Meta-Reinforcement Learning for Mastering Multiple Skills and Generalizing across Environments in Text-based Games

Zhenjie Zhao
Nanjing University of Information Science and Technology
zzhaoao@nuist.edu.cn

Mingfei Sun
University of Oxford
mingfei.sun@cs.ox.ac.uk

Xiaojuan Ma
The Hong Kong University of Scinece and Technology
mxj@cse.ust.hk

Abstract

Text-based games can be used to develop task-oriented text agents for accomplishing tasks with high-level language instructions, which has potential applications in domains such as human-robot interaction. Given a text instruction, reinforcement learning is commonly used to train agents to complete the intended task owing to its convenience of learning policies automatically. However, because of the large space of combinatorial text actions, learning a policy network that generates an action word by word with reinforcement learning is challenging. Recent research works show that imitation learning provides an effective way of training a generation-based policy network. However, trained agents with imitation learning are hard to master a wide spectrum of task types or skills, and it is also difficult for them to generalize to new environments. In this paper, we propose a meta-reinforcement learning based method to train text agents through learning-to-explore. In particular, the text agent first explores the environment to gather task-specific information and then adapts the execution policy for solving the task with this information. On the publicly available testbed ALFWorld, we conducted a comparison study with imitation learning and show the superiority of our method.

1 Introduction

A text-based game, such as Zork (Infocom, 1980), is a text-based simulation environment that a player uses text commands to interact with. For example, given the current text description of a game environment, users need to change the environmental state by inputting a text action, and the environment returns a text description of the next environmental state. Users have to take text actions to change the environmental state iteratively until an expected final state is achieved (Côté et al., 2018). Solving text-based games requires non-trivial natural language understanding/generalization and sequential decision making. Developing agents that can play text-based games automatically is promising for enabling task-oriented, language-based human-robot interaction (HRI) experience (Scheutz et al., 2011). Supposing that a text agent can reason a given command and generate a sequence of text actions for accomplishing the task, we can then use text as a proxy and connect text inputs and outputs of the agent with multi-modal signals, such as vision and physical actions, to allow a physical robot operate in the physical space (Shridhar et al., 2021).

Given a text instruction or goal, reinforcement learning (RL) (Sutton and Barto, 2018) is commonly used to train agents to finish the intended task automatically. In general, there are two approaches to train a policy network to obtain the corresponding text action: generation-based methods that generate a text action word by word and choice-based methods that select the optimal action from a list of candidates (Côté et al., 2018). The list of action candidates in a choice-based method may be limited by pre-defined rules and hard to generalize to a new environment. In contrast, generation-based methods can generate more possibilities and potentially have a better generalization ability. Therefore, to allow a text agent to fully explore in an environment and obtain best performance, a generation-based method is needed (Yao et al., 2020). However, the combinatorial action space precludes reinforcement learning from working well on a generation-based policy network. Recent research shows that imitation learning (Ross et al., 2011) provides an effective way to train a generation-based policy network using demonstrations or dense reinforcement signals (Shridhar et al., 2021). However, it is still difficult for the trained policy to master multiple task types or skills and generalize across environments (Shridhar et al., 2021). For example, an agent trained on the task type of *slicing an apple* cannot work on a task of *pouring water*. Such lack of the ability to generalize precludes the agent from working on a real interaction scenario. To achieve real-world HRI experience with text agents, two requirements should be fulfilled: 1) a trained agent should master multiple skills simultaneously and work on any task type that it has seen during training; 2) a trained agent should also generalize to unseen environments.

Meta-reinforcement learning (meta-RL) is a commonly used technique to train an agent that generalizes across multiple tasks through summarizing experience over those tasks. The underlying idea of meta-RL is to incorporate meta-learning into reinforcement learning training, such that the trained agent, *e.g.*, text-based agents, could master multiple skills and generalize across different environments (Finn et al., 2017; Liu et al., 2020). In this paper, we propose a meta-RL based method to train text agents through learning-to-explore. In particular, a text agent first explores an environment to gather task-specific information. It then updates the agent's policy towards solving the task with this task-specific information for better generalization performance. On a publicly available testbed, ALFWorld (Shridhar et al., 2021), we conducted experiments on all its six task types (*i.e.*, *pick & place*, *examine in light*, *clean & place*, *heat & place*, *cool & place*, and *pick two & place*), where for each task type, there is a set of unique environments sampled from the distribution defined by their task type (see Section 5.1 for statistics). Results suggest that our method generally masters multiple skills and enables better generalization performance on new environments compared to ALFWorld (Shridhar et al., 2021). We provide further analysis and discussion to show the importance of task diversity for meta-RL. The contributions of this paper are:

- From the perspective of human-robot interaction, we identify the generalization problem of training an agent to master multiple skills and generalize on new environments. We propose to use meta-RL methods to achieve it.

- We design an efficient learning-to-explore approach which enables a generation-based agent to master multiple skills and generalize across a wide spectrum of environments.

2 Related Work

2.1 Language-based Human-Robot Interaction

Enabling a robot to accomplish tasks with language goals is a long-term study of human-robot interaction (Scheutz et al., 2011), where the core problem is to ground language goals with multi-modal signals and generate an action sequence for the robot to accomplish the task. Because of the characteristic of sequential decision making, reinforcement learning (Sutton and Barto, 2018) is commonly used. Previous research works using reinforcement learning have studied the problem on simplified block worlds (Janner et al., 2018; Bisk et al., 2018), which could be far from being realistic. The recent interests on embodied artificial intelligence (embodied AI) have contributed to several realistic simulation environments, such as Gibson (Xia et al., 2018), Habitat (Savva et al., 2019), RoboTHOR (Deitke et al., 2020), and ALFRED (Shridhar et al., 2020). However, because of physical constraints in a real environment, gap between a simulation environment and a real world still exists (Deitke et al., 2020; Shridhar et al., 2021). Researchers have also explored the idea of finding a mapping between vision signals of a real robot and language signals directly (Blukis et al., 2020), but this mapping requires detailed annotated data and it is usually expensive to obtain physical interaction data. An alternative method of deploying an agent on a real robot is to train the agent on abstract text space, such as TextWorld (Côté et al., 2018), and then connect text with multi-modal signals of the robot (Shridhar et al., 2021). For example, by connecting text with the simulated environment ALFRED (Shridhar et al., 2020), researchers have shown that the trained text agent has better generalization ability than training an embodied agent end-to-end directly (Shridhar et al., 2021). However, how to make a text agent generalize across different tasks so that one robot can work on tasks of different types and in unseen environments is still a challenging problem, which is the focus of this paper.

2.2 Text-based Games

The success of deep reinforcement learning (RL) on Atari games (Mnih et al., 2015) inspires the use of RL on text-based games. There are a variety of ways to use deep reinforcement learning on text-based games. For example, using the deep Q-learning (DQN) framework, Narasimhan et al. (2015) leverage the Long Short-Term Memory (LSTM) as the policy network to predict action for each state. In (He et al., 2016), researchers propose the deep reinforcement relevance network (DRRN), which encodes states and actions separately and then calculates Q-values by integrating the information of the two channels. However, the compositional and combinatorial properties of natural language lead to large state and action spaces, which makes solving text-based games with deep reinforcement learning very challenging. To deal with this problem, in fiction-style text games, Adhikari et al. (2020) use a graph-aided transformer (GATA) to capture game dynamics so that it can plan well and select text actions more effectively. Ammanabrolu and Riedl (2019) learn a knowledge graph during the exploration of an agent, and use it to prune the action space. Furthermore, Murugesan et al. (2021) show that incorporating common sense knowledge also helps reduce the action space and allows an agent to choose an action more effectively. Recently, Yao et al. (2020) show that given a text state, a fine-tuned language model GPT can generate a corresponding text action set, which significantly reduces the action space and also improves the performance. Previous research works mainly focus on learning an agent to solve one text game effectively. However, in reality, we usually hope an agent can learn a wide spectrum of tasks and generalize well to unseen environments. In (Adolphs and Hofmann, 2020), in terms of environments and task descriptions, researchers show that an actor-critic framework with action space pruning can learn an agent to generalize to unseen games that belongs to the same family when training. In this paper, with meta-reinforcement learning, we investigate if an agent can master multiple task types and generalize to unseen environments.

2.3 Meta-reinforcement Learning

Meta-learning is a machine learning paradigm that tries to leverage common knowledge among tasks to generalize to new data (Thrun and Pratt, 1998; Vilalta and Drissi, 2002). Meta-reinforcement learning, in particular, augments Markov decision processes with particular task labels, and tries to use shared experience of interacting with different tasks to adapt to a new task efficiently (Liu et al., 2020). In general, there are three ways of conducting meta-reinforcement learning: memory-based methods, optimization-based methods, and learning-to-explore. For memory-based methods, researchers have proposed RL^2 (Duan et al., 2016), which uses a recurrent neural network (RNN) to encode a "fast" RL algorithm, and the RNN module is trained with another "slow" RL algorithm. Memory-based methods are usually hard to optimize and suffer from the sample efficiency problem (Duan et al., 2016). For optimization-based methods, in (Finn et al., 2017), researchers propose a model-agnostic meta-reinforcement learning algorithm that uses a nested optimization procedure to obtain maximal rewards with limited number of sample trajectories. Optimization-based methods usually require on-policy reinforcement learning algorithms and are hard to use value-based methods (Finn et al., 2017), which also leads to the sample efficiency problem. Learning-to-explore is a newly proposed meta-reinforcement learning approach that can potentially leverage any reinforcement learning method with good optimization properties by decoupling an episode into two stages: exploration and execution (Rakelly et al., 2019; Liu et al., 2020). The exploration stage is used to recognize task-specific information, which could be useful for the execution stage for fast and efficient adaptation.

For embodied AI, using meta-reinforcement learning, researchers have explored to improve generalization ability of an agent to unseen environments (Wortsman et al., 2019). However, as aforementioned, deploying such an agent on a real robot is still a challenging problem owing to the domain gap between a simulation environment and a physical environment. In this paper, we instead try to use the learning-to-explore method of meta-reinforcement learning to increase the generalization ability of a text agent so that it can master multiple skills and work on new environments, which can potentially facilitate real-world human-robot interaction applications.

3 Problem Formulation

3.1 Text-based Game Preliminary

Given a language goal g, playing a text-based game can be modeled as a partially observable Markov decision process (POMDP) $(S, P, A, \Omega, O, R, \gamma)$ (Côté et al., 2018), where S is the set of environmental states, P is the set of transition probabilities, A is the set of actions, Ω is the set of observations, O is the set of observation probabilities, R is the reward function, and γ is the discount factor. If we input an action a_t to the environment, it will transition from the current state s_t to a new state s_{t+1} with probability $P(s_{t+1}|s_t, a_t)$, output an observation o_{t+1} based on the new state with probability $O(o_{t+1}|s_{t+1})$, and get a reward $R(g, a_t, s_t)$ depending on the goal g, the current action a_t, and the current state s_t. Given the initial environment state s_0 and a goal g, we want to learn a policy $\pi(a|o, g)$ that can generate an action sequence $(a_0, a_1, \ldots, a_T)$ to accomplish the task and obtain maximal discounted reward $\sum_{t=0}^{T} \gamma^t R(g, a_t, s_t)$. In text-based games, o and a refer to text sentences.

3.2 Learning-to-Explore in Text-based Games

In meta-reinforcement learning, we consider a family of POMDPs $\{(S_\mu, A_\mu, \Omega_\mu, \gamma, O_\mu, R, P_\mu)\}$ indexed by μ, where $\mu \in \mathcal{M}$ denotes a task and $\mathcal{M}$ denotes the family of POMDPs or tasks. Here, we consider that the reward function is independent of tasks and can be applied for all POMDPs. The tasks in the family have task-dependent set of states S_μ, actions A_μ, observations Ω_μ, observation probabilities O_μ, and dynamics P_μ. Following the setting in (Liu et al., 2020), given a goal g, a task-based meta-reinforcement learning problem consists of sampling a task $\mu \sim p(\mu)$ and running a trial, where a trial contains an exploration episode, followed by several execution episodes. We also call a goal as a task type or a skill because it usually constrains how an agent solves a task μ. We call a POMDP without the reward function as an environment, contextualized with the task specifier μ, since it defines a game environment that an agent can interact with. A task denoted by μ then contains a task type, an environment, and a reward function. Given a set of training tasks $\mathcal{M}_{train}$, we want to train a policy $\pi(a|o, g)$ that can generalize well across a set of testing tasks $\mathcal{M}_{test}$. For training, we first fit a task-specific feature vector z'_μ using the exploration episode, and then use it to adapt to the task quickly during execution. The task-specific adaptation helps the policy π to recognize which task type it works on and generalize well on a new unseen environment.

4 Method

We use neural networks to map observations to actions. Given the general setting of meta-reinforcement learning through learning-to-explore, our method contains three modules: an execution

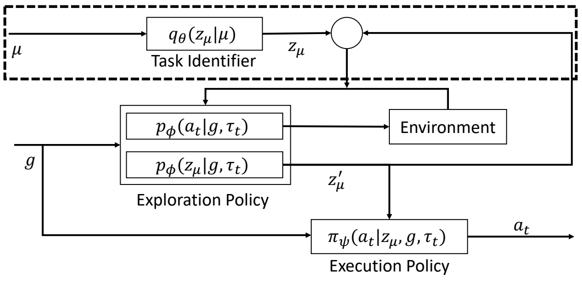

Figure 1: Overview of our method, where g is the language goal, μ denotes a task index, z_μ and z'_μ are hidden feature vectors of a task, and a_t is the generated text action. The dotted line box is only used during training. For simplicity, we did not draw the inputs of roll-out trajectories.

policy neural network $\pi_{\boldsymbol{\psi}}$, a task identifier neural network $q_{\boldsymbol{\theta}}$, and an exploration policy neural network $p_{\boldsymbol{\phi}}$, where $\boldsymbol{\psi}$, $\boldsymbol{\theta}$, and $\boldsymbol{\phi}$ denote parameters of the three neural networks, respectively. As shown in Figure 1, an exploration policy $p_{\boldsymbol{\phi}}$ is trained to generate a task-specific feature vector z'_μ, which is then input to an execution policy $\pi_{\boldsymbol{\psi}}$ for generating actions. During training, a task identifier is used to generate supervised signals z_μ of z'_μ, and is not used during testing. Because of z'_μ, $\pi_{\boldsymbol{\psi}}$ can adapt quickly and generalize well in a new task.

The $\pi_{\boldsymbol{\psi}}$, $q_{\boldsymbol{\theta}}$ and $p_{\boldsymbol{\phi}}$ are all encoder-decoder architectures. For $\pi_{\boldsymbol{\psi}}$, it takes a goal g and a K-step roll-out trajectory $\tau_t = (o_0, a_{t-K}, o_{t-K+1}, \ldots, a_{t-1}, o_t)$ from time $t-K+1$ to time t as inputs, and outputs the current action a_t, where o_0 is obtained by executing the "look" action at the beginning. o_0 is used because it is the only observation that lists the different areas of the room. $q_{\boldsymbol{\theta}}$ takes a task index μ as an input and outputs the task-specific feature z_μ, which is only used during training. $p_{\boldsymbol{\phi}}$ takes a goal and a K-step roll-out trajectory $\tau_t = (o_0, a_{t-K}, o_{t-K+1}, \ldots, a_{t-1}, o_t)$ as inputs and outputs an estimated task-specific feature z'_μ.

Our goal is to make an execution policy $\pi(a_t|g, \tau_t)$ generalizable across tasks. If we train π using imitation learning, it is critical to have enough training samples of $\{(g, \tau_t, a_t)\}$ following some distributions to have good generalization performance. But because of the combinatorial complexity of τ_t, it is hard to obtain enough data of a_t. Learning from conditional variational auto-encoder (CVAE) (Sohn et al., 2015), we factorize π with a task-specific hidden variable z and use z to facilitate the generation of a_t, namely,

$$\pi(a_t|g,\tau_t) = \int_{z\in Z} p(z|g,\tau_t)\pi(a_t|z,g,\tau_t)dz, \tag{1}$$

where $Z \sim \mathcal{N}(z_\mu, \sigma^2 I)$ is assumed to follow a Gaussian distribution, the aforementioned task-specific feature vector z_μ is the mean vector and σ^2 is the variance. During testing, we can then generate actions by first generating a task-specific hidden variable z with $p(z|g,\tau_t)$ and then generating the action with $\pi(a_t|z,g,\tau_t)$. Because z encodes task-specific features, it helps π generate more proper actions for the current task μ.

Optimizing (1) amounts to maximise evidence lower bound (ELBO) (Sohn et al., 2015):

$$\begin{aligned}\text{ELBO}(a_t,g,\tau_t) = \mathbb{E}_{q(z|a_t,g,\tau_t)}[\log \pi(a_t|z,g,\tau_t)] \\ - \text{KL}(q(z|a_t,g,\tau_t))||p(z|g,\tau_t)),\end{aligned} \tag{2}$$

where $q(z|a_t,g,\tau_t))$ is the approximate posterior probability of z and $p(z|g,\tau_t)$ is the prior probability of z. To implement (2), we use the execution policy network $\pi_{\boldsymbol{\psi}}(a_t|z_\mu,\tau_t)$, the task identifier $q_{\boldsymbol{\theta}}(z_\mu|\mu)$, and the exploration policy network $p_{\boldsymbol{\phi}}(z'_\mu|g,\tau_t)$ to approximate the execution policy, the posterior, and the prior, respectively, and assume that both $q_{\boldsymbol{\theta}}(z_\mu|\mu)$ and $p_{\boldsymbol{\phi}}(z'_\mu|g,\tau_t)$ are Gaussian. It is easy to show that the new objective is:

$$\mathbb{E}_{q_{\boldsymbol{\theta}}(z_\mu|a_t,g,\tau_t)}[\log \pi_{\boldsymbol{\psi}}(a_t|z_\mu,g,\tau_t)] - \frac{||z_\mu - z'_\mu||_2^2}{2\sigma^2}, \tag{3}$$

where we assume σ^2 is the same for both the posterior and prior. In the following, we introduce the details of the execution policy network, the task identifier, and the exploration policy network.

4.1 Execution Policy

The architecture of the execution policy network is similar to the policy network in (Shridhar et al., 2021). In particular, a QANet (Yu et al., 2018) is used to first encode g, τ_t as a recurrent hidden state h_t and then decode h_t to get a_t. Different from (Shridhar et al., 2021), during encoding, we concatenate the initial encoding h_{RNN} and z_μ as an input to obtain h_t, namely,

$$\begin{aligned}h_{RNN} &= \text{Encode}(g,\tau_t),\\ h_t &= \text{GRU}(\text{ReLU}(\mathbf{W}(h_{RNN}\oplus z_\mu)+\mathbf{b}), h_{t-1}),\end{aligned}$$

where $\oplus$ denotes the concatenation operation, $\mathbf{W} \in R^{d_e\times 2d_e}$ is a weight matrix, $\mathbf{b} \in R^{d_e}$ is a bias vector, $h_{RNN} \in R^{d_e}$, $h_t \in R^{d_h}$, d_e is the dimension of z_μ, d_h is the dimension of h_t, GRU denotes a gated recurrent unit, and ReLU denotes a ReLU activation function. Compared to selecting text actions from a set of valid actions, generating text actions word by word is more likely to explore multiple possibilities for performing actions to achieve higher rewards (Yao et al., 2020). However, Shridhar et al. (2021) show that when trained from a sparse reinforcement learning signal in ALFWorld, generation-based methods are hard to get good performance. Because it is relatively easy to get demonstrations from a text-based game, similar to (Shridhar et al., 2021), the imitation learning method DAgger (Ross et al., 2011) is used to train a generation-based execution policy $\pi_{\boldsymbol{\psi}}$. In this case, optimizing the execution policy network is to optimize the first term of (3).

4.2 Task Identifier

We use a task identifier $q_{\boldsymbol{\theta}}(z_\mu|\mu)$ to approximate the approximate posterior $q(z|a_t,g,\tau_t)$. The task identifier is used to generate task-specific features during training. We implement it as a simple two-layer fully connected network as:

$$z_\mu = \text{ReLU}(\mathbf{W}_2\text{ReLU}(\mathbf{W}_1\mathbf{e}(\mu)+\mathbf{b}_1)+\mathbf{b}_2),$$

where $\mathbf{e}(\mu)$ is the one-hot encoding of the task index μ, $\mathbf{W}_2 \in R^{d_e\times d_e}$, $\mathbf{W}_1 \in R^{d_e\times N}$, $\mathbf{b_1}, \mathbf{b_2} \in R^{d_e}$, d_e is the dimension of the task embedding z_μ, N is the number of training game environments.

4.3 Exploration Policy

We use an exploration policy network $p_{\boldsymbol{\phi}}(z'_\mu|g,\tau_t)$ to approximate the prior $p(z|g,\tau_t)$. The exploration policy needs to explore the environment to gather task-specific trajectory within T^{exp}. Because we train the model end-to-end, it will optimize the agent to explore the environment in this fixed number of steps, which also saves time. The architecture is similar to the execution policy network. An encoder takes g, τ_t as inputs and generates a hidden state h_t, and the hidden state is then used to obtain z'_μ via a fully connected layer:

$$z'_\mu = \text{ReLU}(\mathbf{W}h_t+\mathbf{b}),$$

where $\mathbf{W} \in R^{d_e\times d_h}$, $\mathbf{b} \in R^{d_e}$.

Algorithm 1: The training procedure

```
Input: training tasks $\mathcal{M}_{train}$
Output: execution policies $\pi_\psi$, exploration policy $p_\phi$

initialize hyper-parameters $M_{step}, B, T^{exp}, T^{exec}$
initialize $\pi_\psi$, $p_\phi$, and $q_\phi$

$i \leftarrow 0$
while True do
    if $i > M_{step}$ then
        break
    end

    randomly sample $B$ games $\mathcal{M}_B$ from $\mathcal{M}_{train}$

    // Evaluate the task identifier
    calculate $z_\mu$ with $q_\phi$

    // Exploration
    execute "look" and get $o_0$
    for $t$=1:$T^{exp}$ do
        $a_t \leftarrow p_\phi(a_t|g, \tau_t)$
        compose $\tau_t$ by adding $a_t$ and $o_t$
        evaluate $\mathcal{M}_B$ with $a_t$ and get $o_{t+1}$
        $z'_\mu \leftarrow p_\phi(z'_\mu|g, \tau_t)$
        calculate (4) and update
    end

    // Execution
    execute "look" and get $o_0$
    for $t$=1:$T^{exec}$ do
        $a_t \leftarrow \pi_\psi(a_t|z_\mu, g, \tau_t)$
        compose $\tau_t$ by adding $a_t$ and $o_t$
        get demonstrations from $\mathcal{M}_B$
        calculate likelihood of $a_t$ using
          demonstrations and update
        if done then
            break
        end
    end

    $i \leftarrow i + B$
end
```

For the exploration policy, in addition to obtain z'_μ, we also decode h_t to get an exploration action a_t: $p_\phi(a_t|g, \tau_t)$. In other words, we adopt a multi-task learning method to train the exploration network. In this way, the exploration policy also learns how to solve the problem, which could help the learning of z'_μ. We optimize the following multi-task objective:

$$\mathcal{L} = \mathcal{L}_\mu + \mathcal{L}_{dqn}, \tag{4}$$

where $\mathcal{L}_\mu$ is the task embedding loss and $\mathcal{L}_{dqn}$ is the DQN loss. In particular, $\mathcal{L}_\mu$ is the second term in (3), except that we do not consider the coefficient $1/2\sigma^2$. For the DQN loss $\mathcal{L}_{dqn}$, we use the deep Q-learning (DQN) method to train the exploration policy. Unlike the execution policy network, we do not use demonstrations here because we want the policy network to *explore* the environment more. DQN is an off-policy method that can leverage replay buffer to deal with the sample efficiency problem. Here, we use DQN for its simplicity, but it is possible to use other more sophisticated off-policy methods. Because it is generally difficult to train a generation-based text agent with only the sparse rewards provided by the environment, we adopt the choice-based method to train the text agent. We empirically turn the reward function to be dense by adding the second term in Eq(3) to the reward function: $R_{new} = 0.5 \times R_{old} + 0.5 \times ||z_\mu - z'_\mu||_2^2$ to encourage per-step optimization, where R_{old} is the reward provided by the environment.

The training procedure of the proposed method, as presented in Algorithm 1, runs as follows: first, we randomly samples a batch of tasks $\mathcal{M}_B$ from $\mathcal{M}_{train}$; second, with task indices, we evaluates $q_{\boldsymbol{\theta}}$ to obtain the task-specific features z_μ; third, the exploration agent explores $\mathcal{M}_B$ by taking actions with p_ϕ, and updates p_ϕ according to Eq(4) through a DQN learning. z'_μ is also obtained by p_ϕ during exploring; fourth, the execution agent takes actions with π_ψ and we update the likelihood (the first term in Eq(3)) with demonstrations of the training data. The end-to-end training runs iteratively up to a maximal step M_{step}. In Algorithm 1, B denotes the sampling size of tasks, T^{exp} is the step number of exploration, and T^{exec} is the step number of execution.

5 Experiments

To demonstrate the generalization ability of our meta-reinforcement learning algorithm across tasks, we conducted a set of experiments with the ALFWorld platform (Shridhar et al., 2021). Text environments of ALFWorld are aligned with 3D simulated environments from ALFRED (Shridhar et al., 2020), which makes ALFWorld a good proxy for our human-robot interaction scenario.

5.1 Dataset

The ALFWorld dataset (Shridhar et al., 2021) contains six task types, including *pick & place, examine in light, clean & place, heat & place, cool & place, and pick two & place*. While all the task types require some basic common sub-tasks such as finding an object, picking it up, and placing it to a particular place; some task types require more complex interactions with certain objects (*e.g.*, heating an object with a heat source). Each task type contains a set of training environments, and two sets of test environments. The first test set (**seen**) contains environments that are different, but sampled from the same game distributions as the training set (*e.g.*, same rooms but with different scene layouts). The second test set (**unseen**) contains environments that do not appear in the training set (*i.e.*, unseen rooms with different receptacles and scene layouts). The statistics of the dataset is shown in Table 1. The task types *pick & place* and *pick two & place* have more training environments than others. Our generalization goal is to train a text agent on the training set of all tasks simultaneously, and during testing, given any task type, the agent can have good performance on both seen and unseen environments, *i.e.*, the agent masters all the six task types and generalizes well on both seen and unseen environments.

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 tasks	3553	140	134

Table 1: The statistics of the ALFWorld dataset.

5.2 Baseline and Implementation Details

We compare our method (denoted as Ours) with the state-of-the-art generation-based agent (denoted as ALFWorld). Transfer learning is another way to improve the generalization ability of an agent, but it usually considers transferring knowledge from a source task to a target task without the setting of multiple tasks (Zhuang et al., 2021). We leave it as a future direction to investigate. We adopt the implementation of ALFWorld from the original paper (Shridhar et al., 2021) and use their pre-trained model for conducting all comparison experiments. For the hyper-parameters in Algorithm 1, T^{exp} is set as 10 empirically, $M_{step} = 500,000$ (50K), $B = 10$, $T^{exec} = 50$ are kept as the default values of ALFWorld. The trajectory length K is set as 3 empirically. Following ALFWorld (Shridhar et al., 2021), we use beam search with width 10 for decoding. We ran all experiments on a server with Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz, 32G Memory, Nvidia GPU 2080Ti, Ubuntu 16.04.

5.3 Evaluation Metric

We use success rate as the evaluation metric for our experiment. In particular, for $|\mathcal{M}_{test}|$ text games being evaluated, if an agent can finish S tasks, then the success rate of the agent is $sr = \frac{S}{|\mathcal{M}_{test}|}$. Similar to (Shridhar et al., 2021), we evaluate three times on the testing data and report averaged scores.

	ALFWorld		**Ours**	
task type	**seen**	**unseen**	**seen**	**unseen**
pick & place	46.7	34.7	**51.4**	**50.0**
examine in light	25.7	**22.2**	**38.5**	**22.2**
clean & place	44.4	39.8	**48.1**	**54.8**
heat & place	**58.3**	44.9	50.0	**56.5**
cool & place	38.7	47.6	**44.0**	**76.2**
pick two & place	**23.6**	**27.4**	12.5	23.5
all tasks	39.3	37.6	**41.4**	**49.3**

Table 2: Experiment results of the generalization ability on each individual task type and the union of them.

5.4 Results and Analysis

We show the performance of our model on both seen and unseen test sets in Table 2, compared with numbers computed using the code and model checkpoint provided by ALFWorld (Shridhar et al., 2021). We observe that in most experiment settings, our method outperforms ALFWorld. This is especially obvious in the unseen setting, where the testing environments contain unseen rooms with different receptacles and scene layouts, our method outperforms ALFWorld by a significant margin. This suggests that the task-specific features generated by our agent indeed enable the agent learning from a wide spectrum of task types. The larger performance gap between our method and ALFWorld on the unseen test set (*e.g.*, 49.3 *vs* 37.6 when testing on the union of all task types) further advocates that the task-specific features generated by our method are useful when tackling with completely unfamiliar environments.

On the other hand, we observe that our method's performance on the *pick two & place* tasks are lower than ALFWorld. As mentioned in (Shridhar et al., 2021), the *pick two & place* task type is unique and is considerably more difficult compared to other tasks, in the sense that it is the only task type which requires an agent to grasp and operate more than one object. Intuitively, this aligns with the common sense that a person who has learned to ride all kinds of bicycles can easily ride a new bicycle, but does not necessarily know how to drive a car. We suspect that the decrease in performance may be caused by the agent being overfitting to the majority of training data in which only single object is picked up. Namely, the current developed method could work better on scenarios where a text-based game has the same difficulty level. In other words, the current developed method can only work on scenarios where a text-based game has the same difficulty level as the majority of training games, and it is still hard to generalize to tasks with a higher difficulty level. As a future direction, we plan to investigate the explainability of why an end-to-end trained agent works on certain tasks through counterfactuals (Pearl and Mackenzie, 2018), and improve our method to specifically tackle such problems where a certain dimension of task representations is significantly different from and unbalanced in the majority of training data.

Finally, compared to a dedicated model trained specifically on one task type (Table 2 left in (Shridhar et al., 2021)), the performance of our method is generally $10\% \sim 20\%$ behind, and there is still a lot of room for improvement to achieve human-level intelligence. However, our method shows that learning task-specific features through meta-reinforcement learning help an agent generalize across a wide spectrum of task types, which is vital towards real-world applications of human-robot interaction.

5.5 Discussion

To investigate whether different task types help improve performance of each other, we experimented with a setting where an agent is trained on the six task types separately with our method. The results are shown in Table 3. Compared to the setting where the agent is trained on the union of all task types (Table 2), the performance shows a significant drop in most of the task types. This trend is especially clear in the *pick two & place* tasks. When trained solely on this type of tasks, our agent produces a zero success rate. This suggests that for a meta-reinforcement learning based method like ours, it is essential to have a diverse set of task types as well as a large enough training dataset.

task type	**seen**	**unseen**
pick & place	57.1	25.0
examine in light	23.1	11.1
clean & place	51.9	58.1
heat & place	31.3	30.4
cool & place	12.0	9.5
pick two & place	0.0	0.0

Table 3: Testing results of training a separate agent on each of the six task types.

6 Conclusion

We study the generalization issue of text-based games, and develop a meta-reinforcement learning method with a learning-to-explore approach. In particular, we first use an exploration policy network to learn a task-specific feature vector, and use this feature vector to help another execution policy network adapt to a new task. To train the exploration and execution policy network, we use a task identifier to embed a task index, and maximize the likelihood of the execution policy network end-to-end. To demonstrate the generalization ability of our method, we conducted a set of experiments on the publicly available testbed ALFWorld. In general, we find that our method has better generalization performance on a wide spectrum of task types and environments. We leave the investigation of explanability, the unbalance problem of task types, and the training speed as the future research directions.

Acknowledgments

The authors thank Mohit Shridhar for clarifying the experiment section in ALFWorld, Xingdi (Eric) Yuan and Marc-Alexandre Côté for their insightful discussions and editing of the initial version of this paper, as well as constructive suggestions from all anonymous reviewers. Zhenjie Zhao is supported by the Startup Foundation for Introducing Talent of NUIST. This work is also supported by the Hong Kong General Research Fund (GRF) with grant No. 16204420.

References

Ashutosh Adhikari, Xingdi Yuan, Marc-Alexandre Côté, Mikuláš Zelinka, Marc-Antoine Rondeau, Romain Laroche, Pascal Poupart, Jian Tang, Adam Trischler, and Will Hamilton. 2020. Learning dynamic belief graphs to generalize on text-based games. In *Advances in Neural Information Processing Systems*, volume 33, pages 3045–3057. Curran Associates, Inc.

Leonard Adolphs and Thomas Hofmann. 2020. LeDeepChef deep reinforcement learning agent for families of text-based games. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7342–7349.

Prithviraj Ammanabrolu and Mark Riedl. 2019. Playing text-adventure games with graph-based deep reinforcement learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies, Volume 1 (Long and Short Papers), pages 3557–3565, Minneapolis, Minnesota. Association for Computational Linguistics.

Yonatan Bisk, K. Shih, Yejin Choi, and Daniel Marcu. 2018. Learning interpretable spatial operations in a rich 3D blocks world. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Valts Blukis, Ross A. Knepper, and Yoav Artzi. 2020. Few-shot object grounding and mapping for natural language robot instruction following. In *Proceedings of the Conference on Robot Learning*.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, J. Moore, Matthew J. Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. Textworld: A learning environment for text-based games. In *Computer Games Workshop at ICML/IJCAI 2018*.

Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. 2020. RoboTHOR: An open simulation-to-real embodied AI platform. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. 2016. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv:1611.02779*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1621–1630, Berlin, Germany. Association for Computational Linguistics.

Infocom. 1980. Zork I. http://ifdb.tads.org/viewgame?id=0dbnusxunq7fw5ro.

Michael Janner, Karthik Narasimhan, and Regina Barzilay. 2018. Representation learning for grounded spatial reasoning. *Transactions of the Association for Computational Linguistics*, 6:49–61.

Evan Zheran Liu, Aditi Raghunathan, Percy Liang, and Chelsea Finn. 2020. Explore then execute: Adapting without rewards via factorized meta-reinforcement learning. *arXiv:2008.02790*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Pushkar Shukla, Sadhana Kumaravel, Gerald Tesauro, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell. 2021. Text-based RL agents with commonsense knowledge: New challenges, environments and baselines. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):9018–9027.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Lisbon, Portugal. Association for Computational Linguistics.

Judea Pearl and Dana Mackenzie. 2018. *The book of why*. Basic Books, New York.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. 2019. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5331–5340. PMLR.

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.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. 2019. Habitat: A platform for embodied AI research. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9338–9346.

Matthias Scheutz, Rehj Cantrell, and Paul Schermerhorn. 2011. Toward humanlike task-based dialogue processing for human robot interaction. *AI Magazine*, 32(4):77–84.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10740–10749.

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)*.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

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

Sebastian Thrun and Lorien Pratt, editors. 1998. *Learning to learn*. Kluwer Academic Publishers.

Ricardo Vilalta and Youssef Drissi. 2002. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95.

Mitchell Wortsman, Kiana Ehsani, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. 2019. Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6750–6759.

Fei Xia, Amir R. Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. 2018. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9068–9079.

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.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018. Fast and accurate reading comprehension by combining self-attention and convolution. In *International Conference on Learning Representations*.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. 2021. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76.