Learning Dialogue Policy Efficiently Through Dyna Proximal Policy Optimization

Chenping Huang(✉) and Bin Cao

College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China
{huangchenping,bincao}@zjut.edu.cn

Abstract. Many methods have been proposed to use reinforcement learning to train dialogue policy for task-oriented dialogue systems in recent years. However, the high cost of interacting with users has seriously hindered the development of this field. In order to reduce this interaction cost, the Deep Dyna-Q (DDQ) algorithm and several variants introduce a so-called *world model* to simulate the user's response and then use the generated simulated dialogue data to train the dialogue policy. Nevertheless, these methods suffer from two main issues. The first is limited training efficiency due to the Deep-Q Network used. The second is that low-quality simulation dialogue data generated by the world model may hurt the performance of the dialogue policy. To solve these drawbacks, we propose the Dyna Proximal Policy Optimization (DPPO) algorithm. DPPO combines the Proximal Policy Optimization (PPO) algorithm with the world model and uses a deactivation strategy to decide when to stop using the world model for subsequent training. We have conducted experiments on the task of movie ticket booking. Experiments show that our algorithm combines the advantages of DDQ and PPO, which significantly reduces the interaction cost required during training and has a higher task success rate.

Keywords: Dialogue policy · Reinforcement learning · World model deactivation · PPO

1 Introduction

Human-machine collaboration methods have been deployed in various scenarios. As a common way of Human-machine collaboration, dialogue systems have been used in E-commerce scenarios, where the dialogue system is used to answer simple questions from customers, while the human customer service is responsible for answering questions that are difficult for the dialogue system to handle. An important branch of dialogue systems is task-oriented dialogue systems, which is designed to help users complete specific tasks, such as booking movie tickets or hotels. This type of system is usually implemented in a pipeline manner [2], where

H. Gao et al. (Eds.): CollaborateCom 2022, LNICST 460, pp. 396–414, 2022.
https://doi.org/10.1007/978-3-031-24383-7_22

following four components are involved (as shown in the Fig. 1): (1) Natural Language Understanding (NLU) [4,24], which parses user utterance into intentions and entities; (2) Dialogue State Tracking (DST) [3,9], which manages the conversation history and outputs the current dialogue state; (3) Policy Learning (PL) [32,33], which learns how to choose the next action based on the current dialogue state; (4) Natural Language Generation (NLG) [16,28], which converts the system action into natural language as a response to the user. The work of this paper is to improve the PL component in the task-oriented dialogue system.

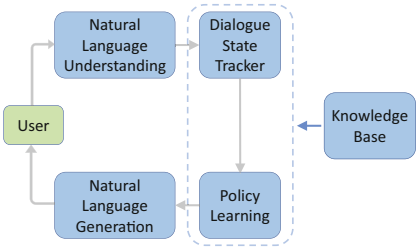


Fig. 1. Pipeline task-oriented dialogue system.

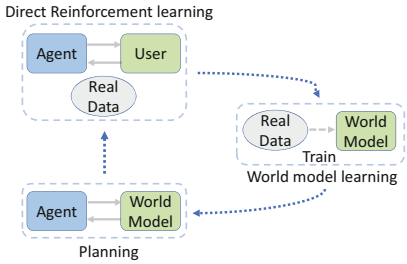


Fig. 2. The process of training the agent by combining a world model.

In recent years, Reinforcement Learning (RL) has been the primary method to optimize the dialogue policy [12,15,23,26,31], where the user and the dialogue system are regarded as the environment and the agent respectively. At the beginning of each conversation, the user performs the first action (makes an utterance) according to their goal, generating the initial state of the environment. After that, for each action (response) of the agent, the environment will give a reward and update the state of the environment. The agent optimizes the dialogue policy according to the reward value to maximize the expected value of the accumulated reward. When using RL to train an agent, it is necessary to let the agent interact with the environment many times. Due to the high labor cost, the cost of directly interacting with the user to train the agent is very high.

Researchers have proposed some methods to reduce the cost of training agents. The most common method is to adopt a user simulator that can simulate user behavior instead of the real user to train the agent, and then let the trained agent interact with the real user for further improvement [5,26,27,31]. However, due to the complexity of real conversations and biases in the design of user simulators, there always exists a discrepancy between real users and simulated users [21].

Another method is to reduce the number of user interactions required to optimize the dialogue policy. The most representative algorithm is Deep Dyna-Q (DDQ) [15], which combines the Dyna-Q framework [22] and the Deep-Q Network (DQN) [13]. After DDQ was proposed, some follow-up works [29,31] extended DDQ and proposed some variants of DDQ. The training process of DDQ is shown in Fig. 2, and it consists of three consecutive stages: (1) *direct*

reinforcement learning, the agent interacts with the real user, and the agent is improved through RL; (2) *world model learning*, based on the real conversation data obtained in the previous stage, supervised learning is used to improve the world model to make it behave more like real user; (3) *indirect reinforcement learning*, or referred as *planning*, the agent interacts with the world model instead of the user, and improves the agent through RL. Because DDQ uses the world model to generate a large number of simulated dialogues to train the agent, it significantly reduces the number of interactions between the agent and the user required for training.

Although DDQ reduces the training cost to some extent, it still suffers from the following drawbacks. First of all, DDQ and most of its variants [29,31] are based on the DQN algorithm, and the characteristic of DQN does not lend itself to the following real-world situations: (1) To effectively train an agent with RL, it is generally necessary to pre-train the agent first. For DQN, the common pre-training method is to use a rule-based agent to generate some experience to pre-fill the experience replay buffer [15]. However, implementing a rule-based agent requires expert knowledge, which causes additional costs. (2) Considering that DQN only supports selecting one action at a turn and to contain more slots in the corresponding utterance, the size of the predefined action set for DQN could be very large. Specifically, for n slots, totally $2^n - 1$ actions could be predefined at most, and such a large action space leads to the exponential growth of computational requirements [30]. As shown in Fig. 3, three slots are involved and to support the system to recommend with different slots, the predefined action set for DQN contains all possible slot combinations, i.e., seven actions.

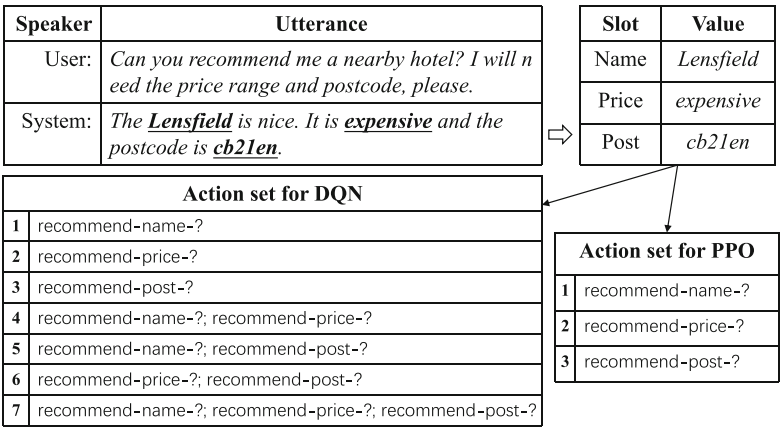


Fig. 3. An example for illustrating action set.

Furthermore, another limitation of DDQ occurs in planning, where the effectiveness of agent training largely depends on the quality of the simulated dialogue from the world model. Specifically, due to the complexity of user behavior,

the simulated user experience inevitably has some deviations from the real user experience. Peng et al. [15] pointed out that in the early stage, training the agent with a large amount of low-quality simulated dialogue can improve the performance of the agent. However, in the later stage of training, low-quality simulated experience may hurt the agent's performance.

To address the above drawbacks of DDQ, we propose a new algorithm called Dyna Proximal Policy Optimization (DPPO). First, DPPO draws on the idea of integrating the world model, but the RL algorithm it uses is Proximal Policy Optimization (PPO) [19]. Compared with DQN, (1) PPO can directly use human-human dialogues to pre-train agents through imitation learning [6], without the need to use rule-based agents; (2) PPO supports multiple actions to be selected at a turn through thresholding the probability. So in the example shown in Fig. 3, the predefined action set for PPO only needs to contain three actions, thus avoiding the difficulty of training due to the large action set.

Second, DPPO uses a deactivation strategy to deal with the situation where simulated experience from the world model may harm the agent's performance. A similarity index is used to continuously monitor the similarity between the world model and the user during the training process. If the similarity is lower than a fixed threshold, we stop using the world model, and then the agent only learns by interacting with the user. Through this strategy, the agent can eliminate the harmful effects of simulated dialogue.

To summarize, our contributions are three-fold:

- To learn dialogue policy efficiently, we propose the DPPO algorithm, which can avoid the inherent shortcomings of DQN by integrating the PPO algorithm with a world model.
- We propose a world model deactivation strategy, which can stop the poorly performing world model in time to avoid the harmful influence of the world model on the agent's performance.
- We conduct extensive experiments on a movie-ticket booking dialogue dataset, and the results show that DPPO outperforms DDQ and its variant in terms of the task success rate.

The rest of this paper is organized as follows. Section 2 presents the preliminaries for dialogue system and reinforcement learning. Section 3 introduces our method in detail. Section 4 validates our method through a series of experiments. Section 5 summarizes related work in the field of dialogue systems. Section 6 concludes the content of the paper.

2 Preliminaries

2.1 Some Concepts of Pipeline System

In order to better understand the task-oriented dialogue system, it is necessary to understand some related concepts first.

- **slot**: In a task-oriented dialogue system, there is usually some task-related information. For example, the information involved in the task of booking movie tickets includes the movie ticket name, time, date, etc., while the information involved in the task of booking a hotel includes the hotel name, room number, and price range. Task-oriented dialogue systems usually use slots to store this information. These slots are usually designed in advance by domain experts.
- **knowledge base**: To accomplish user goals, task-oriented systems often require access to a task-related knowledge base. For example, the knowledge base may store information about all candidate movie theaters for a movie ticket booking task, including their names, cities, star ratings, movie titles and start times.
- **user goal**: When implementing a task-oriented dialogue system, it is usually assumed that the user has a goal to achieve, expressed as $G = (C, R)$ [17], where C is a set of constraints and R is a set of requests. Taking movie ticket booking as an example, the constraints specified by the user might be the name and date of the movie, while the requests could be the location of the movie theater and the start time of the movie. Figure 4 is an example of a user goal in a movie reservation scene. It specifies the user's requirements for booking tickets (movie name, number of tickets, movie start time), and the information the user wants to obtain (cinema name).
- **action**: For each user utterance, the natural language understanding model will parse out the corresponding user action, which is usually represented by the triple [intent, slot type, slot value]. For example, for the user's utterance "I want to watch the movie zootopia, can you book a ticket for me?", the user action extracted by the natural language understanding model may be [inform, movie name, zootopia]. System actions are also generally represented using such triples. For example, if the system wants to further ask the user for the movie's start time that the user wants to book, the system action may be [request, starttime, ?]. Then, the natural language generation model generates the utterance "What time would you like to see it?" based on [request, starttime, ?].
- **dialog state**: The dialog state tracking model maintains the current dialog state. When the research focus is on the dialogue policy learning model, in order to simplify the experiment, researchers often use a rule-based dialogue state tracker for dialogue state generation. The dialog state usually includes four parts: (1) the current user action; (2) the system action of the last turn; (3) the slots the user has informed or requested so far; (4) the knowledge base query result. Different system implementations may include some different additional information in the dialog state.

2.2 Reinforcement Learning

In reinforcement learning, two roles are shown in Fig. 5, namely the environment and the agent. The environment is the world in which the agent lives. At each

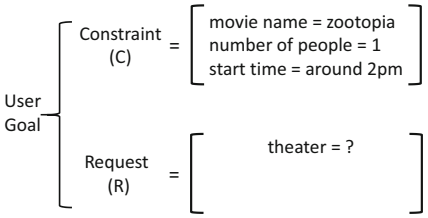


Fig. 4. An example of user goal.

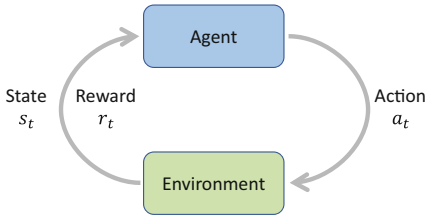


Fig. 5. The agent-environment interaction in a Markov decision process.

time step t, the agent will observe the current state of the environment s_t and then make an action a_t. For a_t, the environment gives a reward r_t indicating whether a_t is good or bad. Moreover, based on a_t, the state of the environment changes, causing the agent to observe a new state s_{t+1} at the next time step. The interaction between the environment and the agent will form a sequence of states, actions, and rewards, which is called a trajectory:

$$\tau = (s_1, a_1, r_1, s_2, a_2, r_2, s_3, a_3, r_3, \cdots) \tag{1}$$

The finite-horizon undiscounted return is the sum of rewards obtained in a trajectory:

$$R(\tau) = \sum_{t=1}^{T} r_t \tag{2}$$

where T is the final time step. The goal of reinforcement learning is to optimize the policy so as to maximize the expected return:

$$J(\pi) = \underset{\tau \sim \pi}{E} [R(\tau)] \tag{3}$$

where π represents the agent's policy.

When training dialogue policies with reinforcement learning, the user can be seen as the environment, and the dialogue system can be seen as the agent. During the interaction between the user and the system, the dialogue state tracker will give the current dialog state s_t, and the dialogue policy in the system selects the action a_t based on s_t. Then, based on the user's new utterance, the dialogue state tracker gives a new dialogue state s_{t+1}. Moreover, the user needs to give a reward r in the form of a numerical value according to whether the system completes the user's goal. However, in the actual training process, generally, only at the end of each conversation can the user give a reward r according to whether the user goal is completed. In other turns, the reward may be fixed at -1, designed to drive the system to accomplish the user goal in a shorter interaction process.

The agent's policy π determines how the agent chooses actions in each state:

$$a_t \sim \pi(\cdot|s_t) \tag{4}$$

In deep reinforcement learning, a multi-layer perceptron can be used to represent the policy π. The input of the multi-layer perceptron is the state s, and

the output is the probability of the agent performing the action a_t. For the multi-layer perceptron, its parameters are generally represented by θ. In task-oriented dialogue systems, the action space is determined in advance.

There is a class of methods called policy gradient algorithms in reinforcement learning. The basic idea of the policy gradient algorithms is to maximize $J(\pi_\theta)$ by using gradient ascent on θ, as shown in Eq. 5.

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_{\theta_k}) \tag{5}$$

Since $J(\pi_\theta)$ is an expected value and cannot be calculated directly, it needs to be estimated by using some methods. The reinforcement learning method we use, PPO, belongs to the policy gradient method, which we describe in more detail in Sect. 3.2.

3 DPPO Implementation

We first describe the workflow of our proposed DPPO algorithm and then introduce its main components in detail.

3.1 The Workflow

As shown in Algorithm 1, the workflow of DPPO can be divided into the following four stages:

- **Direct reinforcement learning** (lines 3–9): The agent interacts with a user, collects real data, and uses the real data to optimize the dialogue policy.
- **World model training** (lines 11–12): The world model is trained via supervised learning based on the data collected in the direct reinforcement learning phase.
- **Planning** (lines 13–19): The world model plays the role of the user, and the agent interacts with the world model. The simulated experience from the world model is collected and used to optimize the dialogue policy.
- **World model deactivation** (lines 20–23): DPPO monitors the agent's performance interacting with different environments (user and world model) and sets the deactivation flag f_{stop} to true if necessary.

In the initial training phase, efficient training of the agent in the DPPO algorithm can be achieved by using the low-cost simulation dialogue from the world model. When the deactivation strategy is triggered and the flag f_{stop} is set to true, the DPPO algorithm will only perform direct reinforcement learning. Thus, the negative influence of the simulated experience from the world model can be avoided.

Algorithm 1. DPPO for Dialogue Policy Learning.

Parameter: b, N, K, α

$f_{stop} \leftarrow$ **false**
for $i = 1$ to N **do**
  # Direct Reinforcement Learning
  repeat
    agent interacts with user
    for per turn, save (s, a, r, t) to D^{ppo} and save (s, a, r, t, a^u) to $D^{worldmodel}$
  until $size(D^{ppo}) \geq b$
  optimize dialogue policy based on D^{ppo}, then clear D^{ppo}
  calculate success rate r^u based on these conversations
  if not f_{stop} **then**
    # Training World Model
    optimize world model based on $D^{worldmodel}$
    # Planning
    repeat
      agent interacts with world model
      for per turn, save (s, a, r, t) to D^{ppo}
    until $size(D^{ppo}) \geq (K-1) * b$
    optimize dialogue policy based on D^{ppo}, then clear D^{ppo}
    calculate success rate r^w based on these conversations
    # Self-adaptive stopping
    if $\frac{r^w}{r^u} < \alpha$ **then**
      $f_{stop} \leftarrow$ **true**
    end if
  end if
end for

3.2 Direct Reinforcement Learning and Planning

The working mechanisms in direct reinforcement learning and planning phases are similar: the agent interacts with the environment, collects experience, and optimizes the dialogue policy. The main difference is the environment, i.e., the user and the world model. Therefore, when we discuss using PPO to optimize the dialogue policy below, we do not deliberately distinguish which environment the dialogue comes from.

The policy learning module contains two neural networks, namely the dialogue policy network π_θ and the value network V_ϕ, as shown in Fig. 6. The policy network π_θ indicates how the agent will act in the form of probability. For example, given a dialogue state s, $\pi_\theta(a_k|s) = 0.6$ means that the probability that the next action a_k taken by the agent is 0.6. The value network V_ϕ is used to estimate the value of the state s when the agent acts according to the policy network π_θ. When optimizing the policy network π_θ, the estimated value $v(s)$ calculated by V_ϕ needs to be used.

We treat task-completion dialogue as a Markov decision process which consists of a sequence of $<state, action, reward>$. Specifically, at each step, the agent observes the dialogue state s, then chooses an action a according to the

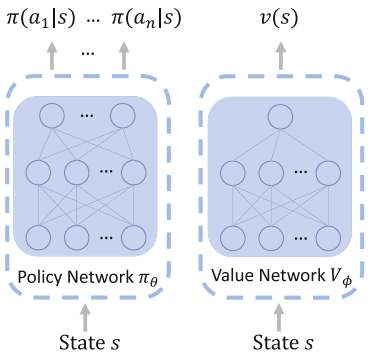


Fig. 6. Two networks in the Policy Learning module.

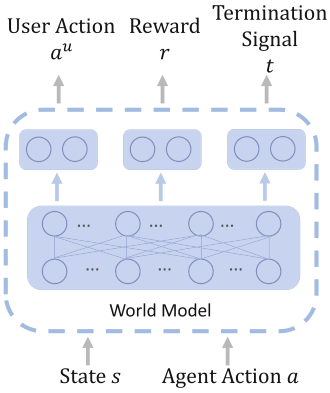


Fig. 7. The world model architecture.

policy π_θ. Afterward, the user or the world model responds with an action a^u and gives a reward r. The DST in the dialogue system updates the dialogue state s to s' according to a^u. From the perspective of the agent, a conversation of length T can be expressed as $\{(s_1, a_1, r_1), (s_2, a_2, r_2), \cdots, (s_T, a_T, r_T)\}$ where the triple $(s_t, a_t, r_t)(t \in [1, T])$ represents a turn of dialogue. These dialog data are all saved in a buffer D^{ppo}.

Given these data, we update the policy π_θ via

$$\theta = \arg\max_{\theta} \underset{s,a\sim\pi_{\theta_{old}}}{E} [\min(L_1, L_2)], \tag{6}$$

Here L_1 and L_2 are given by

$$L_1 = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} \hat{A}^{\pi_{\theta_{old}}}(s, a) \tag{7}$$

$$L_2 = clip(\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}, 1-\epsilon, 1+\epsilon)\hat{A}^{\pi_{\theta_{old}}}(s, a) \tag{8}$$

where θ_{old} is the parameters of the policy network before the update, ϵ is a hyper-parameter (usually $\epsilon \approx 0.2$), and $\hat{A}^{\pi_{\theta_{old}}}(s, a)$ is an estimator of the advantage function.

Let $r(\theta)$ denote the probability ratio $\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}$. L_1 and L_2 are very similar, except that the first term in L_2 is $clip(r(\theta), 1-\epsilon, 1+\epsilon)$ instead of $r(\theta)$. $clip(r(\theta), 1-\epsilon, 1+\epsilon)$ clip $r(\theta)$ into the range $[1-\epsilon, 1+\epsilon]$.

We use the general advantage estimation method [18] to calculate the estimator $\hat{A}_t^{\pi_{\theta_{old}}}(s, a)$ of the advantage function at timestep t:

$$\hat{A}_t^{\pi_{\theta_{old}}}(s, a) = \sum_{l=1}^{T-t} (\gamma\lambda)^{l-1} \delta_{t+l-1}^{v} \tag{9}$$

where

$$\delta_t^v = \begin{cases} -v(s_t), & t = T; \\ -v(s_t) + r_t + \gamma v(s_{t+1}) & t \in [1, T-1]. \end{cases} \tag{10}$$

in which $\gamma \in [0, 1]$ and $\lambda \in [0, 1]$ are hyperparameters.

For the value network V_ϕ, its loss is:

$$loss_v = (v(s_t) - v_t^{targ})^2 \tag{11}$$

where

$$v_t^{targ} = \begin{cases} r_t, & t = T; \\ r_t + \gamma v(s_{t+1}) & t \in [1, T-1]. \end{cases} \tag{12}$$

3.3 World Model Training

The world model we used is the same design as that of DDQ [15]. Specifically, it is implemented as a multi-task deep neural network (as shown in Fig. 7). In each dialogue turn, the world model takes the current dialogue state s and the last agent action a as input and outputs the predicted user action a^u, reward r, and termination signal t. Its calculation formula is as follows:

$$\begin{aligned} h &= tanh(W_h(s, a) + b_h) \\ a^u &= softmax(W_a h + b_a) \\ r &= W_r h + b_r \\ t &= sigmoid(W_t h + b_t) \end{aligned} \tag{13}$$

where (s, a) is the concatenation of s and a, and W and b are the trainable parameters of the world model.

In the direct reinforcement learning stage, the dialogue data (s, a, r, t, a^u) between the agent and the user will be saved in the buffer $D^{worldmodel}$. In the stage of world model training, we use the data in $D^{worldmodel}$ to train the world model so that it can predict user behavior. In the planning stage, the agent interacts with the world model and the simulated dialogue data is collected to optimize the policy network π_θ.

3.4 World Model Deactivation

Due to the complexity of user behavior, the world model may not simulate user behavior well. When using simulation data from the world model to train the agent, the inconsistency between the world model and the user's behavior will hurt the agent's performance. To address this issue, we propose the deactivation strategy in DPPO to stop training with the world model.

The world model deactivation strategy is designed based on the success rate which is the ratio of the number of successfully completed dialogues to the total number of dialogues. Specifically, we express the user goal as $G = (C, R)$ [17], where C is a set of constraint and R is a set of requests. Taking movie ticket

booking as an example, the constraints specified by the user might be the name and date of the movie, while the requests could be the location of the movie theater and the start time of the movie. When the movie ticket is booked and the booked movie ticket meets the user's constraints, a dialogue is considered successful.

We use r^w to represent the success rate of the dialogue between the agent and the world model and r^u to denote the success rate of the dialogue between the agent and the user. From the agent's point of view, the inconsistency between the world model and the user can be reflected in the difference between r^w and r^u. Therefore, we use $\frac{r^w}{r^u}$ to represent the similarity between the world model and the user. Assuming that the world model can accurately predict the user's behavior, the expected value of $\frac{r^w}{r^u}$ is close to 1. However, due to the actual prediction deviation of the world model, $\frac{r^w}{r^u}$ will deviate from 1. Our deactivation strategy is that when $\frac{r^w}{r^u} < \alpha$, we set the flag f_{stop} to true to stop using the world model in subsequent training.

4 Experiments and Results

4.1 Dataset and User Simulator

For comparison with DDQ, we use the exact same dataset and user simulator as in paper [15].

Dataset. The original raw data in a movie-ticket booking scenario was collected via Amazon Mechanical Turk and then annotated based on a schema defined by domain experts. This annotation schema contains 11 intents and 16 slots. The dataset contains 280 conversations, and their average length is 11 turns.

User Simulator. Since the agent needs to interact with the user many times when training and the interaction cost of the real user is too high, we use a user simulator [10] to simulate the user's interaction with the agent in the experiment. For each agent's action, the user simulator will reply with a simulated user action. At the end of the dialogue, the user simulator will calculate a reward value based on whether the dialogue task is completed. At the end of the dialogue, if the task is completed successfully, the reward value is $2 * L$; if the task fails, the reward value is $-L$. L is the maximum length of the dialogue, set to 40 in the experiment. In order to encourage the agent to complete the task in a short conversation, the agent will receive a reward of -1 on each turn.

4.2 Baselines

In order to benchmark the performance of DPPO, we used several algorithms to train task-oriented dialogue agents:

– **DDQ**: The Deep Dyna-Q [15] algorithm, which uses DQN to optimize the agent and uses a continuously optimized world model to generate simulation data to expand the number of conversations.

- **D3Q**: The Discriminative Deep Dyna-Q [21] algorithm (a variant of DDQ), which trains a discriminator based on the idea of Generative Adversarial Network (GAN) to select a high-quality part from a large number of simulated data for training the agent.
- **PPO**: The Proximal Policy Optimization [19] algorithm, a policy-based RL algorithm using a clipping mechanism to ensure the stability of the training process.
- **DPPO**: Our Dyna Proximal Policy Optimization algorithm. The algorithm also uses the world model to expand the number of conversations. However, unlike DDQ, it uses PPO to optimize the agent and uses a deactivation strategy to control when to stop training with the world model.
- **DPPO w/o stop**: Except for not using the deactivation strategy, everything else is the same as DPPO.

For DDQ, the hidden layers of the Q-value network are two fully connected layers, and the hidden layer size is 80. The activation functions of the hidden layer are all tanh. Reward discount coefficient γ is 0.9. The ϵ-greedy strategy is used when selecting actions during training, where ϵ is 0.1. The agent's experience pool size for both user experience and simulation experience is 5000. The world model in DDQ uses two shared hidden layers and three hidden layers related to specific tasks. The parameter K in planning is set to 5. The size of these layers is 80.

For PPO and DPPO, the hidden layers of their policy and value networks are two fully connected layers with a size of 64 and an activation function of relu. According to the default parameter values of PPO in the code Baselines [1], the hyperparameter ϵ is set to 0.2, γ is set to 0.99, and λ is set to 0.95. In each stage of direct reinforcement learning, they need to collect 1024 turns of data. The configuration of the world model is the same as the configuration in DDQ. The parameter K in planning is also set to 5. In planning, the maximum length of a simulated dialogue is 40. Moreover, the experimental results reported in this paper are the average of five runs, and the random number seeds used in these five runs are all different.

4.3 Results

We conduct several experiments: (1) To find an appropriate value of α in the world model deactivation strategy, a parameter tuning experiment on α for DPPO is conducted; (2) To verify whether DPPO can achieve a higher task success rate, a comparative study among DPPO, DDQ and D3Q is performed; (3) To verify whether DPPO can use real conversation data more effectively, we compare DPPO with PPO; (4) To verify whether the world model deactivation strategy is useful, we perform an ablation study for DPPO.

[1] Openai baselines https://github.com/openai/baselines.

Tuning for α. Figure 8 shows the learning curves of several agents trained with DPPO when the parameter α takes different values, and Table 1 is the specific value at some time in Fig. 8. We propose DPPO not only to pursue a high final dialogue success rate, but also to hope that the algorithm can perform well when it can only interact with the environment a small number of times. In Fig. 8, each agent interacts with the environment the same number of times for each epoch in training. Around the 500th epoch, the performance of all agents tends to be stable, and we regard the performance of each agent at this moment as the final performance that the agent can achieve. At the 100th and 150th epochs, the number of interactions between the agent and the environment is still relatively small. We use the agent's performance at these moments to measure the algorithm's performance at low interaction times.

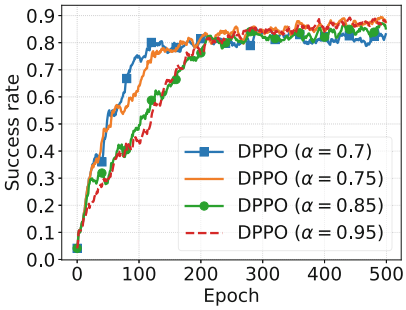


Fig. 8. Performance of DPPO with different α in the deactivation strategy.

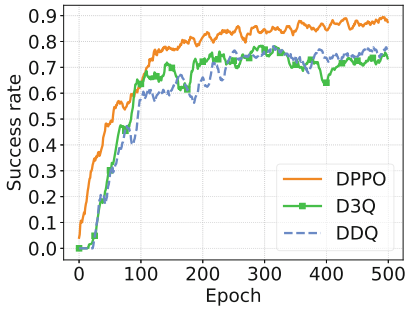


Fig. 9. DPPO vs. DDQs.

Table 1. The performance of DPPO when the parameter α in the world model deactivation strategy takes different values.

Agent (DPPO)	100th epochs			150th epochs			Final performance		
	Success	Reward	Turns	Success	Reward	Turns	Success	Reward	Turns
$\alpha = 0.70$	75.09%	40.46	21.28	76.81%	43.94	18.46	82.42%	52.28	15.24
$\alpha = 0.75$	64.16%	25.67	24.64	76.94%	42.56	21.53	86.93%	56.65	17.32
$\alpha = 0.85$	49.60%	8.01	25.04	64.53%	26.94	23.00	84.89%	54.6	16.55
$\alpha = 0.95$	43.76%	0.05	26.94	65.39%	28.43	22.08	87.51%	57.69	16.63

From Table 1, it can be found that at the 100th and 150th epoch, the dialogue success rates of DPPO ($\alpha = 0.7$) and DPPO ($\alpha = 0.75$) both reach 64% and 76%, while the dialogue success rates of DPPO ($\alpha = 0.85$) and DPPO ($\alpha = 0.95$)

are more than 10% lower. When considering final performance, the final dialogue success rate of DPPO ($\alpha = 0.75$) is 4% higher than that of DPPO ($\alpha = 0.7$). Therefore, we consider $\alpha = 0.75$ to be a suitable parameter, and in subsequent experiments, we set α for DPPO to be 0.75.

DPPO Vs. DDQs. Figure 9 and Table 2 show the results of the comparative experiment among DPPO, DDQ, and D3Q. D3Q is a variant of DDQ, so we refer to these two algorithms collectively as DDQs. The reinforcement learning methods they use to optimize dialogue policies are all DQN. In this comparative experiment, we mainly want to compare the final performance that DPPO and DDQs can achieve. It can be seen quantitatively from Table 2 that the task success rate of DPPO is more than 10% higher than that of DDQ and D3Q.

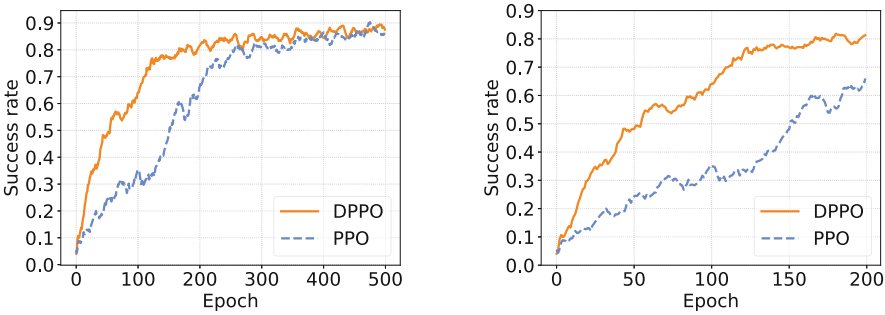
Table 2. Comparison between DPPO and DDQs.

Agent	Final performance		
	Success	Reward	Turns
DDQ	76.33%	42.41	20.37
D3Q	73.13%	41.23	15.06
DPPO	86.93%	56.65	17.32

Another thing worth noting in Fig. 9 is that at the beginning of training, both DDQ and D3Q have a task success rate of 0, while DPPO has a task success rate of approximately 0.05. This is because DPPO can facilitate imitation learning to pre-train the dialog policy using human-human dialogs, but DDQ and D3Q cannot.

DPPO vs. PPO. Figure 10 compares the learning curve of DPPO and PPO. Specifically, Fig. 10a shows the final performance of DPPO and PPO, while Fig. 10b highlights the performance they can achieve with relatively few interactions with the environment.

It can be found from Table 3 that although the final dialogue success rates of PPO and DPPO are very similar, the dialogue success rates of DPPO are about 30% and 28% higher than that of PPO at the 100th and 150th epochs, respectively. This is because, compared with PPO, DPPO uses the world model to generate a large amount of simulated data during the training process to assist the training of the agent, thereby speeding up the training speed of the agent. This feature of DPPO makes DPPO more suitable for use in real environments where interaction with the environment is expensive compared to PPO.



(a) Learning curve within 500 epochs. (b) Learning curve within 200 epochs.

Fig. 10. DPPO vs. PPO.

Table 3. Comparison between DPPO and PPO.

Agent	100th epochs			150th epochs			Final performance		
	Success	Reward	Turns	Success	Reward	Turns	Success	Reward	Turns
PPO	34.97%	−12.15	30.24	48.03%	5.89	25.47	86.31%	56.38	16.39
DPPO	64.16%	25.67	24.64	76.94%	42.56	21.53	86.93%	56.65	17.32

Unlike PPO discarding all real data immediately after optimizing the dialogue policy, DPPO will use these real data to train the world model, and in the subsequent epochs, the world model can continuously generate simulated data. In this way, DPPO indirectly extends the life cycle of each batch of real data. This explains why DPPO has higher data utilization than PPO.

Ablation Study. In order to verify the effect of the world model deactivation strategy, we compare DPPO with DPPO w/o stop, and the results are shown in Fig. 11. Although DPPO w/o stop can achieve better results in the early training stage than DPPO, the final dialogue success rate performance of DPPO w/o stop is about 3% lower than DPPO. Using DPPO w/o stop is not a bad option when interacting with the environment is very expensive. But if the final performance of the agent is more important than the training cost, using DPPO is better than DPPO w/o stop.

To further analyze why the final performance of DPPO w/o stop is lower than DPPO, we show in Fig. 12 the task success rate curves of DPPO w/o stop interacting with the user and world model respectively during the training process. During the initial period of training, the rising speeds of r^u and r^w are very similar; but after the 60th epoch, the gap between them gradually increases; in the later stage, there is always an apparent gap between r^u and r^w. The interval between r^u and r^w indicates the inconsistency between the user and world model. When the agent's performance is weak, the agent may not be aware of the inconsistency. However, when the agent's performance reaches a certain

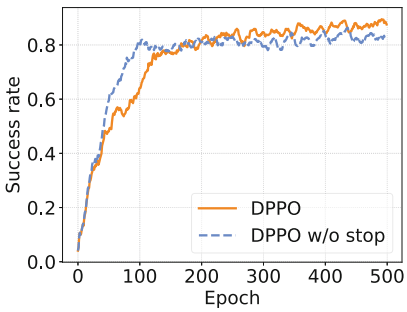


Fig. 11. DPPO vs. DPPO w/o stop.

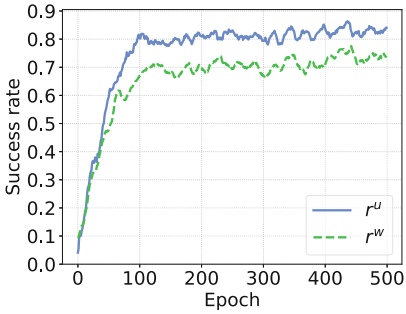


Fig. 12. r^u and r^w for DPPO w/o stop.

level, the inconsistency will hinder the further improvement of the agent. The deactivation strategy used by DPPO continuously monitors the degree of inconsistency detected by the agent. When the degree reaches a specified threshold, DPPO will no longer use the simulation data generated by the world model for training.

5 Related Work

When using RL to train dialogue policy in a dialogue system, the high cost of interacting with users is a problem that must be carefully considered. Existing work focusing on this problem can be grouped into three types, and we discuss them as follows:

The first type is to build a user simulator to completely replace the user [1,7,8,20]. The most popular user simulator is built on hand-crafted rules with a stack-like agenda based on the user goal [10,17,34]. At present, most of the work that uses RL to learn dialogue policy uses this type of user simulator [5,26,31]. However, implementing an agenda-based user simulator requires much expert knowledge. Moreover, the agenda-based user simulator is tightly coupled with the corresponding dialogue scene, making it difficult to migrate to other scenes.

The second type of work is to treat both user and system as agents and train two agents directly from human-human dialogues [11,14,23,25]. Papangelis et al. [14] train the user and system agents end-to-end. In their work, these two agents can only communicate through natural language. Takanobu et al. [23] use the actor-critic framework to train user and system agents and conduct experiments on larger datasets. However, the environment in this type of work is dynamic, which makes it more difficult for training when compared with the static environment in DDQ or our method.

The third type is to design a training framework that reduces the number of interactions between the agent and the user during training instead of eliminating the user's participation like the previous two methods. An important representative of this method is DDQ [15], where a learnable world model is introduced to the dialogue policy training process. Afterward, some work [8,29,31]

are devoted to improving DDQ. In order to control the quality of simulated experience, inspired by the generative adversarial network, Su et al. [21] trained a discriminator to filter out low-quality simulated data. Zhang et al. [31] assume that the number of user interactions available (budget) is fixed and small. To make the best use of the budget, they designed a scheduling method to control when the agent interacts with the user and when it interacts with the world model. Unlike these works, we replace DQN with PPO to overcome the inherent drawbacks of DDQ and propose a deactivation strategy to avoid the damage of the low-quality simulation experience to the agent's performance.

6 Conclusion

In this paper, we propose the Dyna Proximal Policy Optimization (DPPO) algorithm to learn dialogue policy efficiently. DPPO overcomes the drawbacks of existing methods of DDQ and its variants by (1) using Proximal Policy Optimization (PPO) to improve the training efficiency of the agent; (2) proposing a deactivation strategy to avoid the harmful effect from the low-quality simulated experience of the world model. The experimental results on the task of booking movie tickets confirm the effectiveness of our method. The idea behind the DPPO algorithm can also be extended to solve other reinforcement learning problems that are expensive to interact with the environment.

Acknowledgement. This work was supported by the Key Research Project of Zhejiang Province (2022C01145).

References

1. Asri, L.E., He, J., Suleman, K.: A sequence-to-sequence model for user simulation in spoken dialogue systems. arXiv preprint arXiv:1607.00070 (2016)
2. Chen, H., Liu, X., Yin, D., Tang, J.: A survey on dialogue systems: recent advances and new frontiers. ACM SIGKDD Explor. Newsl. **19**(2), 25–35 (2017)
3. Feng, Y., Wang, Y., Li, H.: A sequence-to-sequence approach to dialogue state tracking. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 1714–1725 (2021)
4. Firdaus, M., Golchha, H., Ekbal, A., Bhattacharyya, P.: A deep multi-task model for dialogue act classification, intent detection and slot filling. Cogn. Comput. **13**(3), 626–645 (2021)
5. Gordon-Hall, G., Gorinski, P., Cohen, S.B.: Learning dialog policies from weak demonstrations. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 1394–1405 (2020)
6. Hussein, A., Gaber, M.M., Elyan, E., Jayne, C.: Imitation learning: a survey of learning methods. ACM Comput. Surv. (CSUR) **50**(2), 1–35 (2017)
7. Keizer, S., et al.: Parameter estimation for agenda-based user simulation. In: Proceedings of the SIGDIAL 2010 Conference, pp. 116–123 (2010)

8. Kreyssig, F., Casanueva, I., Budzianowski, P., Gasic, M.: Neural user simulation for corpus-based policy optimisation of spoken dialogue systems. In: Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue, pp. 60–69 (2018)
9. Lee, C.H., Cheng, H., Ostendorf, M.: Dialogue state tracking with a language model using schema-driven prompting. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 4937–4949 (2021)
10. Li, X., Lipton, Z.C., Dhingra, B., Li, L., Gao, J., Chen, Y.N.: A user simulator for task-completion dialogues. arXiv preprint arXiv:1612.05688 (2016)
11. Liu, B., Lane, I.: Iterative policy learning in end-to-end trainable task-oriented neural dialog models. In: 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 482–489. IEEE (2017)
12. Lu, K., Zhang, S., Chen, X.: Goal-oriented dialogue policy learning from failures. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 2596–2603 (2019)
13. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
14. Papangelis, A., Wang, Y.C., Molino, P., Tur, G., Uber, A.: Collaborative multi-agent dialogue model training via reinforcement learning. In: 20th Annual Meeting of the Special Interest Group on Discourse and Dialogue, p. 92 (2019)
15. Peng, B., Li, X., Gao, J., Liu, J., Wong, K.F.: Deep Dyna-Q: integrating planning for task-completion dialogue policy learning. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 2182–2192 (2018)
16. Peng, B., et al.: Few-shot natural language generation for task-oriented dialog. In: Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 172–182 (2020)
17. Schatzmann, J., Thomson, B., Weilhammer, K., Ye, H., Young, S.: Agenda-based user simulation for bootstrapping a POMDP dialogue system. In: Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers, pp. 149–152 (2007)
18. Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P.: High-dimensional continuous control using generalized advantage estimation. arXiv e-prints arXiv:1506.02438, June 2015
19. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
20. Shi, W., Qian, K., Wang, X., Yu, Z.: How to build user simulators to train RL-based dialog systems. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 1990–2000 (2019)
21. Su, S.Y., Li, X., Gao, J., Liu, J., Chen, Y.N.: Discriminative deep Dyna-Q: robust planning for dialogue policy learning. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 3813–3823 (2018)
22. Sutton, R.S.: Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In: Machine Learning Proceedings 1990, pp. 216–224. Elsevier (1990)
23. Takanobu, R., Liang, R., Huang, M.: Multi-agent task-oriented dialog policy learning with role-aware reward decomposition. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 625–638 (2020)

24. Teng, D., Qin, L., Che, W., Zhao, S., Liu, T.: Injecting word information with multi-level word adapter for Chinese spoken language understanding. In: ICASSP 2021–2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 8188–8192. IEEE (2021)
25. Tseng, B.H., Dai, Y., Kreyssig, F., Byrne, B.: Transferable dialogue systems and user simulators. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 152–166 (2021)
26. Wang, H., Peng, B., Wong, K.F.: Learning efficient dialogue policy from demonstrations through shaping. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 6355–6365 (2020)
27. Wang, H., Wong, K.F.: A collaborative multi-agent reinforcement learning framework for dialog action decomposition. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 7882–7889 (2021)
28. Wen, T.H., Gasic, M., Mrksic, N., Su, P.H., Vandyke, D., Young, S.: Semantically conditioned lstm-based natural language generation for spoken dialogue systems. arXiv preprint arXiv:1508.01745 (2015)
29. Wu, Y., Li, X., Liu, J., Gao, J., Yang, Y.: Switch-based active deep Dyna-Q: efficient adaptive planning for task-completion dialogue policy learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 7289–7296 (2019)
30. Zahavy, T., Haroush, M., Merlis, N., Mankowitz, D.J., Mannor, S.: Learn what not to learn: Action elimination with deep reinforcement learning. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 31. Curran Associates, Inc. (2018)
31. Zhang, Z., Li, X., Gao, J., Chen, E.: Budgeted policy learning for task-oriented dialogue systems. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 3742–3751 (2019)
32. Zhao, Y., Wang, Z., Huang, Z.: Automatic curriculum learning with over repetition penalty for dialogue policy learning, vol. 35(16), pp. 14540–14548 (2021)
33. Zhao, Y., Wang, Z., Zhu, C., Wang, S.: Efficient dialogue complementary policy learning via deep Q-network policy and episodic memory policy. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 4311–4323 (2021)
34. Zhu, Q., et al.: ConvLab-2: an open-source toolkit for building, evaluating, and diagnosing dialogue systems. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 142–149 (2020)