tasks in ALFWorld (Shridhar et al., 2021) derived from the AL-

[3]Both ALFWorld and ScienceWorld provide deterministic failure messages: `Nothing happens` and `No known action matches that input`, respectively.

[4]Our method is also similar to reward shaping (Ng et al., 1999), where the function A is external to the environment and independent of past states, ensuring the process remains Markovian.

[5]For example, in Figure 4, we derived 1,962 successful trajectories *without* failed actions and 1,426 unsuccessful trajectories from a pool of 4,000 episodes. The successful and unsuccessful trajectories both had 713/618 unique goals and shared 382 goals. We could generate 2,468 successful perturbed trajectories.

Task Type	Train	Seen	Unseen
Pick & Place	790	35	24
Examine in Light	308	13	18
Clean & Place	650	27	31
Heat & Place	459	16	23
Cool & Place	533	25	21
Pick Two & Place	813	24	17
All	3,553	140	134

Table 1: Statistics of task types in ALFWorld. The seen set contains known tasks, but the object locations and quantities are different from the training set, whereas the unseen set contains new tasks in the unseen scenes.

FRED benchmark (Shridhar et al., 2020). Table 1 lists the numbers of tasks in the training, seen, and unseen sets. We obtained expert demonstrations by running an expert agent on the training tasks. The expert agent could access permissible actions and used a hand-coded deterministic policy provided by the simulator.

Each of the seen and unseen sets has two goal versions: templated-based and human-annotated. Both versions use the same scenes, but the human-annotated goals introduce unseen verbs and nouns, creating additional generalization challenges. Appendix F illustrates the differences between these goal types.

To ensure comparability with prior work, we followed the same experimental setup as in Micheli and Fleuret (2021). We used GPT-2 medium (355M) (Radford et al., 2018) as our pre-trained LLM. We reimplemented the action modeling (**AM**) and iterated action modeling (**IAM**), which yield better performance than the original versions. For AM, we randomly sampled *seven* expert demonstrations per task type (42 in total) to train an initial policy. For IAM, we started with the AM policy and let the agent learn by interacting with the environment over 10 trials, each consisting of 400 episodes. The two variants of our approach are as follows:

IAM-FA applies our failed action-aware (**FA**) objective $\hat{J}$ described in Section 3.2.

IAM-FA + DA incorporates our data augmentation (**DA**) described in Section 3.3 to further train the IAM-FA policy offline.

Implementation. We used the Transformers library (Wolf et al., 2020). We optimized the model parameters by using Adafactor (Shazeer and Stern, 2018). We followed most of the hyperparameter settings in Micheli and Fleuret (2021), as detailed in Appendix A. During the online interaction, the trajectory history could exceed the maximum context length (i.e., 1000 tokens). We handled this case by truncating the oldest action and observation pairs while always retaining the initial observation that contains the goal.

Training times. All models were trained using a single A6000 GPU per run. AM took around 5 minutes to train. IAM/IAM-FA took around 2 hours (i.e., 12 minutes per trial). IAM-FA + DA took around 30 minutes.

Comparisons. We compared our approach against the following methods:

BUTLER (Building Understanding in TextWorld via Language for Embodied Reasoning, Shridhar et al., 2021) is a text agent trained with 50K episodes via imitation learning (Ross et al., 2011). The authors also present the **Seq2Seq** baseline, an encoder-decoder model based on Transformers (Vaswani et al., 2017) trained with all expert demonstrations.

ExpeL (Experiential Learning, Zhao et al., 2024) is an in-context learning-based agent that extracts insights from successful/unsuccessful trajectories and then composes prompts by combining these insights with similar successful examples. Each insight is a list of instructions generated by GPT-4. ExpeL can be combined with **Reflexion** (Shinn et al., 2023) to retry failed tasks multiple times with a long-term memory buffer.

ETO (Exploration-based Trajectory Optimization, Song et al., 2024) pairs unsuccessful trajectories produced by a base agent with expert trajectories and trains the policy by using direct preference optimization (DPO, Rafailov et al., 2023). The authors annotate each action in an expert trajectory with a chain-of-thought rationale (Wei et al., 2022) using GPT-4. These annotated expert trajectories are used to fine-tune the base agent with supervised fine-tuning (**SFT**), followed by policy updates using ETO. Their model uses Llama-2-7B-chat (Touvron et al., 2023) as the pre-trained LLM.

4.1 Main results

Table 2 shows the results of the various methods. IAM-FA + DA outperforms other methods that learn from failure (i.e., ExpeL and ETO). IAM-FA consistently performs better than the baseline IAM, suggesting that our objective $\hat{J}$ is effective.

	Method	Template Seen	Template Unseen	Human Seen	Human Unseen
Shridhar et al. (2021)	Seq2Seq	10	9	–	–
	BUTLER	40	37	–	–
Zhao et al. (2024)	ExpeL	–	59	–	–
	ExpeL + Reflexion	–	64	–	–
Song et al. (2024)	SFT	60	67	–	–
	ETO	69	72	–	–
Micheli and Fleuret (2021)	AM	48	35	19	17
	IAM	76	68	35	37
This work	IAM-FA	82	76†	39	44
	IAM-FA + DA	**87**†	**78**†	**51**†	**55**†

Table 2: Success rates (%) on the *seen* and *unseen* sets of *template*-based and *human*-annotated goals in ALFWorld. For AM and IAM variants, we report the average success rate computed over five training/test runs from different random seeds. The other results are taken directly from the papers. The symbol † indicates a statistically significant difference ($p < 0.05$, Mann–Whitney U test) compared with IAM (see Appendix B for more details).

Method	Pick Sn	Pick Un	Examine Sn	Examine Un	Clean Sn	Clean Un	Heat Sn	Heat Un	Cool Sn	Cool Un	Pick Two Sn	Pick Two Un
AM	59	40	43	36	65	32	50	63	47	27	12	5
IAM	85	70	51	37	94	75	90	**95**	80	80	43	32
IAM-FA (ours)	87	81	62	54	94	78	**94**	**95**	79	87	64	48
IAM-FA + DA (ours)	**93**	**84**	**77**	**56**	**97**	**83**	92	**95**	**88**	**89**	**70**	**53**

Table 3: Success rates (%) by task type for template-based goals. Sn = Seen; Un = Unseen.

AM performs poorly, but this is expected since it is trained with only 1.2% of expert demonstrations.

To better understand the range of task difficulties, we inspected the results by task type on template-based goals. Table 3 shows that success rates vary significantly across tasks. The most challenging task is "Pick Two & Place", which requires the agent to find an object, pick it up, find a location to place it, and repeat this process for a second object within the 50-step limit. Appendix C shows success cases where IAM-FA + DA completes the task but IAM fails.

4.2 Ablation studies

How important is the failed action-aware objective for using augmented data effectively? We removed our objective $\hat{J}$ from the training process, which is equivalent to running IAM and performed further training with augmented data without $\hat{J}$. Table 4 shows the results.[6] The sharp drops in performance indicate that $\hat{J}$ and our data augmentation work together to achieve the best results.

[6]The success rates in the ablation studies and additional experiments are also averaged over five training/test runs.

How does our data augmentation compare with alternative strategies? We evaluated three alternative strategies. First, we excluded the successful perturbed trajectories and used only the successful ones. Second, we used only the unsuccessful trajectories. Third, we combined both successful and unsuccessful trajectories without perturbation. As shown in Table 5, all these alternative strategies perform worse than our approach.

Is filtering failed actions a viable alternative to our objective? As discussed in Section 3.2, a simple way to tackle failed actions is to remove them from the training data. We tested this idea by (1) filtering out failed actions from the trajectories and (2) discarding any trajectory that contained at least one failed action. Table 6 shows that the results are mixed. While IAM with action filtering performs slightly better on the human-annotated goals, it does not allow the agent to experience failure. In contrast, IAM-FA retains failed actions,

Method	Template Sn	Template Un	Human Sn	Human Un
IAM-FA + DA	**87**	**78**	**51**	**55**
w/o $\hat{J}$	81	73	45	49

Table 4: Ablation of the failed action-aware objective.

Method	Template Sn	Template Un	Human Sn	Human Un
IAM-FA + DA	**87**	**78**	**51**	**55**
augment w/ $\tilde{\tau}$	85	76	44	48
augment w/ τ'	70	61	33	36
augment w/ $\tilde{\tau} \cup \tau'$	82	70	40	47

Table 5: Ablations of different augmentation strategies.

Method	Template Sn	Template Un	Human Sn	Human Un
IAM-FA	**82**	**76**	39	44
IAM w/ act. filtering	79	74	**40**	**45**
IAM w/ traj. filtering	80	75	36	43

Table 6: Ablations of handling failed actions.

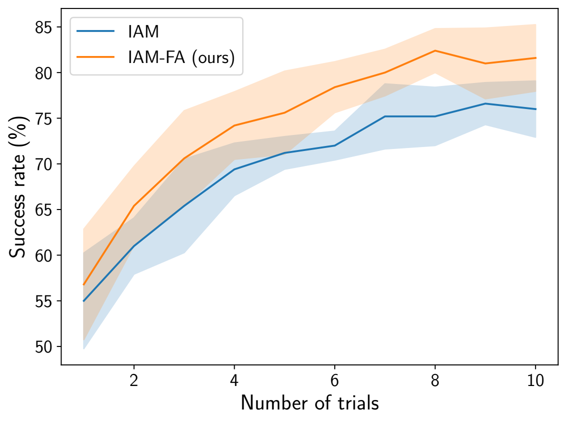


Figure 5: Mean success rate (%) of the policy at each trial computed over five training/test runs on the seen set of ALFWorld's template-based goals. Given the same number of episodes, IAM-FA helps the agent learn effective policies more quickly than IAM, leading to higher success rates across all trials.

allowing the agent to learn from failure and benefit from our data augmentation method.

4.3 Generalization and scaling results

How useful are self-collected trajectories for policy improvement? We have already seen in Figure 4 that the agent progressively collects successful trajectories over time. To assess their utility, we evaluated the policy at the end of each trial. Figure 5 shows that IAM-FA achieves higher success rates than IAM in every trial, indicating that it learns more effectively from the same number of episodes.

Can our approach be applied to a new environment? We applied our method to ScienceWorld (Wang et al., 2022), a different text-based environment with diverse scientific tasks. Following Song et al. (2024), we used a subset of 24 task types with identical training, seen, and unseen splits (1,483/194/211 tasks). We compared our method against SFT and ETO from their work, which achieved strong results on this task set. We reused the hyperparameter settings from ALFWorld *without* any modification. For AM, we sampled seven expert demonstrations per task type, resulting in 164 demonstrations in total. We also trained AM*, where the asterisk indicates training with all expert demonstrations in this split, to estimate an upper-bound performance. Table 7 shows the average scores on seen and unseen sets.[7] AM* scores better than SFT and is competitive with ETO, while using 19.7x fewer parameters (GPT-2 medium 355M vs. Llama-2-7B-chat) and no GPT-4 annotations.

In ScienceWorld, IAM-FA + DA improves performance and nearly matches AM* on the unseen set, despite using far fewer expert demonstrations. IAM-FA also outperforms both AM and IAM, but the gap is smaller than in ALFWorld. We observed that successful trajectories in ScienceWorld contain fewer intermediate failed actions, which in turn limits the contribution of our failed action-aware objective during training. Appendix D shows a success case in ScienceWorld.

How does our approach perform with other LLMs without tuning? We tested our approach with GPT-2 small (124M) and Pythia-410M (Biderman et al., 2023) using the same hyperparameters as GPT-2 medium. These experiments assessed whether IAM-FA and IAM-FA + DA generalize across model scales without adjustment. As shown in Table 9 (Appendix E), both variants outperform AM and IAM, with larger improvements on the harder human-annotated goals. These results indicate the robustness of our method when it is applied to smaller or mid-sized language models.

[7] ScienceWorld returns an intermediate score ranging between 0 and 100 to reflect the task progress.

Method	Seen	Unseen
SFT	67.4	53.0
ETO	73.8	65.0
AM*	71.5	61.8
AM	59.8	54.6
IAM	65.8	59.4
IAM-FA (ours)	66.3	60.5
IAM-FA + DA (ours)	67.9	61.3

Table 7: Average scores on the seen and unseen sets in ScienceWorld. The top section includes methods trained with all expert demonstrations, including AM*. Results for SFT and ETO are taken directly from Song et al. (2024) for comparison. The bottom section reports results from methods that rely on limited expert data and learn through online interaction. All values are averaged over five training/test runs with different random seeds.

Can our method scale to a larger language model? To evaluate scalability, we conducted additional experiments using Llama-3.1-8B (Grattafiori et al., 2024) with parameter-efficient fine-tuning (PEFT, Mangrulkar et al., 2022) via low-rank adaptation (LoRA, Hu et al., 2022). We used LoRA with $r = 16$, $\alpha = 32$, dropout $= 0.05$, and applied it to all linear layers. Because LoRA updates only a small number of parameters, we increased the learning rate for each method by a factor of ten over its original value. We also halved the number of episodes per trial from 400 to 200 to keep training time manageable. IAM and IAM-FA required approximately four hours to train. All models used the same hyperparameters across ALFWorld and ScienceWorld. Tables 10 and 11 (Appendix E) show that IAM-FA and IAM-FA + DA outperform AM and IAM on both ALFWorld and ScienceWorld, confirming that our method remains robust when scaled to a larger model.

5 Related work

Text-based environments designed as interactive games date back to the late 1970s (Lebling et al., 1979), distinguishing them from earlier symbolic systems like SHRDLU (Winograd, 1972). For a comprehensive survey of text-based environments, we refer readers to Jansen (2022). Recent environments adopt the OpenAI Gym interface (Brockman et al., 2016), providing a unified API for agents to interact with the environment (Côté et al., 2018; Hausknecht et al., 2020; Shridhar et al., 2021; Wang et al., 2022; Yao et al., 2022; Zhou et al., 2024). Text agents have been developed using imitation learning (Shridhar et al., 2021), reinforcement learning (Yao et al., 2020; Carta et al., 2023), and hybrid approaches (Lin et al., 2023; Song et al., 2024). Another line of work focuses on in-context learning, which uses prompts (e.g., instructions or few-shot examples) to interact with LLMs without parameter updates (Yao et al., 2023; Shinn et al., 2023; Zhao et al., 2024). Most studies rely on commercial LLMs, making fair comparisons and reproducibility difficult. Our work is inspired by Micheli and Fleuret (2021), who introduced iterated action modeling to improve policies by using self-collected trajectories. We extended that framework by addressing intermediate failed actions within successful trajectories and by leveraging unsuccessful ones to improve learning.

Learning from failure has been extensively studied in non-text-based environments. Shiarlis et al. (2016) introduced an inverse reinforcement learning (IRL) algorithm that extracts a reward function from failed demonstrations. However, modeling reward functions in partially observable settings remains challenging (Choi and Kim, 2011). Andrychowicz et al. (2017) proposed hindsight experience replay (HER), which generates alternative goals for unsuccessful trajectories. While effective in static robotic environments, HER is difficult to apply to open-ended text-based tasks. In text-based settings, our work is closely related to Zhao et al. (2024); Song et al. (2024), who pair successful and unsuccessful trajectories to support learning. In contrast, our approach does not require commercial LLMs or expert-annotated trajectories to create augmented training data. Our perturbation method is deterministic, computationally efficient, and easy to implement.

6 Conclusion

We addressed the problem of successful trajectories that contain intermediate failed actions. Learning directly from such trajectories can degrade task performance. Our failed action-aware objective suppresses the impact of failed actions by assigning zero return at those steps. It also enables the use of unsuccessful trajectories to generate additional training data. Together, these techniques lead to consistent improvements over existing baselines. Extending our approach to embodied agents remains an important direction for future work.

Limitations

The term $A(\boldsymbol{o}_t)$ in Eq. (3) is defined as a deterministic 0-1 advantage function. It assigns zero return to failed actions and a fixed return of one to all non-failed actions, assuming the environment gives a reward only at the end of a successful trajectory. This setting does not distinguish non-failed actions that contribute to meaningful subgoals from those that do not. One possible extension is to design graded or learned advantage functions that reflect subgoal relevance.

Our perturbation function assumes the availability of successful and unsuccessful trajectories that share the same goal. In some environments, finding exact goal matches may be difficult. One possible workaround is to represent goals by using sentence embeddings and select pairs with high semantic similarity.

IAM-FA and IAM-FA + DA rely on the presence of deterministic failure messages returned by the environment. This assumption holds in ALFWorld and ScienceWorld, where failure signals are consistent and well-defined. In environments with ambiguous or missing failure feedback, identifying failed actions may be more challenging. One possible solution is to train a classifier that predicts failures based on the observation and action context.

Acknowledgments

This work is partially supported by the JST MOONSHOT Project under Grant Number JPMJMS2236.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. 2023. Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 287–318. PMLR.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *ArXiv*.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. 2023. Grounding large language models in interactive environments with online reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 3676–3713. PMLR.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2019. BabyAI: First steps towards grounded language learning with a human in the loop. In *International Conference on Learning Representations*.

Jaedeug Choi and Kee-Eung Kim. 2011. Inverse reinforcement learning in partially observable environments. *Journal of Machine Learning Research*, 12(21):691–730.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. TextWorld: A learning environment for text-based games. *ArXiv*.

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. 2022. rocTHOR: Large-scale embodied AI using procedural generation. In *Advances in Neural Information Processing Systems*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *ArXiv*.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7903–7910.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey

Levine, Raia Hadsell, and Konstantinos Bousmalis. 2019. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. 2022. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 991–1002. PMLR.

Peter Jansen. 2022. A systematic survey of text worlds as embodied natural language environments. In *Proceedings of the 3rd Wordplay: When Language Meets Games Workshop (Wordplay 2022)*, pages 1–15, Seattle, United States. Association for Computational Linguistics.

Peter Jansen and Marc-alexandre Cote. 2023. TextWorldExpress: Simulating text games at one million steps per second. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 169–177, Dubrovnik, Croatia. Association for Computational Linguistics.

Leslie P. Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. 2018. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 651–673. PMLR.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. 2017. AI2-THOR: An Interactive 3D Environment for Visual AI. *ArXiv*.

Lebling, Blank, and Anderson. 1979. Special feature zork: A computerized fantasy simulation game. *Computer*, 12(4):51–59.

Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. 2023. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. PEFT: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

Vincent Micheli and Francois Fleuret. 2021. Language models are few-shot butlers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9312–9318, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999*, pages 278–287. Morgan Kaufmann.

Philip Osborne, Heido Nõmm, and André Freitas. 2022. A Survey of Text Games for Reinforcement Learning Informed by Natural Language. *Transactions of the Association for Computational Linguistics*, 10:873–887.

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. 2018. Asymmetric actor critic for image-based robot learning. In *Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*.

Lerrel Pinto and Abhinav Gupta. 2016. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3406–3413.

Dean A. Pomerleau. 1991. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. 2018. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language models are unsupervised multitask learners.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA. PMLR.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2016. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4596–4604.

Kyriacos Shiarlis, Joao Messias, and Shimon Whiteson. 2016. Inverse reinforcement learning from failure. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, AAMAS '16, pages 1060–1068, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

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

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 *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10737–10746.

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

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and error: Exploration-based trajectory optimization of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7584–7600, Bangkok, Thailand. Association for Computational Linguistics.

Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*, second edition. The MIT Press.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, pages 23–30. IEEE.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*.

Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. 2019. Learning to speak and act in a fantasy text adventure game. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 673–683, Hong Kong, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11279–11298, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Terry Winograd. 1972. Understanding natural language. *Cognitive Psychology*, 3:1–191.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

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

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. 2020. Keep CALM and explore: Language models for action generation in text-based games. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8736–8754, Online. Association for Computational Linguistics.

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

Sarah Young, Dhiraj Gandhi, Shubham Tulsiani, Abhinav Gupta, Pieter Abbeel, and Lerrel Pinto. 2020. Visual imitation made easy. *ArXiv*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19632–19642.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*.

Number of epochs	50
Batch size	1
Gradient accumulation step	7
Learning rate	5e-5
LR schedule	Constant
Max sequence length	1000
LLM's decoding	Greedy

(a) AM

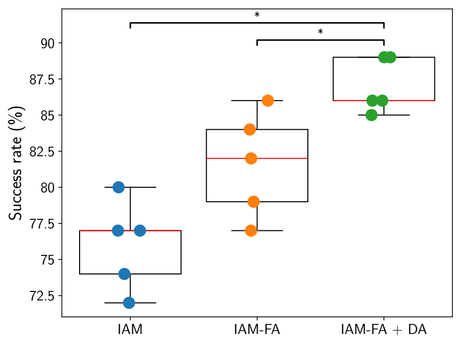
Number of trials	10
Episodes per trial	400
Batch size	1
Gradient accumulation step	8
Learning rate	1e-5
LR schedule	Constant
Max action length	50
Max sequence length	1000
LLM's decoding	Sampling

(b) IAM and IAM-FA

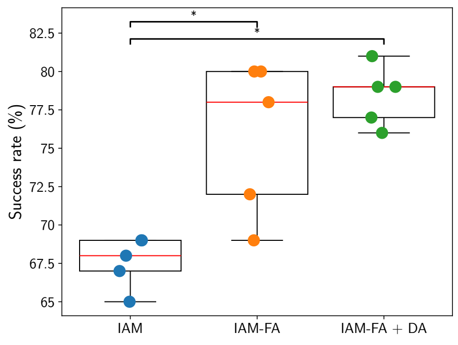
Number of epochs	2
Batch size	1
Gradient accumulation step	8
Learning rate	1e-5
LR schedule	Linear
Max sequence length	1000
LLM's decoding	Greedy

(c) IAM-FA + DA

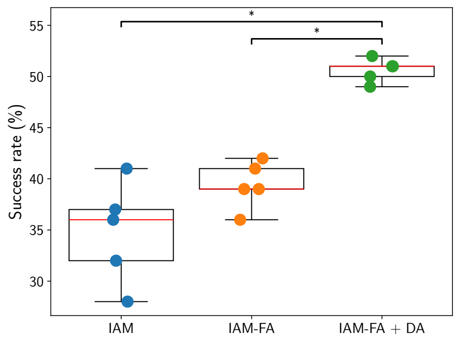
Table 8: Key hyperparameters for AM, IAM, IAM-FA, and IAM-FA + DA.



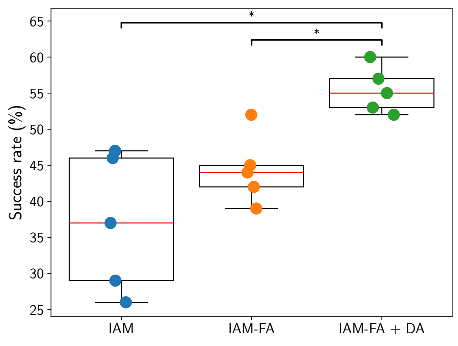
(a) Seen set of template-based goals



(b) Unseen set of template-based goals



(c) Seen set of human-annotated goals



(d) Unseen set of human-annotated goals

Figure 6: Boxplots showing success rates and results of the Mann–Whitney U test for the seen and unseen sets of template-based and human-annotated goals in ALFWorld. Each dot represents the success rate from a random seed. The symbol $*$ indicates a statistically significant difference at $p < 0.05$.

A Hyperparameters

Table 8 lists the key hyperparameters used in our experiments. All other settings follow the default values in the Transformers library. We used the same hyperparameters for both ALFWorld and ScienceWorld. For Llama-3.1-8B experiments with LoRA, we increased each learning rate by a factor of ten and reduced the number of episodes per trial from 400 to 200 in IAM and IAM-FA.

B Statistical significance tests

We performed five training/test runs using different random seeds. We then conducted statistical significance tests to evaluate the performance differences among IAM variants. For each combination of goal type (*template*-based or *human*-annotated) and data split (*seen* or *unseen*), we conducted statistical significance tests using the Mann–Whitney U test across all pairs of IAM variants (IAM, IAM-FA, and IAM-FA + DA). Figure 6 shows the corresponding boxplots for the four settings in ALFWorld.

C Example trajectories in ALFWorld

Figure 7 shows successful trajectories for the "Clean & Place" and "Pick Two & Place" tasks. Both are from the unseen set of template-based goals. IAM-FA + DA successfully completes these tasks, while IAM fails.

D Example trajectory in ScienceWorld

ScienceWorld (Wang et al., 2022) tests agents on tasks aligned with an elementary school science curriculum (e.g., boiling water or finding living things). Figure 8 shows a successful trajectory for the "find-plant" task completed by IAM-FA + DA.

E Results for other LLMs

This appendix presents additional results using LLMs other than GPT-2 medium. Table 9 shows results on ALFWorld using GPT-2 small and Pythia-410M, both evaluated with the same hyperparameters as GPT-2 medium. These experiments test the robustness of our method without hyperparameter tuning.

Tables 10 and 11 show results when using Llama-3.1-8B with PEFT via LoRA. All hyperparameters were adjusted consistently across ALFWorld and ScienceWorld, as described in Section 4.3. These experiments demonstrate that our method scales effectively to a larger model.

	Template		Human	
Method	Seen	Unseen	Seen	Unseen
AM	34	30	14	16
IAM	40	37	19	22
IAM-FA	57	47	27	27
IAM-FA + DA	**70**	**64**	**40**	**40**

(a) GPT-2 small (124M)

	Template		Human	
Method	Seen	Unseen	Seen	Unseen
AM	34	25	16	15
IAM	59	55	29	29
IAM-FA	72	64	31	37
IAM-FA + DA	**75**	**67**	**39**	**45**

(b) Pythia-410M

Table 9: Results on ALFWorld using GPT-2 small and Pythia-410M. All models were trained with the same hyperparameters as GPT-2 medium. We report mean success rates (%) over five training/test runs with different random seeds.

	Template		Human	
Method	Seen	Unseen	Seen	Unseen
AM	61	66	25	36
IAM	81	78	35	44
IAM-FA	84	80	38	46
IAM-FA + DA	**92**	**92**	**50**	**57**

Table 10: Results on ALFWorld using Llama-3.1-8B. All models were fine-tuned with LoRA. We report mean success rates (%) over five training/test runs with different random seeds.

Method	Seen	Unseen
AM*	79.0	70.6
AM	65.4	56.9
IAM	70.0	60.3
IAM-FA	71.6	63.3
IAM-FA + DA	73.6	65.0

Table 11: Results on ScienceWorld using Llama-3.1-8B. The top section shows the AM* result, trained with all expert demonstrations. The bottom section presents methods that use limited demonstrations and learn through online interaction. All models were fine-tuned with LoRA and evaluated over five training/test runs with different random seeds.

```
Looking quickly around you, you see a cabinet 6, a cabinet
5, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1,
 a coffeemachine 1, a countertop 3, a countertop 2, a
countertop 1, a drawer 3, a drawer 2, a drawer 1, a fridge
1, a garbagecan 1, a microwave 1, a shelf 3, a shelf 2, a
shelf 1, a sinkbasin 1, a stoveburner 4, a stoveburner
3, a stoveburner 2, a stoveburner 1, and a toaster 1.

Your task is to: put a clean knife in countertop.

> go to sinkbasin 1
You arrive at loc 30. On the sinkbasin 1, you see a
butterknife 2, a cup 2, a egg 2, and a spoon 1.

> take knife 2 from sinkbasin 1
Nothing happens.

> go to countertop 1
You arrive at loc 13. On the countertop 1, you see a apple
3, a apple 2, a creditcard 1, and a peppershaker 2.

...[truncated]...

> go to countertop 3
You arrive at loc 6. On the countertop 3, you see a bread
1, a houseplant 1, a knife 1, a peppershaker 3, a pot 1, a
soapbottle 1, a spatula 3, and a statue 3.

> take knife 1 from countertop 3
You pick up the knife 1 from the countertop 3.

> go to sinkbasin 1
You arrive at loc 30. On the sinkbasin 1, you see a
butterknife 2, a cup 2, a egg 2, and a spoon 1.

> clean knife 1 with sinkbasin 1
You clean the knife 1 using the sinkbasin 1.

> look
You are facing the sinkbasin 1. Next to it, you see
nothing.

> go to countertop 1
You arrive at loc 13. On the countertop 1, you see a apple
3, a apple 2, a creditcard 1, and a peppershaker 2.

> put knife 1 in/on countertop 1
You put the knife 1 in/on the countertop 1.
```

(a) Clean & Place

```
Looking quickly around you, you see a bed 1, a desk 2, a
desk 1, a drawer 6, a drawer 5, a drawer 4, a drawer 3,
a drawer 2, a drawer 1, a garbagecan 1, a laundryhamper 1,
a safe 1, a shelf 6, a shelf 5, a shelf 4, a shelf 3,
a shelf 2, and a shelf 1.

Your task is to: find two cd and put them in safe.

...[truncated]...

> go to drawer 4
You arrive at loc 24. The drawer 4 is closed.

> open drawer 4
You open the drawer 4. The drawer 4 is open. In it, you
see a cd 3, and a creditcard 1.

> take cd 3 from drawer 4
You pick up the cd 3 from the drawer 4.

> go to safe 1
You arrive at loc 8. The safe 1 is closed.

> open safe 1
You open the safe 1. The safe 1 is open. In it, you see
nothing.

> put cd 3 in/on safe 1
You put the cd 3 in/on the safe 1.

> close safe 1
You close the safe 1.

> go to desk 1
You arrive at loc 14. On the desk 1, you see a alarmclock
2, a bowl 1, a cd 1, a cellphone 1, a desklamp 1, and a
mug 1.

> take cd 1 from desk 1
You pick up the cd 1 from the desk 1.

> go to safe 1
You arrive at loc 8. The safe 1 is closed.

> open safe 1
You open the safe 1. The safe 1 is open. In it, you see a
cd 3.

> put cd 1 in/on safe 1
You put the cd 1 in/on the safe 1.
```

(b) Pick Two & Place

Figure 7: Examples of successful trajectories on the unseen set of template-based goals in ALFWorld, completed by IAM-FA + DA.

```
Your task is to find a(n) plant. First, focus on the thing. Then, move it to
the yellow box in the bedroom.

> look around
This room is called the living room. In it, you see:
    the agent
    a substance called air
    a chair. On the chair is: nothing.
    a couch. On the couch is: a white pillow.
    a desk. On the desk is: a drawer.
    a finger painting
You also see:
    A door to the hallway (that is open) [score=0]

> teleport to greenhouse
You teleport to the greenhouse. [score=17]

> look around
This room is called the greenhouse. In it, you see:
    the agent
    a substance called air
    a bee hive. The bee hive door is closed.
    a flower pot 1 (containing a apple tree in the adult stage, soil)
    a flower pot 2 (containing a peach tree in the adult stage, soil)
    a flower pot 6 (containing a banana tree in the adult stage, soil)
    a flower pot 7 (containing a substance called water)
    a flower pot 8 (containing a substance called water)
    a jug (containing nothing)
    a shovel
    a sink, which is turned off. In the sink is: nothing.
You also see:
    A door to the hallway (that is open)
    A door to the outside (that is open) [score=17]

> focus on peach tree
You focus on the peach tree. [score=67]

> pick up flower pot 2
You move the flower pot 2 to the inventory. [score=75]

> teleport to bedroom
You teleport to the bedroom. [score=83]

> move flower pot 2 containing peach tree and soil in inventory to yellow box
You move the flower pot 2 to the yellow box. [score=100]
```

Figure 8: Example of a successful trajectory in ScienceWorld completed by IAM-FA + DA.

F Examples of template-based and human-annotated goals

The human-annotated goals include 66 unseen verbs and 189 unseen nouns that are not present in the template-based goals (Shridhar et al., 2021). Table 12 shows examples for each task type.

Task Type	Example
Pick & Place	(a) put some spraybottle on toilet. (b) put the spray bottle on the back of the toilet.
Examine in Light	(a) examine the statue with the desklamp. (b) Pick up the statue from the table and turn on the lamp.
Clean & Place	(a) clean some dishsponge and put it in countertop. (b) Put a wet sponge on the counter.
Heat & Place	(a) heat some apple and put it in diningtable. (b) Put a heated apple on the table.
Cool & Place	(a) put a cool egg in microwave. (b) Put a cold egg in the microwave.
Pick Two & Place	(a) find two knife and put them in drawer. (b) place two knifes in the kitchen drawer.

Table 12: Examples of (a) template-based and (b) human-annotated goals for each task type.