Language Model Adaption for Reinforcement Learning with Natural Language Action Space

Jiangxing Wang[1] **Jiachen Li**[1] **Xiao Han**[1] **Deheng Ye**[2] **Zongqing Lu**[1,3†]
[1]School of Computer Science, Peking University
[2]Tencent Inc.
[3]BAAI
{jiangxiw,lijiachen,hanx}@stu.pku.edu.cn dericye@tencent.com
zongqing.lu@pku.edu.cn

Abstract

Reinforcement learning with natural language action space often suffers from the curse of dimensionality due to the combinatorial nature of the natural language. Previous research leverages pretrained language models to capture action semantics and reduce the size of the action space. However, since pretrained models are typically trained on general corpora, there can be an unpredictable mismatch between the priors encoded in pretrained models and the characteristics of the specific RL environment. To address this issue, we propose Mutual-Information Regularized Policy Optimization, MIPO. MIPO enables implicit and dynamic reduction of the action space. Starting from the prior provided by the pretrained language model, our method dynamically adjusts the prior during the learning process based on the guidance of mutual information regularization. Theoretically, we demonstrate that this policy optimization process leads to the monotonic improvement on the mutual-information regularized RL objective. Empirically, we conduct experiments in various environments and demonstrate the effectiveness of MIPO.

1 Introduction

Deep reinforcement learning (RL) has gained great success from mastering the game of Go (Silver et al., 2017) to training state-of-the-art large language models (Ouyang et al., 2022). Despite its success in various domains, the low sample efficiency of RL has consistently hindered its wide adoption in real-world applications where the cost of interaction is not negligible (Li, 2017). Furthermore, as suggested in Azar et al. (2017), the sample complexity of RL algorithms will be inevitably influenced by the size of the action space. This issue becomes even more pronounced in scenarios with enormous action spaces, such as multi-agent reinforcement learning (Zhang et al., 2021) and open-world reinforcement learning (Yuan et al., 2023a), exacerbating the problem of sample efficiency.

One natural solution to tackle the challenges of large action space is to eliminate irrelevant actions accordingly to avoid the exploration of unnecessary state-action pairs. While a general principle of action space reduction is hard to find, for reinforcement learning problems with natural language action space (He et al., 2016), leveraging action semantics provides a means to infer the potential usefulness of actions. Previous work has explored the use of pretrained language models for action space reduction (Martin et al., 2022). Despite being a viable solution, pretrained models encode prior knowledge derived from general corpora, which may differ from the specifics of the actual environment. To alleviate such a mismatch, some studies (Yao et al., 2020; Xu et al., 2022) utilize the domain-specific dataset to finetune the pretrained model for better adaptation. Others (Zahavy et al., 2018; Ammanabrolu and Hausknecht, 2019; Jain et al., 2020) assume the feedback of action admissibility from the environment, which assesses the relevance of each action to the current task, and use it to prune the action space. However, neither the domain-specific dataset nor the admissibility signal is always available. This leads to the question of whether it is possible to adjust the prior knowledge without incurring additional costs or making assumptions.

To address this challenge, we propose **M**utual-**I**nformation Regularized **P**olicy **O**ptimization, namely **MIPO**. Unlike previous work where the hard action mask is used to prune the action space, we use the pretrained language model to generate a prior policy over the action space of the agent. The agent is subsequently trained to optimize the environmental reward as well as being close to this prior. Then, based on the marginalization of the agent's policy, we dynamically adjust the general

†Corresponding author.

prior provided by the pretrained language model to adapt to the specific environment. Such a learning process can be unified as a mutual-information (MI) regularized reinforcement learning problem, where the standard RL objective is augmented with the mutual information between the agent's policy and non-text state information. In theory, we prove that MIPO achieves the monotonic improvement on the MI-regularized objective. To validate our method in practice, we perform a series of experiments on text-based games. The experimental results show that MIPO outperforms baselines in terms of training speed and final performance.

Our contribution can be summarized as follows. First, we propose MIPO, an MI-regularized policy optimization method that achieves implicit dynamic action space reduction without requiring additional datasets or assumptions. Then, we prove that our method achieves monotonic improvement in the MI-regularized objective. Finally, we empirically validate our method and demonstrate its effectiveness.

2 Related Work

2.1 RL in Text-Based Games

In text-based games (Côté et al., 2019), the environment takes an action described in natural language and returns the natural language description of the current state. Many studies have focused on solving text-based games using only text input. While early work among them mostly focuses on the network structure (Narasimhan et al., 2015; He et al., 2016), recent studies incorporate techniques such as hierarchical structure (Adolphs and Hofmann, 2020; Zhu et al., 2023), multi-passage reading comprehension (Guo et al., 2020), and well-designed exploitation and exploration mechanism (Tuyls et al., 2022) to achieve better performance.

Other than using pure text input, another line of research combines the knowledge graph into the learning algorithm to achieve better state representation. For example, Ammanabrolu and Riedl (2019); Ammanabrolu and Hausknecht (2019); Adhikari et al. (2020) consider different GNN structures for the representation of knowledge graph, Xu et al. (2020, 2021) introduce a hierarchical structure for the state representation and the decision-making process, and Atzeni et al. (2021) combine case-based reasoning with knowledge graph for the better exploration strategy.

Technically, our algorithm can be combined with any of the above methods. However, we choose to combine the knowledge graph into our algorithm, as it is a more general setting where the state does not contain only text information.

2.2 Action Space Reduction

The sample complexity of the RL algorithm generally grows with the size of the action space. This becomes a problem for text-based games due to its combinatorial nature on the action space. However, as natural language actions can be understood by humans and language models, various methods have been proposed to reduce the size of the action space using its semantic nature. In Ammanabrolu and Riedl (2019), a rule-based method is used to extract information from the knowledge graph and perform subsequent action selection. While this method requires domain knowledge about the environment and the corresponding knowledge graph, some other methods (Zahavy et al., 2018; Ammanabrolu and Hausknecht, 2019; Jain et al., 2020) assume the feedback of action admissibility from the environment and use it to prune the action space. To remove the assumption about the environment or the requirement of domain knowledge, Martin et al. (2022) utilize the power of pretrained language models to perform the action space reduction. However, as prior knowledge of the pretrained language model is generally derived from general corpora, it may exhibit a certain level of mismatch of the actual environment. To alleviate such a mismatch, domain-specific dataset (Yao et al., 2020; Xu et al., 2022) is collected to further tune the pretrained language models for better adaption.

Unlike these works, our method makes no assumption on the domain knowledge, the admissibility feedback, or the domain-specific dataset. The adaption of the pretrained language model is guided by the policy of the agent, which is derived from the mutual-information regularized objective. To the best of our knowledge, only Shi et al. (2023) uses the same setting of our paper, where the pretrained language model is tuned via self-supervised learning. We take it as a baseline in our experiments.

2.3 Regularization in RL

The goal of RL algorithms is to learn a policy that optimizes a single objective, the cumulative discounted reward. However, in practice, we may want to also incorporate other features into the learned policy. This is usually achieved via incorporating a regularization term into the policy optimization process. For example, SQL (Haarnoja et al., 2017) and SAC (Haarnoja et al., 2018a) use the entropy of policy as a regularization term to encourage exploration. The entropy term can be considered as the KL divergence between the policy and a uniform prior policy, therefore we can change the uniform prior to other fixed prior to achieve different purposes. For example, in offline RL (Levine et al., 2020), BRAC (Wu et al., 2019) and CQL (Kumar et al., 2020) use the KL to control the distance between the learned policy and the behavior policy to mitigate the instability arising from previously unseen actions. Such a regularization is also used in PPO (Schulman et al., 2017) and is further used for the optimization of large language models (Ouyang et al., 2022; Rafailov et al., 2023). While previous methods consider a fixed prior, Grau-Moya et al. (2018) and Leibfried and Grau-Moya (2020) consider a dynamic prior learned along the policy optimization process as an improvement of SAC (Haarnoja et al., 2018a). Unlike these methods, we consider the case where the state information can be decomposed into textual and non-textual components, we then use a pretrained language model as the prior to handle the textual components and adapt this language model dynamically during the policy optimization process.

2.4 Foundation Models for RL

Foundation models (Yang et al., 2023) pretrained on diverse and large-scale datasets contain valuable prior knowledge, providing useful insight on the downstream tasks. Given the success of foundation models, using foundation models to achieve a better decision-making process has become a popular research topic. Except from the action space reduction, foundation models have also been used in many different perspectives to enhance the performance of RL algorithms. For example, it can serve as a reward model (Kwon et al., 2022), dynamic model (Zhong et al., 2022), or affordance function (Ahn et al., 2022). It can also be used to generate intrinsic reward (Du et al., 2023) to encourage exploration. For long-horizon tasks, foundation models have been used for different hierarchical methods (Nottingham et al., 2023; Wang et al., 2023; Yuan et al., 2023b) to generate sub-goals. It can also be used to parameterize the agent's policy directly (Carta et al., 2023). Unlike these methods, we focus on using pretrained language model to reduce the action space.

3 Background

Reinforcement Learning Problem. We often formulate the RL problem as a Markov Decision Process (MDP) (Bellman, 1957). An MDP can be defined by a tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \rho_0, \gamma \rangle$. At each state $s \in \mathcal{S}$, the agent can choose one of the actions $a \in \mathcal{A}$ to execute and receive a reward signal $r(s,a)$ from reward function $R(s,a) : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$. The state will then transition from the current state s to the next state s', governed by transition function $P(s'|s,a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$. In this problem, the goal of the agent is to properly select action following policy $\pi(a|s)$ at each state to maximize the expected cumulative discounted reward $\mathbb{E}_{\rho_0,\pi,P}[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t)]$, where the expectation is over initial state distribution ρ_0, agent policy π and transition function P, $\gamma \in (0,1)$ is the discount factor, and t denotes timestep.

The RL objective can also be represented by the reward function and the state marginal distribution as follows:

$$(1-\gamma)\,\mathbb{E}_{\rho_0,\pi,P}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t)\Big] = \mathbb{E}_{\rho_\pi(s)}\,\mathbb{E}_{\pi(a|s)}\Big[r(s,a)\Big],$$

where $\rho_\pi(s) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t P_t^\pi(s)$, and $P_t^\pi(s)$ denotes the probability that policy π visits state s at timestep t.

RL with Natural Language Action Space. For RL with natural language action space, each action a is paired with the corresponding semantic and therefore can be encoded into the action representation h_a using pretrained language models. Without loss of generality, we further assume each state can be decomposed into two components, $s = \{\mathrm{s}_{\text{text}}, \mathrm{s}_{\text{non-text}}\}$, where s_{text} represents the state information that can be aligned with action semantics and $\mathrm{s}_{\text{non-text}}$ represents the state information that cannot be aligned with action semantics. We choose pretrained language models (Sun et al., 2022) for the state-action semantic alignment, as in our experiment scenario, the state information can be decomposed into knowledge graph information and text information.

GATA. Our implementation is based on GATA (Adhikari et al., 2020), which also serves as an important baseline in our experiments. In GATA, the state information is assumed to be decomposed into knowledge graph information $\mathrm{s}_{\text{graph}}$ and text information s_{text}. Each component of the state information is encoded separately and subsequently concatenated to form the joint representation of the state $h_s = [h_{s_{\text{graph}}}, h_{s_{\text{text}}}]$. The state representation h_s and the action representation h_a will be then utilized as inputs of state-action value function $Q(s,a)$, and this function will be updated via the following DQN (Silver et al., 2017) update rule to get the optimal state-action value function $Q^*(s,a)$:

$$Q(s,a) = r(s,a) + \gamma\, \mathbb{E}_{P(\cdot|s,a)}\left[\arg\max_{a'} Q(s',a')\right].$$

4 Method

In this section, we present Mutual-Information Regularized Policy Optimization (MIPO). MIPO leverages a pretrained language model conditioned on the state variable s_{text} to generate a prior policy $\pi^{\text{prior}}(a|\, s_{\text{text}})$. Then in Section 4.1, we incorporate the KL divergence between the current policy $\pi(a|s)$ and the prior policy into the policy iteration process. In this way, we encourage the policy to optimize the environmental reward, as well as being close to the prior policy. Inspired by the mutual-information (MI) regularization, we exploit the marginalization of the agent's policy to adapt the prior policy to the environment. As shown in Section 4.2, this alternative optimization process yields a monotonic improvement on the MI-regularized RL objective. Finally, in Section 4.3, we show how to implement this optimization process in practice.

4.1 KL-Regularized RL for Fixed Prior

Taking s_{text} and the semantics of each action as inputs, a pretrained language model is able to produce a prior policy on the action space $\pi^{\text{prior}}(a|\, s_{\text{text}})$. Instead of directly using it as a hard mask for explicit action space reduction, we aim to learn a policy that can maximize the environmental reward as well as minimize the distance between the agent's policy and the prior policy, which serves as implicit action space reduction.

We choose the KL divergence to measure the distance such that the policy evaluation operator can be defined as follows:

$$\begin{aligned} Q(s,a) &= r(s,a) \\ &+ \gamma\, \mathbb{E}_{P,\pi}\left[Q(s',a') - \alpha \log \frac{\pi(a'|s')}{\pi^{\text{prior}}(a'|\, {s_{\text{text}}}')}\right], \end{aligned} \tag{1}$$

where the expectation is over $P(\cdot|s,a)$ and $\pi(\cdot|s')$, and α is the coefficient for the trade-off between the actual return and the KL term. Note that if we take $\pi(\cdot|s')$ inside the expectation, the log term becomes $D_{\text{KL}}(\pi \| \pi^{\text{prior}})$. Then, the following optimization problem is used to achieve the policy improvement:

$$\pi(\cdot|s) = \arg\max_{\pi'} \mathbb{E}_{\pi'}\left[Q(s,\cdot) - \alpha \log \frac{\pi'(\cdot|s)}{\pi^{\text{prior}}(\cdot|\, s_{\text{text}})}\right]. \tag{2}$$

As shown in Su and Lu (2022), by repeatedly applying Equation (1) and (2), the policy converges to the optimal policy in terms of the following the KL-regularized RL objective:

$$\mathbb{E}_{\rho_0,\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t \left(r(s_t,a_t) - \alpha \log \frac{\pi(a_t|s_t)}{\pi^{\text{prior}}(a_t|\, s_{\text{text},t})}\right)\right],$$

which can alternatively be represented as:

$$\mathbb{E}_{\rho_\pi(s)}\, \mathbb{E}_{\pi}\left[r(s,a) - \alpha \log \frac{\pi(a|s)}{\pi^{\text{prior}}(a|\, s_{\text{text}})}\right].$$

The first term is for the reward maximization and the second term is for the KL distance minimization. It is worth noting that the regularization terms in Equation (1) and (2) are not duplicated, as we need to consider the effect of regularization in both the policy evaluation process and the policy improvement process to ensure the improvement over the regularized RL objective.

4.2 MI-Regularized RL for Dynamic Prior

In the previous section, we discuss incorporating the KL divergence between the policy and the prior into the RL objective. However, as the encoded knowledge in pretrained language models often stems from general corpora, there is a possibility that the prior provided by the pretrained language model does not align well with specific environments. Consequently, rigidly constraining the policy to be close to the prior may be suboptimal. To solve this problem, a mechanism is needed to dynamically adjust the prior to comprehend the environment.

As we make no assumption on the environmental feedback, the agent's policy emerges as the sole teacher within this environment. Fortunately, it serves as a good one as it is designed to optimize the environmental reward, which contains enough information to differentiate between good and bad actions. Considering that the agent's policy incorporates more inputs compared to the prior policy, we use the conditional marginalization of the agent's policy to adapt the prior policy as follows:

$$\begin{aligned} &\pi^{\text{prior}}(a|\, s_{\text{text}}) \\ &\quad = \mathbb{E}_{\rho_\pi(s_{\text{non-text}}\,|\, s_{\text{text}})}\left[\pi(a|\, s_{\text{text}}, s_{\text{non-text}})\right], \end{aligned} \tag{3}$$

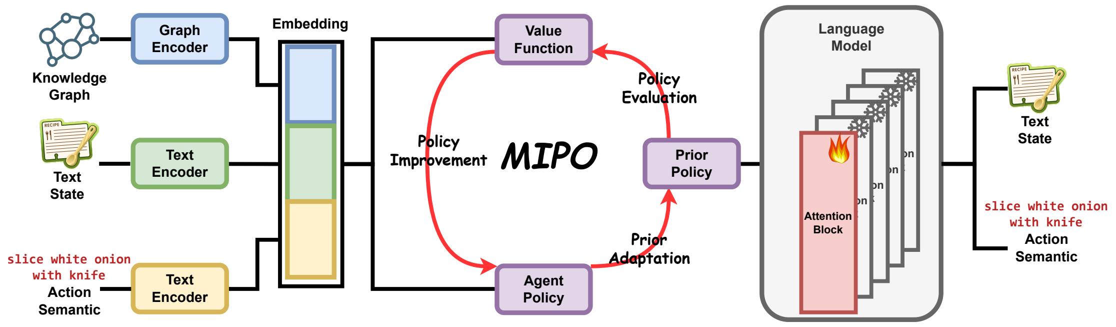

Figure 1: The MIPO Framework. Embeddings produced by different encoders are concatenated into a single one and then fed into the policy and the value network. MIPO iteratively applies policy evaluation, policy improvement, and prior adaptation to achieve the monotonic improvement on the MI-regularized RL objective.

where $\rho_\pi(s) = \rho_\pi(\mathrm{s_{text}})\rho_\pi(\mathrm{s_{non\text{-}text}} \,|\, \mathrm{s_{text}})$ is the conditional factorization of the state marginal distribution of policy π. Such an association with the agent's policy indicates that the adaptation of the prior policy should be performed on the states that are currently visited by the agent's policy, which is crucial from a theoretical perspective as shown later.

Incorporating this prior adaptation process into the policy iteration framework leads to an alternating optimization process of $(\pi, \pi^{\mathrm{prior}})$. In the following theorem, we prove that this alternating optimization can be unified as the optimization of MI-regularized RL objective and ensures the monotonic improvement on this objective.

Theorem 1. *If a sequence* $(\pi_k, \pi_{\mathrm{k}}^{\mathrm{prior}})_{k=0}^{\infty}$ *is obtained by iteratively applying Equation* (1),(2) *and* (3)*, then it exhibits the monotonic improvement property on the MI-regularized RL objective,* $J(\pi_{k+1}, \pi_{\mathrm{k+1}}^{\mathrm{prior}}) \geq J(\pi_k, \pi_{\mathrm{k}}^{\mathrm{prior}})$*. Here, the MI-regularized RL objective* $J(\pi, \pi^{\mathrm{prior}})$ *is defined as:*

$$J(\pi, \pi^{\mathrm{prior}}) = \mathbb{E}_{\rho_\pi(s)}\, \mathbb{E}_\pi \left[r(s,a) - \alpha \log \frac{\pi(a|s)}{\pi^{\mathrm{prior}}(a|\, \mathrm{s_{text}})} \right].$$

Proof. See Appendix A.1. □

In this theorem, we first treat the prior policy as a fixed one and prove that, by applying Equations (1) and (2), the policy iteration with a fixed prior leads to the improvement on the MI-regularized RL objective, $J(\pi_{k+1}, \pi_{\mathrm{k}}^{\mathrm{prior}}) \geq J(\pi_k, \pi_{\mathrm{k}}^{\mathrm{prior}})$. Then, we fix the agent's policy and perform the prior policy adaption by applying Equation (3), which also leads to the improvement on the objective, $J(\pi_{k+1}, \pi_{\mathrm{k+1}}^{\mathrm{prior}}) \geq J(\pi_{k+1}, \pi_{\mathrm{k}}^{\mathrm{prior}})$. Put them together, we can have the monotonic improvement $J(\pi_{k+1}, \pi_{\mathrm{k+1}}^{\mathrm{prior}}) \geq J(\pi_k, \pi_{\mathrm{k}}^{\mathrm{prior}})$.

In contrast to the policy iteration that optimizes the KL-regularized RL objective, we now have two variables π and π^{prior}. Therefore, the augmented term does not simply stand for the KL divergence between π and an arbitrary π^{prior}, but it now stands for the mutual information between a and $\mathrm{s_{non\text{-}text}}$, $\mathrm{MI}(a; \mathrm{s_{non\text{-}text}} \,|\, \mathrm{s_{text}})$.

From an information theoretical perspective, this objective implies that among policies sharing the same RL objective, we favor those with lower dependency on $\mathrm{s_{non\text{-}text}}$ when $\mathrm{s_{text}}$ is given. Such policies are easier to learn by the prior, as the prior takes only $\mathrm{s_{text}}$ as the input.

4.3 MIPO framework

In Section 4.1 and 4.2, we glance at our algorithm from a theoretical perspective. In this section, we discuss the practical implementation of our algorithm using neural networks.

In MIPO, the agent is equipped with three different network modules. One critic network $Q(s, a; \theta)$ takes the knowledge graph, text information, and the semantics of actions as input and outputs the state-action value for each action. One policy network $\pi(a|s;\psi)$ also takes the knowledge graph, text information, and the semantics of actions as input and outputs the probability for each action. One prior network $\pi^{\mathrm{prior}}(a|\, \mathrm{s_{text}}; \phi)$ takes only the text information and the semantics of actions and outputs the prior probability for each action. The framework of MIPO is illustrated in Figure 1.

For the update of $Q(s, a; \theta)$, we follow Equation (1), and update it by minimizing the following TD error:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathcal{D}} \left[\left(Q(s,a;\theta) - \left[r(s,a) + \gamma \left(\hat{Q}(s',a') - \alpha \log \frac{\pi(a'|s')}{\hat{\pi}^{\mathrm{prior}}(a'|\, \mathrm{s_{text}}')} \right) \right] \right)^2 \right], \tag{4}$$

where $\mathcal{D}$ denotes the replay buffer, and $\hat{Q}$ and $\hat{\pi}^{\mathrm{prior}}$ are respectively the target critic and the target prior policy. These target networks are used to stabilize the training of $Q(s, a; \theta)$ and periodically updated via copying the parameters of $Q(s, a; \theta)$ and $\pi^{\text{prior}}(a|\, s_{\text{text}}; \phi)$.

To update $\pi(a|s; \psi)$, we follow Equation (2), and update it via minimizing the following objective:

$$\mathcal{L}(\psi) = \mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_{\pi(a|s;\psi)}\left[\alpha \log \frac{\pi(a|s;\psi)}{\hat{\pi}^{\text{prior}}(a|\, s_{\text{text}})} - Q(s,a)\right]\right], \tag{5}$$

where, again, the target prior $\hat{\pi}^{\text{prior}}$ is used to stabilize the training of $\pi(a|s; \psi)$.

Following Equation (3), we update the prior policy $\pi^{\text{prior}}(a|\, s_{\text{text}}; \phi)$ via the the maximum likelihood estimation as follows:

$$\mathcal{L}(\phi) = \mathbb{E}_{\tilde{\mathcal{D}}}\left[\mathbb{E}_{\pi(a|s)}\left[\log \pi^{\text{prior}}(a|\, s_{\text{text}}; \phi)\right]\right], \tag{6}$$

where $\tilde{\mathcal{D}}$ is another replay buffer much smaller than $\mathcal{D}$, such that it stores only the recent transitions and properly approximates the state marginal distribution $\rho_\pi(s)$. The overall learning algorithm of MIPO is summarized in Appendix A.2.

5 Experiments

5.1 Experiment Settings

We conduct two sets of experiments on cooking games provided in TextWorld (Côté et al., 2019) and house-holding tasks based on VirtualHome (Puig et al., 2018).

For the TextWorld environment, all tasks are generated using the code provided in GATA (Adhikari et al., 2020) and follow the same task setup as described in GATA. In this experiment, agents will receive recipes described in natural language including ingredients and directions. It will also receive a knowledge graph including triplets like *[white onion, fridge, in]* describing the current state of the environment. Based on this information, agents will take natural language actions like *"take white onion from fridge"* to finish the given recipes. There are four difficulty levels, each containing 100 different tasks for training and 20 different tasks for test. In each task, the goal of the agent is to collect different ingredients and cook them based on the recipe.

We make the following modifications to make tasks more challenging. First, we remove the recipe information from the knowledge graph. Therefore, the agent has to use both the information from the

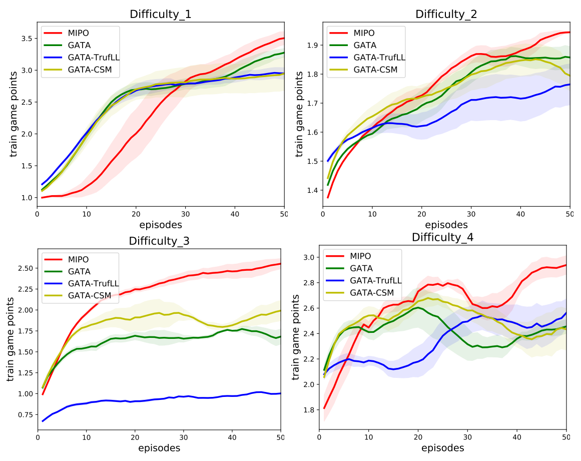

Figure 2: Learning curves of all the methods in TextWorld cooking games.

knowledge graph and the information from texts to make proper decisions. Second, some actions are filtered in a hard-code manner in GATA, we recover these actions and result in a larger action space. Third, the text information contains only recipe information. With the above modifications, our setting is much harder than the original setting in GATA. More details about experiments and hyperparameters are included in Appendix A.3.

For the VirtualHome environment, we customize a set of house-holding tasks where agents are supposed to follow the given natural language instructions to tidy the room. For example, agents will receive natural language instructions like *"put waterglass-283 into / on kitchencabinet-238"*, and a knowledge graph including triplets like *[waterglass-283, kitchentable-232, on]*. Based on these information, agents will take natural language actions like *"walktowards waterglass-283"* to finish the given instructions. In difficulty 1, agents will be given one instruction, and in difficulty 2, agents will be given two instructions. For the knowledge graph used in the VirtualHome environment, we use a rule-based filter to remove task-irrelevant objects in the scenarios (for example, floor, wall, ceiling) and only keep task-relevant objects. We then consider task-relevant objects in the scenarios and their relations as edges. The relations we considered in the knowledge graph of VirtualHome include "inside", "on", "facing", "close", "is", and "holds".

5.2 Baselines

We consider the following three baselines compared against our method in the experiments.

GATA (Adhikari et al., 2020): a knowledge-

	D1	D2	D3	D4
MIPO	**3.53**$\pm$0.20	**1.94**$\pm$0.01	**2.57**$\pm$0.12	**2.95**$\pm$0.16
GATA	3.29$\pm$0.13	1.85$\pm$0.07	1.68$\pm$0.18	2.46$\pm$0.11
GATA-TrufLL	2.94$\pm$0.18	1.77$\pm$0.15	0.99$\pm$0.00	2.59$\pm$0.19
GATA-CSM	2.95$\pm$0.53	1.79$\pm$0.11	1.99$\pm$0.26	2.43$\pm$0.33
GATA-SayCan	0.08$\pm$0.02	0.11$\pm$0.00	0.05$\pm$0.00	0.10$\pm$0.02

Table 1: Train game points in TextWorld cooking games. D1 is short for difficulty 1, the same for others.

graph based method, where the relational graph convolution network (Schlichtkrull et al., 2018) is used to process the information of knowledge graph and a transformer (Vaswani et al., 2017) encoder is used to process the text information of both the state and the action. Putting them together, we can have the representation of each state-action pair and then use it as the input of $Q(s, a)$. The agent interacts with the environment using an ϵ-greedy policy and is trained via DQN (Silver et al., 2017). More specifically, we use the GATA-GTF variant for our experiment, as the construction of the knowledge graph is beyond the scope of this paper.

GATA-TrufLL (Martin et al., 2022): a reinforcement learning method paired with action space reduction using pretrained language models. As TrufLL was originally proposed for the language generation tasks, we re-implement TrufLL based on the framework of GATA. For the action space reduction, we use the probability threshold method as the truncation function. The truncation function is shown as follows:

$$g_{p_{\text{th}}(\alpha)}(a|\, s_{\text{text}}) = \mathbb{1}_{f_{\text{LM}}(a|\, s_{\text{text}})>\lambda},$$

which keeps actions with a probability $f_{\text{LM}}(a|\, s_{\text{text}})$ greater than λ. We use a fixed language model to perform action space reduction, and the agent is trained as GATA with the reduced action space.

GATA-CSM (Shi et al., 2023): a reinforcement learning method paired with action space reduction using pretrained language models. To adapt the pretrained model to the current task, CSM collects trajectories with good performance using a heap-based replay buffer and adapts the pretrained model based on this replay buffer. As CSM was originally proposed for the action generation setting, we re-implement CSM based on the framework of GATA. We use the same action space reduction method and training method as GATA-TrufLL for CSM, except the language model is not fixed, but dynamically adjusted based on the collected trajectories.

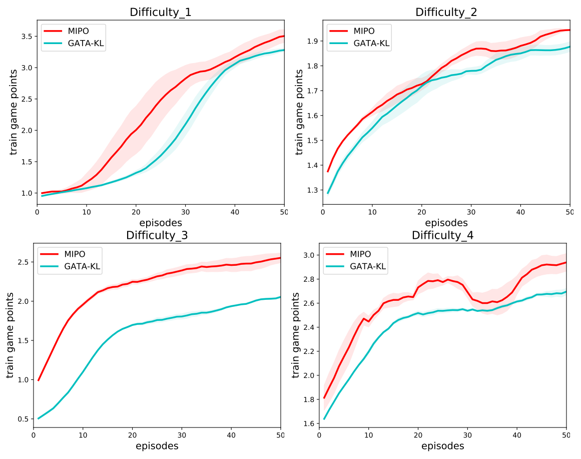

Figure 3: Learning curves of MIPO and GATA-KL in TextWorld cooking games.

GATA-SayCan (Shi et al., 2023): a post-processing method to utilize the knowledge in the pretrained language model. In SayCan, we assume a learned state-action value function. When executed in the environment, the action is chosen based on not just the state-action value function, but also taking the probability provided by the language model into consideration as follows:

$$Q^{\text{combined}}(s, a) = Q(s, a) \times f_{\text{LM}}(a|\, s_{\text{text}}).$$

The state-action value function used for SayCan is trained via GATA.

MIPO: the proposed method in this paper. We implement our algorithm based on GATA with minimum changes to adapt to the policy iteration. For the prior policy π^{prior}, we use the output of the language model as the logit, and get the prior policy as follows:

$$\pi^{\text{prior}}(a|\, s_{\text{text}}) = \frac{\exp(f_{\text{LM}}(a|\, s_{\text{text}})/\beta)}{\sum_{a\in A}\exp(f_{\text{LM}}(a|\, s_{\text{text}})/\beta)}.$$

5.3 Implementation Details

We use DistilBERT (Sanh et al., 2019) for methods paired with a language model. To compute $f_{\text{LM}}(a|\, s_{\text{text}})$, we first get the last hidden state from the language model for both the text state information and the action semantic as the representation of the state and action. Then these representations are used to compute the cosine similarity, which serves as the probability $f_{\text{LM}}(a|\, s_{\text{text}})$. For methods that adjust the language model, we fix all previous transformer blocks and only train the last one.

5.4 TextWorld

Main Result. We plot the learning curves of all methods in Figure 2 and present all final results in

	D1	D2	D3	D4
MIPO	**3.12**±0.20	**1.94**±0.04	**2.16**±0.04	**2.64**±0.21
GATA	2.73±0.26	1.77±0.16	1.47±0.05	2.25±0.33
GATA-TrufLL	2.52±0.24	1.72±0.09	0.9±0.0	2.15±0.57
GATA-CSM	2.58±0.48	1.78±0.10	1.75±0.18	2.47±0.33
GATA-SayCan	0.08±0.02	0.08±0.02	0.1±0.07	0.05±0.0

Table 2: Test game points of final models in TextWorld cooking games.

Table 1. MIPO obtains the best performance in all four difficulty levels. It achieves the improvement of **7.3%**, **4.9%**, **29.1%** and **13.9%** respectively in four different levels compared to the best baseline.

Except for the fourth difficulty level, GATA-TrufLL achieves a worse result compared to GATA, which indicates that the prior knowledge encoded in the language model deviates from the fact in the environment at least in some cases. As the language model is fixed in GATA-TrufLL, it has no chance to correct the mistake. For GATA-CSM, as it dynamically adjusts the language model based on the interaction with the environment, it always outperforms or achieves the same level of performance compared to GATA-TrufLL. However, it cannot consistently achieve better performance than GATA. One potential reason is that GATA-CSM uses a heap-based replay buffer to collect trajectories for language model adaption. Though this mechanism certainly collects high-quality data, those trajectories may be largely off-policy. Therefore, the language model may have known the fact of some high-quality states, but the agent may be not able to reach those states, which reduces the effectiveness of the action space reduction. On the other hand, our method collects trajectories from $\rho_\pi(s)$, which leads to an on-policy language model adaption. As the adaption is performed on-policy, it alleviates the mismatch on those states that are currently visited by the agent, therefore achieving effective action space reduction and outperforming all other baselines in all four difficulty levels. As we can see from Table 1, GATA-SayCan fails hard in all difficulty levels and is outperformed by its base policy GATA, which indicates that a mismatched language model can make things even worse when not properly used and further demonstrates the importance of the language model adaption.

Generalization. We present the generalization ability of each algorithm in Table 2 (the performance on the test set for all difficulty levels). As we can see, MIPO outperforms all baselines in the four difficulty levels, though we do not make any claim about MIPO having the advantage in terms of generalization ability. This improvement in the test set may be simply the result of MIPO's better learning in the training set. Note that GATA-CSM and GATA-TrufLL still rely on the language model for execution, making it challenging to deploy them on devices with limited computational resources. In contrast, MIPO distills the knowledge from the language model into the policy, resulting in significantly reduced computational requirements compared to GATA-TrufLL and GATA-CSM. This advantage of our algorithm highlights its suitability for environments with constrained computation resources.

Ablation Study. To further verify our method, we perform an ablation study in this section. We propose an ablation baseline, GATA-KL. In GATA-KL, all other perspectives are the same as MIPO, except the language model is fixed and will not be trained. As we can see from Figure 3, MIPO outperforms GATA-KL in all four difficulty levels, indicating that the performance improvement is indeed achieved by the dynamic adaption of the language model. As we can see, MIPO achieves the largest gap against GATA-KL in Difficulty 3. This is because the prior policy resulting from the language model has a relatively low performance on Difficulty 3 (it achieves 0.47 in Difficulty 3, 1.02, 1.45, and 2.15 in Difficulty 1, 2, and 4 respectively). Therefore the necessity of language model adaptation is demonstrated best in this difficulty level.

5.5 VirtualHome

We further evaluate our method in the VirtualHome environment. All final results are presented in Table 3. As we can see, the comparison among GATA, GATA-TrufLL, and GATA-CSM is similar to the TextWorld experiment, where GATA-TrufLL achieves a worse result compared to GATA and GATA-CSM outperforms or achieves the same level of performance compared to GATA-TrufLL but can not consistently outperform GATA. Such a similar result strengthens the conclusion we made in the TextWorld experiment and again highlights not only the necessity of the language model adaption but also the necessity of adapting the language model in the right way.

Different from the TextWorld experiment, GATA-SayCan achieves a much better result here. This is because that the language instruction in this experiment is more direct and clear, so the performance of the language model becomes bet-

	D1	D2
MIPO	**18.52**±6.35	**19.36**±1.86
GATA	5.05±3.89	9.66±0.22
GATA-TrufLL	3.09±3.35	10.86±0.23
GATA-CSM	3.15±4.18	9.8±0.54
GATA-SayCan	11.23±4.42	10.46±0.32

Table 3: Train game points per episode in VirtualHome house-holding tasks.

ter than the previous experiments and can therefore improve the performance of its base model (GATA). However, we notice that GATA-TrufLL and GATA-CSM do not enjoy the same benefits. This is because it is challenging to set a proper threshold for the truncation function. For example, when we take *"put waterglass-283 into / on kitchencabinet-238"* as the instruction, for actions *"walktowards kitchencabinet-235"*, *"walktowards kitchencabinet-236"*, *"walktowards kitchencabinet-237"* and *"walktowards kitchencabinet-238"*, they all have a high probability with the given instruction (0.90 for *"walktowards kitchencabinet-238"* and 0.89 for others). But for another necessary action *"walktowards waterglass-283"*, it only has a probability of 0.80. Therefore, to make sure we retain all necessary actions, we have to set a relatively low threshold (0.75) in this experiment, such that the truncated action space is still very large. For GATA-TrufLL and GATA-CSM, as they need to learn from scratch, it is still very challenging for them to achieve good performance with such a large action space. For GATA-SayCan, it only needs to make sure the probability of *"walktowards kitchencabinet-238"* is greater than other *"walktowards kitchencabinet"* actions ($0.90 > 0.89$), therefore it can fully enjoy the benefit of improved performance of the language model.

As for our method, it achieves the best performance in both difficulties, which matches the result in TextWorld experiments, and again demonstrates the effectiveness of MIPO.

6 Conclusion

In this paper, we propose MIPO, which achieves dynamic language model adaption and implicit action space reduction for reinforcement learning with natural language action space. We show the problem can be formulated as the optimization of the MI-regularized RL objective, and we theoretically prove that MIPO achieves monotonic improvement on the MI-regularized RL objective. We perform experiments on text-based games, and our MIPO outperforms all baselines, which justifies its effectiveness.

Limitations and Potential Risks

One limitation of this work is the convergence in terms of policy. As suggested by the theory, we can only guarantee the monotonic improvement on the objective, but the policy may oscillate between several equivalent points. Although we do not observe any significant impact in our experiment, it is still less satisfying and will be addressed in our future work. One another limitation of our work is the value of α, which is currently set heuristically. Although we find it easy to tune in our experiment, a fully automatic method, like the one used in SAC (Haarnoja et al., 2018b), to set α will be more welcomed and will be included in our future work. To the best of our knowledge, there is no obvious risk to our work.

Acknowledgements

This work was supported in part by NSF China under grant 62250068.

References

Ashutosh Adhikari, Xingdi Yuan, Marc-Alexandre Côté, Mikuláš Zelinka, Marc-Antoine Rondeau, Romain Laroche, Pascal Poupart, Jian Tang, Adam Trischler, and William L Hamilton. 2020. Learning dynamic belief graphs to generalize on text-based games. In *Conference on Neural Information Processing Systems (NeurIPS).*

Leonard Adolphs and Thomas Hofmann. 2020. Ledeepchef deep reinforcement learning agent for families of text-based games. In *Conference on Artificial Intelligence (AAAI).*

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691.*

Prithviraj Ammanabrolu and Matthew Hausknecht. 2019. Graph constrained reinforcement learning for natural language action spaces. In *International Conference on Learning Representations (ICLR).*

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

Mattia Atzeni, Shehzaad Zuzar Dhuliawala, Keerthiram Murugesan, and Mrinmaya Sachan. 2021. Case-based reasoning for better generalization in textual reinforcement learning. In *International Conference on Learning Representations (ICLR)*.

Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. 2017. Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning (ICML)*.

Richard Bellman. 1957. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684.

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. *arXiv preprint arXiv:2302.02662*.

Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. 2019. Textworld: A learning environment for text-based games. In *Computer Games: 7th Workshop at CGW 2018*.

Thomas M Cover. 1999. *Elements of information theory*. John Wiley & Sons.

Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. 2023. Guiding pretraining in reinforcement learning with large language models. *arXiv preprint arXiv:2302.06692*.

Jordi Grau-Moya, Felix Leibfried, and Peter Vrancx. 2018. Soft q-learning with mutual-information regularization. In *International Conference on Learning Representations (ICLR)*.

Xiaoxiao Guo, Mo Yu, Yupeng Gao, Chuang Gan, Murray Campbell, and Shiyu Chang. 2020. Interactive fiction game playing as multi-paragraph reading comprehension with reinforcement learning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. 2017. Reinforcement learning with deep energy-based policies. In *International conference on machine learning (ICML)*. PMLR.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018a. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning (ICML)*.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. 2018b. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

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 *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Vishal Jain, William Fedus, Hugo Larochelle, Doina Precup, and Marc G Bellemare. 2020. Algorithmic improvements for deep reinforcement learning applied to interactive fiction. In *Conference on Artificial Intelligence (AAAI)*.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. 2022. Reward design with language models. In *International Conference on Learning Representations (ICLR)*.

Felix Leibfried and Jordi Grau-Moya. 2020. Mutual-information regularization in markov decision processes and actor-critic learning. In *Conference on Robot Learning (CORL)*.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

Yuxi Li. 2017. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.

Alice Martin, Guillaume Quispe, Charles Ollion, Sylvain Le Corff, Florian Strub, and Olivier Pietquin. 2022. Learning natural language generation with truncated reinforcement learning. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kolby Nottingham, Prithviraj Ammanabrolu, Alane Suhr, Yejin Choi, Hannaneh Hajishirzi, Sameer Singh, and Roy Fox. 2023. Do embodied agents dream of pixelated sheep?: Embodied decision making using language guided world modelling. In *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, et al. 2022. Training language models to follow instructions with human feedback. In *Conference on Neural Information Processing Systems (NeurIPS)*.

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.

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 *Conference on Neural Information Processing Systems (NeurIPS)*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference (ESWC)*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zijing Shi, Yunqiu Xu, Meng Fang, and Ling Chen. 2023. Self-imitation learning for action generation in text-based games. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Kefan Su and Zongqing Lu. 2022. Divergence-regularized multi-agent actor-critic. In *International Conference on Machine Learning (ICML)*.

Kaili Sun, Xudong Luo, and Michael Y Luo. 2022. A survey of pretrained language models. In *International Conference on Knowledge Science, Engineering and Management (KSEM)*.

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

Jens Tuyls, Shunyu Yao, Sham Kakade, and Karthik Narasimhan. 2022. Multi-stage episodic control for strategic exploration in text games. In *International Conference on Learning Representations (ICLR)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023. Describe, explain, plan and select: interactive planning with llms enables open-world multi-task agents. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Yifan Wu, George Tucker, and Ofir Nachum. 2019. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*.

Yunqiu Xu, Meng Fang, Ling Chen, Yali Du, and Chengqi Zhang. 2021. Generalization in text-based games via hierarchical reinforcement learning. In *Findings of the Association for Computational Linguistics at EMNLP 2021*.

Yunqiu Xu, Meng Fang, Ling Chen, Yali Du, Joey Zhou, and Chengqi Zhang. 2022. Perceiving the world: Question-guided reinforcement learning for text-based games. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yunqiu Xu, Meng Fang, Ling Chen, Yali Du, Joey Tianyi Zhou, and Chengqi Zhang. 2020. Deep reinforcement learning with stacked hierarchical attention for text-based games. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Sherry Yang, Ofir Nachum, Yilun Du, Jason Wei, Pieter Abbeel, and Dale Schuurmans. 2023. Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv:2303.04129*.

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. 2020. Keep calm and explore: Language models for action generation in text-based games. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Haoqi Yuan, Chi Zhang, Hongcheng Wang, Feiyang Xie, Penglin Cai, Hao Dong, and Zongqing Lu. 2023a. Plan4mc: Skill reinforcement learning and planning for open-world minecraft tasks. *arXiv preprint arXiv:2303.16563*.

Haoqi Yuan, Chi Zhang, Hongcheng Wang, Feiyang Xie, Penglin Cai, Hao Dong, and Zongqing Lu. 2023b. Skill reinforcement learning and planning for open-world long-horizon tasks. In *Foundation Models for Decision Making Workshop at NeurIPS 2023*.

Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J Mankowitz, and Shie Mannor. 2018. Learn what not to learn: action elimination with deep reinforcement learning. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2021. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, pages 321–384.

Victor Zhong, Jesse Mu, Luke Zettlemoyer, Edward Grefenstette, and Tim Rocktäschel. 2022. Improving policy learning via language dynamics distillation.

In *Conference on Neural Information Processing Systems (NeurIPS)*.

Anjie Zhu, Peng-Fei Zhang, Yi Zhang, Zi Huang, and Jie Shao. 2023. Abstract then play: A skill-centric reinforcement learning framework for text-based games. In *Findings of the Association for Computational Linguistics at ACL 2023*.

A Appendix

A.1 Proof of Theorem 1

Theorem 1. *If a sequence* $(\pi_k, \pi^{\text{prior}}_{\text{k}})_{k=0}^{\infty}$ *is obtained by iteratively applying Equation* (1),(2) *and* (3)*, then it exhibits the monotonic improvement property on the MI-regularized RL objective,* $J(\pi_{k+1}, \pi^{\text{prior}}_{\text{k+1}}) \geq J(\pi_k, \pi^{\text{prior}}_{\text{k}})$*. Here, the MI-regularized RL objective* $J(\pi, \pi^{\text{prior}})$ *is defined as:*

$$J(\pi, \pi^{\text{prior}}) = \mathbb{E}_{\rho_\pi(s)}\, \mathbb{E}_\pi \left[r(s,a) - \alpha \log \frac{\pi(a|s)}{\pi^{\text{prior}}(a|\, \mathrm{s}_{\text{text}})} \right].$$

Proof. We begin our proof by the following definition of the KL-augmented reward:

$$r^{\pi}_{\pi^{\text{prior}}}(s,a) = r(s,a) - \alpha\, \mathbb{E}_{s' \sim P, a' \sim \pi} \left[\log \frac{\pi(a'|s')}{\pi^{\text{prior}}(a'|s')} \right].$$

To avoid the discussion on the corner case, we assume π and π^{prior} take the form of ϵ-soft policy (Sutton and Barto, 2018).

Based on this definition of the KL-augmented reward, and the following way of the policy evaluation (7) and the policy improvement (8):

$$Q(s,a) = r(s,a) + \gamma\, \mathbb{E}_{P,\pi} \left[Q(s',a') - \alpha \log \frac{\pi(a'|s')}{\pi^{\text{prior}}(a'|\, \mathrm{s}_{\text{text}}')} \right], \tag{7}$$

$$\pi(\cdot|s) = \arg\max\nolimits_{\pi'} \mathbb{E}_{\pi'} \left[Q(s,\cdot) - \alpha \log \frac{\pi'(\cdot|s)}{\pi^{\text{prior}}(\cdot|\, \mathrm{s}_{\text{text}})} \right], \tag{8}$$

for a pair of $(\pi_k, \pi^{\text{prior}}_{\text{k}})$, we can follow Theorem 2 in Su and Lu (2022) to show the following conclusion:

$$J(\pi_{k+1}, \pi^{\text{prior}}_{\text{k}}) \geq J(\pi_k, \pi^{\text{prior}}_{\text{k}}).$$

For the new policy π_{k+1} and the old prior policy $\pi^{\text{prior}}_{\text{k}}$, we have the following objective:

$$J(\pi_{k+1}, \pi^{\text{prior}}_{\text{k}}) = \mathbb{E}_{\rho_{\pi_{k+1}}(s)}\, \mathbb{E}_{\pi_{k+1}(a|s)} \left[r(s,a) - \alpha \log \frac{\pi_{k+1}(a|s)}{\pi^{\text{prior}}_{\text{k}}(a|\, \mathrm{s}_{\text{text}})} \right].$$

To update $\pi^{\text{prior}}_{\text{k}}$ with a fixed π_{k+1}, maximizing the above objective is equivalent to minimizing the following objective:

$$\hat{J}(\pi_{k+1}, \pi^{\text{prior}}_{\text{k}}) = \mathbb{E}_{\rho_{\pi_{k+1}}(s)}\, \mathbb{E}_{\pi_{k+1}(a|s)} \left[\log \frac{\pi_{k+1}(a|s)}{\pi^{\text{prior}}_{\text{k}}(a|\, \mathrm{s}_{\text{text}})} \right].$$

Therefore, for $\pi^{\text{prior}}_{\text{k+1}}$ that takes the following form of the prior adaptation:

$$\pi^{\text{prior}}_{\text{k+1}}(a|\, \mathrm{s}_{\text{text}}) = \mathbb{E}_{\rho_{\pi_{k+1}}(\mathrm{s}_{\text{non-text}}\,|\, \mathrm{s}_{\text{text}})} \left[\pi_{k+1}(a|\, \mathrm{s}_{\text{text}}, \mathrm{s}_{\text{non-text}}) \right],$$

similar to Lemma 10.8.1 in Cover (1999), we can have the following proof:

$$\begin{aligned}
&\hat{J}(\pi_{k+1}, \pi^{\text{prior}}_{\text{k}}) - \hat{J}(\pi_{k+1}, \pi^{\text{prior}}_{\text{k+1}}) \\
&= \mathbb{E}_{\rho_{\pi_{k+1}}, \pi_{k+1}} \left[\log \frac{\pi_{k+1}(a|s)}{\pi^{\text{prior}}_{\text{k}}(a|\, \mathrm{s}_{\text{text}})} - \log \frac{\pi_{k+1}(a|s)}{\pi^{\text{prior}}_{\text{k+1}}(a|\, \mathrm{s}_{\text{text}})} \right] \\
&= \mathbb{E}_{\rho_{\pi_{k+1}}(\mathrm{s}_{\text{text}})} \left[\mathbb{E}_{\rho_{\pi_{k+1}}(\mathrm{s}_{\text{non-text}}\,|\, \mathrm{s}_{\text{text}})}\, \mathbb{E}_{\pi_{k+1}} \left[\log \frac{\pi^{\text{prior}}_{\text{k+1}}(a|\, \mathrm{s}_{\text{text}})}{\pi^{\text{prior}}_{\text{k}}(a|\, \mathrm{s}_{\text{text}})} \right] \right] \\
&= \mathbb{E}_{\rho_{\pi_{k+1}}(\mathrm{s}_{\text{text}})} \left[\mathbb{E}_{\pi^{\text{prior}}_{\text{k+1}}} \left[\log \frac{\pi^{\text{prior}}_{\text{k+1}}(a|\, \mathrm{s}_{\text{text}})}{\pi^{\text{prior}}_{\text{k}}(a|\, \mathrm{s}_{\text{text}})} \right] \right] \\
&= \mathbb{E}_{\rho_{\pi_{k+1}}(\mathrm{s}_{\text{text}})} \left[\mathrm{KL}(\pi^{\text{prior}}_{\text{k+1}} \,||\, \pi^{\text{prior}}_{\text{k}}) \right] \\
&\geq 0
\end{aligned}$$

such that:

$$\hat{J}(\pi_{k+1}, \pi^{\text{prior}}_{\text{k+1}}) \leq \hat{J}(\pi_{k+1}, \pi^{\text{prior}}_{\text{k}}),$$

which indicates:

$$J(\pi_{k+1}, \pi^{\text{prior}}_{\text{k+1}}) \geq J(\pi_{k+1}, \pi^{\text{prior}}_{\text{k}}) \geq J(\pi_{k}, \pi^{\text{prior}}_{\text{k}}),$$

and concludes our proof. □

A.2 Pseudocode

Algorithm 1 MIPO

```
1:  Initialize $\theta$ and $\psi$ for critic network and policy network, load parameters $\phi$ for prior network;
2:  Initialize target critic and target prior policy using $\theta$ and $\phi$;
3:  Initialize replay buffer $\mathcal{D}$ and $|\tilde{\mathcal{D}}|$ $(\ll |\mathcal{D}|)$;
4:  for $i = 1$ to $I$ do
5:      $s \leftarrow s_0$
6:      for $t = 0$ to $T$ do
7:          Sample action $a_t$ from $\pi(a_t|s_t;\psi)$
8:          Execute action $a_t$ and get $r_t$, $s_{t+1}$
9:          Store $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$ and $\tilde{\mathcal{D}}$
10:         Sample batch $B$ from $\mathcal{D}$
11:         Update $Q(s, a;\theta)$ with $B$ by (4)
12:         Update $\pi(a|s;\psi)$ with $B$ by (5)
13:         Sample batch $\tilde{B}$ from $\tilde{\mathcal{D}}$
14:         Update $\pi^{\text{prior}}(a|\,\text{s}_{\text{text}};\phi)$ with $\tilde{B}$ by (6)
15:     end for
16:     if $i$ mod update_interval $= 0$ then
17:         Update target critic and target prior policy with $\theta$ and $\phi$
18:     end if
19: end for
```

A.3 Experiment details

For TextWorld experiments, our method and all baselines are implemented using PyTorch based on the code of GATA (Adhikari et al., 2020)[1]. We use the default hyperparameters of GATA-GTF for all methods, except the maximum number of episodes is set from 100k to 60k and we do not use any pretrained word embedding and set all word embeddings to be trainable. As MIPO uses an actor-critic framework, we add an actor head that contains 3 fully-connected layers with 64/64/1 units activated by ReLU except the last one. This actor head takes the state-action representation provided by the GATA network with gradient detached and produces the logit for each action. The learning rate for the policy network and prior network is 5×10^{-4}. Except for the above-mentioned modifications, every detail is set to be the same as GATA. For VirtualHome[2] experiments, the maximum number of episodes is further reduced to 20k to accelerate the training process. Some hyperparameters are also changed to adapt to this faster training process, for example, the batch size is set to 128 and the multi-step hyperparameter is set to 1.

For MIPO[3] and GATA-KL, two hyperparameters α, β are added. β is fixed as 0.04 for all difficulty levels during training. For α, it always start from $\alpha_{start} = 0.5$ and gradually decay to α_{min} in TextWorld experiments. A small-scale parameter search is performed in Difficulty 1 and 4 for α_{min}. For Difficulty 1, we tried $\alpha_{min} = [0.1, 0.15]$ and found 0.15 to be the best. For Difficulty 4, we tried $\alpha_{min} =$

[1]https://github.com/xingdi-eric-yuan/GATA-public, MIT License.
[2]https://github.com/xavierpuigf/virtualhome, MIT license.
[3]Code available at https://github.com/PKU-RL/MIPO.

$[0.1, 0.15, 0.25]$ and found 0.25 to be the best. For Difficulty 2 and 3, α_{min} is set to be 0.1 heuristically. In VirtualHome experiments, α_{min} is set to be 0.1 for both difficulties. Also, for MIPO, a small replay buffer $\tilde{\mathcal{D}}$ is used for the language model adaption, we set the size of $\tilde{\mathcal{D}}$ to be 12500, which is much smaller than the size of $\mathcal{D}$ (500000). For GATA-TrufLL and GATA-CSM, we set the threshold $\lambda = 0.5$. For methods using pretrained language models, we use DistilBERT (Sanh et al., 2019) downloaded from Hugging Face. Spacy with en_core_web_sm model is used in all methods, which is consistent with GATA.

For all tables, we report the mean and standard derivations of three different random seeds and bold the result with the best mean performance. For all plots, we plot the curves with the moving average of a window size of 10 and a half standard deviation. The unit of x-axis is 1k episodes. For the generalization results, as MIPO uses a stochastic policy, we evaluate the performance of MIPO 10 times for each generalization task and report the mean.

In terms of the computation budget, all TextWorld experiments are trained in machines rented from online services. All machines are equipped with 12 CPU cores and 1 RTX 3090 GPU. Depending on the algorithm and the difficulty level, each experiment takes around 5 to 45 hours to complete. For VirtualHome experiments, all methods are trained in a local machine with 4 Nvidia GPUs (A100) and 224 Intel CPU Cores. Depending on the algorithm and the difficulty level, each experiment takes around 2 to 5 days to complete.

In terms of the use of AI assistants. ChatGPT is used to polish some sentences in this paper.

A.4 Adaptation to Different Language Model

MIPO is a general RL algorithm and can be paired with any pretrained language model, so it should be model-agnostic. We conduct the following experiments to validate this point. In this experiment, we change the pretrained language model to GPT2 for MIPO, GATA-Truff, and GATA-CSM. As we can see from Table 4, MIPO-GPT2 still outperforms other baselines, and it also achieves similar performance with MIPO, which indicates that MIPO is agnostic to the selection of pretrained language model.

	D3
MIPO	2.57 ± 0.12
MIPO-GPT2	2.44 ± 0.19
GATA-TrufLL-GPT2	0.12 ± 0.01
GATA-CSM-GPT2	1.40 ± 0.11

Table 4: Train game points per episode in TextWorld cooking games.