Guided Dialogue Policy Learning without Adversarial Learning in the Loop

Ziming Li[1], Sungjin Lee[2], Baolin Peng[3], Jinchao Li[3], Julia Kiseleva[3], Maarten de Rijke[1,4], Shahin Shayandeh[3], Jianfeng Gao[3]
[1]University of Amsterdam, [2]Amazon, [3]Microsoft Research, [4]Ahold Delhaize
{z.li, m.derijke@@uva.nl}, sungjinl@amazon.com,
{baolin.peng, jincli, julia.kiseleva, shahins, jfgao@microsoft.com}

Abstract

Reinforcement Learning (RL) methods have emerged as a popular choice for training an efficient and effective dialogue policy. However, these methods suffer from sparse and unstable reward signals returned by a user simulator only when a dialogue finishes. Besides, the reward signal is manually designed by human experts, which requires domain knowledge. Recently, a number of adversarial learning methods have been proposed to learn the reward function together with the dialogue policy. However, to alternatively update the dialogue policy and the reward model on the fly, we are limited to policy-gradient-based algorithms, such as REINFORCE and PPO. Moreover, the alternating training of a dialogue agent and the reward model can easily get stuck in local optima or result in mode collapse. To overcome the listed issues, we propose to decompose the adversarial training into two steps. First, we train the discriminator with an auxiliary dialogue generator and then incorporate a derived reward model into a common RL method to guide the dialogue policy learning. This approach is applicable to both on-policy and off-policy RL methods. Based on our extensive experimentation, we can conclude the proposed method: (1) achieves a remarkable task success rate using both on-policy and off-policy RL methods; and (2) has potential to transfer knowledge from existing domains to a new domain.

1 Introduction

Task-oriented Dialogue Systems (TDSs), such as Siri, Google Assistant, and Amazon Alexa, aim to offer users assistance with completing tasks. TDSs need dialogue policies to select appropriate actions at each dialogue step according to the current context of the conversation (Chen et al., 2017). The development of RL in robotics and other domains has brought a new view on learning dialogue policies (Williams and Young, 2007; Gašić and Young, 2014; Su et al., 2017): it allows us to train with far more data than can be feasibly collected from actual users. The aim of TDSs is to maximize positive user feedback. TDSs based on RL are amenable to training with user simulators instead of real humans (Schatzmann et al., 2007; Li et al., 2016). User simulators rely on a reward function that scores system actions given dialogue context (Peng et al., 2018b; Williams et al., 2017; Dhingra et al., 2016; Su et al., 2016).

The most straightforward way to design a dialogue reward function is to score the agent based on the dialogue status in a rule-based fashion: if the dialogue ends successfully, a large positive reward will be returned; if the dialogue fails, the reward will be a large negative value; if the dialogue is still ongoing, a small negative value will be returned to encourage shorter sessions (Peng et al., 2018b). However, the rule-based solution is inflexible as it assigns the same negative reward to all the system actions before the dialogue ends. The sparse reward makes the qualities of different actions indistinguishable. Additionally, the rule-based approaches only return a meaningful reward when dialogue finishes, which can delay the penalty for low-quality actions and a high reward for high-quality ones during the conversation itself. Liu and Lane (2018) address the difficulties listed above by employing adversarial training for policy learning by jointly training two systems: (1) a policy model that decides which action to take at each turn, and (2) a discriminator that marks if a dialogue was successful or not. Feedback from the discriminator is used as a reward to push the policy model to complete a task indistinguishably from humans. Improving upon this solution, Takanobu et al. (2019) propose to replace the discriminator with a reward function that acts at the dialogue action level and returns the reward

for the given action relying on the dialogue state, system action, and next dialogue state as its input. However, the described methods are limited to policy gradient-based algorithms, such as REINFORCE (Williams, 1992) and Proximal Policy Optimization (PPO) (Schulman et al., 2017), to alternatively update the dialogue policy and the reward model on the fly, while off-policy methods are not able to benefit from self-learned reward functions. Furthermore, the alternative training of the dialogue agent and the reward model can easily get stuck in local optima or result in mode collapse.

To alleviate the problems mentioned above, in this work we propose a new approach for training dialogue policy by decomposing the adversarial learning method into two sequential steps. First, we learn the reward function using an auxiliary dialogue state generator where the loss from the discriminator can be backpropagated to the generator directly. Second, the trained discriminator as the dialogue reward model will be incorporated into the RL process to guide dialogue policy learning and will not be updated, while the state generator is discarded. Therefore, we can utilize any RL algorithm to update the dialogue policy, including both on-policy and off-policy methods. Additionally, since the reward function is pre-trained in an offline manner, we can first infer common information contained in high-quality human-generated dialogues by distinguishing human-generated dialogues from machine-generated ones, and then make full use of the learned information to guide the dialogue policy learning in a new domain in the style of transfer learning.

To summarize, our contributions are:

- A reward learning method that is applicable to off-policy RL methods in dialogue training.
- A reward learning method that can alleviate the problem of local optima for adversarial dialogue training.
- A reward function that can transfer knowledge learned in existing domains to a new dialogue domain.

2 Related Work

RL methods (Peng et al., 2017; Lipton et al., 2018; Li et al., 2017; Su et al., 2018; Dhingra et al., 2016; Williams et al., 2017; Li et al., 2019), have been widely utilized to train a dialogue agent by interacting with users. The reward used to update the dialogue policy is usually from a reward function predefined with domain knowledge and it could become very complex, e.g., in the case of multi-domain dialogue scenarios. To provide the dialogue policy with a high quality reward signal, Peng et al. (2018a) proposed to make use of the adversarial loss as an extra critic in addition to shape the main reward function. Inspired by the success of adversarial learning in other research fields, Liu and Lane (2018) learns the reward function directly from dialogue samples by alternatively updating the dialogue policy and the reward function. The reward function is a discriminator aiming to assign a high value to real human dialogues and a low value to dialogues generated by the current dialogue policy. In contrast, the dialogue policy attempts to achieve higher reward from the discriminator given the generated dialogue. Following this solution, Takanobu et al. (2019) replaces the discriminator with a reward function a reward function that acts at the dialogue action level, which takes as input the dialogue state, system action, and next dialogue state and returns the reward for the given dialogue action.

The key distinction of our work compared to previous efforts is being able to train dialogue agents with both: (1) off-policy methods in adversarial learning settings; (2) the on-policy based approaches while avoiding potential training issues, such as mode collapse and local optimum. We propose to train (1) reward model and (2) dialogue policy *consecutively*, rather than *alternatively* as suggested in (Liu and Lane, 2018; Takanobu et al., 2019). To train the reward model, we introduce an auxiliary generator that is used to explore potential dialogue situations. The advantage of our setup is the transfer from SeqGAN (Yu et al., 2017) to a vanilla GAN (Goodfellow et al., 2014). In SeqGAN setup, the policy gradient method is essential to deliver the update signal from the discriminator to the dialogue agent. In contrast, in the vanilla GAN, the discriminator can directly backpropogate the update signal to the generator. Once we restore a high-quality reward model, we update the dialogue agent using common RL methods, including both on-policy and off-policy.

3 Learning Reward Functions

In this section, we introduce our method to learn reward functions with an auxiliary generator.

3.1 Dialogue State Tracker

We reuse the rule-based ConvLab dialogue state tracker (Lee et al., 2019) to keep track of the information emerging in the interactions, including the informable slots that show the constraints given by users and requestable slots that indicates what users request. A belief vector is maintained and updated for each slot in every domain.

Dialogue State The collected information from the dialogue state tracker is used to form a structured state representation $state_t$ at every time step t. The final representation is formed by (1) the embedded results of returned entities for a query, (2) the availability of the booking option with respect to a given domain, (3) the state of informable slots, (4) the state of requestable slot, (5) the last user action, and (6) the repeated times of the last user action. The final state representation S is an binary vector with 392 dimensions.

Dialogue Action Each atomic action is a concatenation of domain name, action type and slot name, e.g., *Attraction_Inform_Address*, *Hotel_Request_Internet*. Since in the real scenarios, the response from a human or a dialogue agent can cover combination of atomic actions, we extract the most frequently used dialogue actions from the human-human dialogue collections to form the final action space – A. For example, *[Attraction_Inform_Address, Hotel_Request_Internet]* is regarded as a new action that the policy agent can execute. The final size of A is 300. We utilize one-hot embeddings to represent the actions.

3.2 Exploring Dialogue Scenarios with an Auxiliary Generator

We aim to train a reward function that has the ability to distinguish high-quality dialogues from unreasonable and inappropriate ones. To generate negative samples, we use an auxiliary generator *Gen* to explore the possible dialogue scenarios that could happen in real life. The dialogue scenario at time t is a pair of a dialogue state s_t and the corresponding system action a_t. The dialogue state-action pairs generated by *Gen* are fed to the reward model as negative samples. During reward training, the reward function can benefit from the rich and high-quality negative instances generated by the advanced generator Gen to improve the discriminability. Next, we will explain how states and actions are simulated, and our setup for adversarial leaning.

3.2.1 Action Simulation

To simulate the dialogue actions, we use a Multilayer Perceptron (MLP) as the action generator Gen_a followed by a Gumbel-Softmax function with 300 dimensions, where each dimension corresponds to a specific action from the defined A. The Gumbel-Max trick (Gumbel, 1954) is commonly used to draw a sample u from a categorical distribution with class probabilities p:

$$u = one\_hot(\arg\max_i[g_i + \log p_i]) \quad (1)$$

where g_i is independently sampled from Gumbel $(0, 1)$. Since the $\arg\max$ operation is not differentiable, no gradient can be backpropagated through u. Instead, we employ the soft-argmax approximation (Jang et al., 2016) as a continuous and differentiable approximation to $\arg\max$ and to generate the k-dimensional sample vector y following:

$$y_i = \frac{\exp((\log(p_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(p_j) + g_j)/\tau)} \quad (2)$$

for $i = 1, \ldots, k$. When the temperature $\tau \to 0$, the $\arg\max$ operation is exactly recovered but the gradient will vanish. In contrast, when τ goes up, the Gumbel-Softmax samples are getting similar to samples from a uniform distribution over k categories. In practice, τ should be selected to balance the approximation bias and the magnitude of gradient variance. In our work, p corresponding to the output distribution of generator Gen_a and k equals to the action dimension 300.

3.2.2 State Simulation

Compared to the GANs scenarios in computer vision, the output of the generator in our setting is a discrete vector which makes it challenging to backpropogate the loss from discriminator to the generator directly. To address this problem, we propose to project the discrete representation x in the expert demonstrations to a continuous space with an encoder *Enc* from a pre-trained variational autoencoder (Kingma and Welling, 2013). We assuming the human-human dialogue state s is generated by a latent variable z_{vae} via the decoder *Dec* $p(s|z_{vae}; \psi)$. Then we can regard the variable z_{vae} as a desired representation in a continuous space. Given a human-generated state s, the VAE utilizes a conditional probabilistic encoder Enc to infer z_{vae} as follows:

$$z_{vae} \sim Enc(s) = q_\omega(z_{vae}|s), \quad (3)$$

where ω and ψ are the variational parameters encoder and decoder respectively. The optimization objective is given as:

$$L_{vae}(\omega, \psi) = \underbrace{\mathbb{E}_{z_{vae} \sim q_\omega(z_{vae}|s)}[\log p_\psi(s|z_{vae})]}_{(1)} + \underbrace{KL(q_\omega(z_{vae}|s)||p(z_{vae}))}_{(2)}, \quad (4)$$

where (1) is responsible for encouraging the decoder parameterized with ψ to learn to reconstruct the input x; (2) is the KL-divergence between the encoder distribution $q_\omega(z_{vae}|s;\omega)$ and a standard Gaussian distribution $p(z_{vae}) = N(0, I)$.

The benefit of projecting the state representations to a new space is directly simulating the dialogue states in the continuous space S_{embed} similar to generating realistic images in computer vision. Besides, similar dialogue states are embedded into close latent representations in the continuous space to improve the generalizability. Figure 1 shows the overall process of learning the state projecting function $Enc_\omega(s)$ given dialogue states from real human-human dialogues. We use s_{real} to denote the continuous representation of real state s while s_{sim} for the simulated one.

3.2.3 Adversarial Training

We can approximate the real state-action distribution in a differentiable setup (1) by applying Gumbel-Softmax to simulate actions a_{sim}; and (2) by directly generating simulated states s_{sim} in the continuous space S_{embed}. The auxiliary generator Gen_θ to simulate s_{sim} and a_{sim} has following components:

$$\begin{aligned} h &= MLP_1(z_{sa}) \\ a_{sim} &= f_{Gumbel}(MLP_2(h)) \\ s_{sim} &= MLP_3(h) \\ (s,a)_{sim} &= s_{sim} \oplus a_{sim} \end{aligned} \quad (5)$$

where θ denotes all the parameters in the generator and $\oplus$ is the concatenation operation. During the adversarial training process, the generator Gen_θ takes noise z_{sa} as input and outputs a sample $(s,a)_{sim}$ and it aims to get higher reward signal from the discriminator D_ϕ. The training loss for the generator Gen_θ can be given as:

$$L_G(\theta) = -\mathbb{E}_{(s,a)_{sim} \sim Gen_\theta}(R_\phi((s,a)_{sim}), \quad (6)$$

where $R_\phi((s,a)_{sim}) = -\log(1 - D_\phi((s,a)_{sim})$ and D_ϕ denotes the discriminator measuring the reality of generated state-action pairs $(s,a)_{sim}$.

The discriminator D_ϕ in this work is a MLP

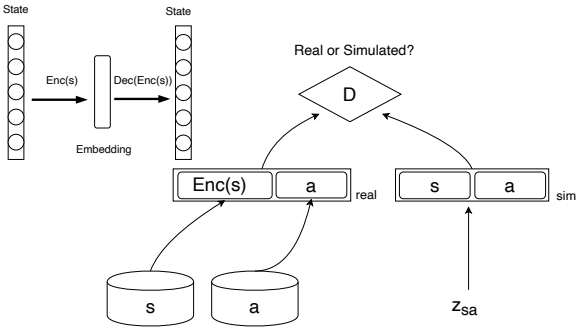

Figure 1: The architecture to simulate state-action representations with a variational autoencoder. z_{sa} is the sampled Gaussian noise.

that takes as input the state-action pair (s,a) and outputs the probability $D(s,a)$ that the sample is from the real data distribution. Since the discriminator's goal is to assign higher probability to the real data while lower scores to simulated data, the objective can be given as the average log probability it assigns to the correct classification. Given an equal mixture of real data samples and generated samples from the generator Gen_θ, the loss function for the discriminator D_ϕ is:

$$\begin{aligned} L_D(\phi) = & \\ & \mathbb{E}_{((s,a)_{sim}) \sim Gen_\theta}(\log(1 - D_\phi((s,a)_{sim}))) \\ & - \mathbb{E}_{(s,a) \sim data}(D_\phi(Enc_\omega(s), a)_{real})). \end{aligned} \quad (7)$$

After the adversarial training is finished, we will keep the discriminator D_ϕ as the reward function for future dialog agent training while the generator Gen_θ will be discarded.

Next, we discuss a suitable experimental environment for validating the presented method.

4 Experiemntal Setup

4.1 Dataset and Training Environment

MultiWOZ (Budzianowski et al., 2018) is a multi-domain dialogue dataset spanning 7 distinct domains,[1] and 10,438 dialogues. The main scenario in this dataset is that a dialogue agent is trying to satisfy the demand from tourists such as booking a restaurant or recommending a hotel with specific requirements. The average number of turns is 8.93 and 15.39 for single and multi-domain dialogues, respectively.

ConvLab (Lee et al., 2019) is an open-source multi-domain end-to-end dialogue system platform offering the annotated MultiWOZ dataset and associated pre-trained reference models. We reuse the rule-based dialogue state tracker from

[1] Attraction, Hospital, Police, Hotel, Restaurant, Taxi, Train.

ConvLab to track the information that emerges during interactions between users and the dialogue agent. Besides, an agenda-based (Schatzmann et al., 2007) user simulator is embedded and used for multi-domain dialogue scenarios.

Evaluation metrics Before a conversation starts, a user goal will be randomly sampled. The user goal consists of two parts: (1) the constraints on different domain slots or booking requirements, e.g., *Restaurant_Inform_Food=Thai*; (2) the slot values that show what the user is looking for, e.g., *Restaurant_Request_phone=?*. We The task is completed successfully, if a dialogue agent has provided all the requested information and made a booking according to the requirements. We use *average turn* and *success rate* to evaluate the efficiency and level of task completion of dialogue agents.

4.2 Architecture and Training Details

Variational AutoEncoder The encoder is a two-layer MLP that takes the discrete state representation (392 dimensions) as input and outputs two intermediate embeddings (64 dimensions) corresponding to the mean and the variance, respectively. For inference, we regard the mean μ as the embedded representation for a given state input s.

Auxiliary Generator The auxiliary generator takes randomly sampled Gaussian noise as input and outputs a continuous state representation and a one-hot action embedding. The input noise is fed to a one-layer MLP first followed by the state generator Gen_s and action generator Gen_a. Gen_s is implemented with a two-layer MLP which output is the simulated state representation (64 dimensions) corresponding to the input noise. The main component of Gen_a is a two-layer MLP followed by a Gumbel-Softmax function. The output of the Gumbel-Softmax function is an one-hot representation (300 dimensions). Specifically, we implemented the 'Straight-Through' Gumbel-Softmax Estimator (Jang et al., 2016) and the temperature for the function is set to 0.8.

Discriminator The discriminator is a three-layer MLP that takes as input the concatenation of latent state representation (64 dimensions) and one-hot encoding of the action (300 dimensions). During adversarial training, the real samples come from the real human dialogues in the training set while the simulated samples have three different sources. The main source is the output of the auxiliary generator introduced above. The second one is a random sample of state-action pairs from the training set where the action in each pair is replaced with a different one to build a simulated state-action pair. As a third source, we keep a history buffer with size $10k$ to record the simulated state-action pairs from the generator, where the state-action pairs are replaced randomly by the newly generated pairs from the generator. To strengthen the reward, we incorporate the human feedback r_{Human} into the pre-trained reward function. As the final reward function to train the dialogue agent we use the mixed reward $r_{GAN\text{-}VAE} = r_{Human} + \log(D(s, a))$.

4.3 Reinforcement Learning Methods

In this work, we validate our pre-trained reward using two different types of RL methods: Deep Q-network (DQN) (Mnih et al., 2015), which is an off-policy RL algorithm, and PPO (Schulman et al., 2017), which is a policy-gradient-based RL method. To speed up the training speed, we extend the vanilla DQN to WDQN, where the real dialogue state-action pairs from the training set are used to warm up the dialogue policy at the very beginning and then gradually removed from the training buffer. We implemented the DQN and PPO algorithms according to the ConvLab RL module.[2]

4.4 Baselines

The handcrafted reward function r_{Human} is defined at the conversation level as follow: if the dialogue agent successfully accomplish the task within T turns, it will receive $T * 2$ as reward; otherwise, it will receive $-T$ as penalty. T is the maximum number of dialogue turns. T is set 40 for experimentation. Furthermore, the dialogue agent will receive -1 as intermediate reward during the dialogue to encourage shorter interactions.

In terms of DQN-based methods, we have *DQN(Human)* trained with r_{Human} and *DQN(GAN-VAE)* trained with $r_{GAN\text{-}VAE}$. We also develop a variant *DQN(GAN-AE)* by replacing the variational autoencoder in *DQN(GAN-VAE)* with an vanilla autoencoder. With respect to WDQN, we provide three different dialogue agents trained with reward functions from *Human*, *GAN-AE*, and *GAN-VAE*.

In terms of PPO-based methods, we implemented Generative Adversarial Imitation Learning

[2]The code of our work is available at https://github.com/cszmli/dp-without-adv

(GAIL) (Ho and Ermon, 2016) and Adversarial Inverse Reinforcement Learning (AIRL) (Takanobu et al., 2019). In *GAIL*, the reward is provided with a discriminator where its parameter will be updated during the adversarial training process. *AIRL* is an adversarial learning method as well. The difference is that the discriminator in *GAIL* is replaced with a reward function that acts at the action level, which takes as input the dialogue state, system action, and the next state and returns the reward for the given dialogue action. For a fair comparison, both the *GAIL* discriminator and the *AIRL* reward model have been pre-trained. We also utilize teacher-forcing (Bengio et al., 2015) for human dialogues to stabilize the adversarial training process.

Next, we report the average performance by running the same method 8 times with different random seeds.

5 Experimental Results

5.1 Results with DQN-based agents

Figure 2 plots the results of DQN-based methods with different reward functions but the same user simulator. The dialogue policy trained with *GAN-VAE* shows the best performance in terms of convergence speed and success rate. In comparison with *GAN-VAE* and *GAN-AE*, the updating signal from the handcrafted reward function r_{Human} can still guide the dialogue policy to a reasonable performance but with a slower speed. This suggests that denser reward signals could speed up the dialogue policy training. Moreover, the policy with r_{Human} converges to a lower success rate compare to *GAN-VAE* and *GAN-AE*. It suggests that, to some extent, the pre-trained reward functions have mastered the underlying information to measure the quality of given state-action pairs. The knowledge that the reward function learned during the adversarial learning step could be generalized to unseen dialogue states and actions to avoid a potential local optimum. In contrast, the dialogue agent *DQN(Human)* only relies on the final reward signal from the simulator at the end of dialogue, which cannot provide enough guidance to the ongoing turns during conversations. This could be the reason why *DQN(Human)* shows lower success rate compare to *DQN(GAN-VAE)* and *DQN(GAN-AE)*. The representation quality of the learned state embeddings leads to higher *GAN-VAE* performance over *GAN-AE*, because *VAE* generalizes better thereby bringing more benefits to the reward functions.

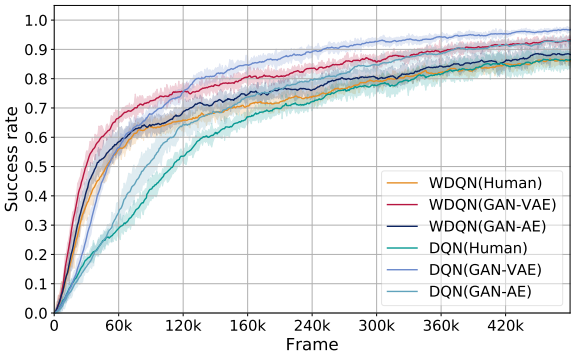

Figure 2: The learning process of DQN-based dialogue agents with different reward functions.

Examining closer WDQN agents, we can see all three methods achieve their inflection points after the first $30k$ frames. Comparing *DQN(Human)* and *WDQN(Human)*, we found the real human-human generated dialogue pairs from training set do alleviate the problem of sparse reward provided by r_{Human} at the start stage of policy training. Similar results could be observed from agents trained with the pre-trained reward function $r_{GAN\text{-}VAE}$. After $24k$ frames, the *WDQN(Human)* curve coincides in position with *DQN(Human)* and they converge to the same point in the end. The faster convergence speed on *WDQN(Human)* did not bring a higher success rate because the dialogue policy still has no access to precise intermediate reward signals for the ongoing dialogue turns.

Dialogue agent	Success Rate	Average Turn
WDQN$_{keep}$(Human)	0.741	19.144
WDQN$_{keep}$(GAN-AE)	0.879	15.118
WDQN(Human)	0.906	13.580
WDQN(GAN-AE)	0.911	13.298
WDQN(GAN-VAE)	0.937	12.260
DQN(Human)	0.870	14.960
DQN(GAN-AE)	0.953	12.300
DQN(GAN-VAE)	**0.985**	**11.040**

Table 1: The final performance of DQN-based dialogue agents with different reward functions.

Table 1 reports the final performance of different dialogue agents during test time. All the agents have been trained with $500k$ frames and we save and evaluate the model that has the best performance during the training stage. Interestingly, *DQN(GAN-VAE)* outperforms *WDQN(GAN-VAE)* while *WDQN(Human)* beats *DQN(Human)*. The warming-up stage in *WDQN(GAN-VAE)* does im-

prove the training speed but it resulted in a lower final success rate. The potential reason is that the real human-human dialogue can bring a strong update signal at the beginning of the training process but at the same time limits the exploration ability of the agent. To verify this finding, we designed two more WDQN agents: *WDQN$_{keep}$(Human)* and *WDQN$_{keep}$(GAN-AE)*, which keep expert dialogues examples during the entire training phase, rather than removing them gradually. Their performance is shown in Table 1. As to agents trained with r_{Human}, there is a huge performance gap, *WDQN(Human)* outperforms *WDQN$_{keep}$(Human)* almost by 15%. The difference in the performance of *WDQN$_{keep}$(GAN-AE)* and *WDQN(GAN-AE)* is significantly smaller because the pre-trained reward function brings more precise and consistent update signals that are explored and disclosed during the adversarial training step.

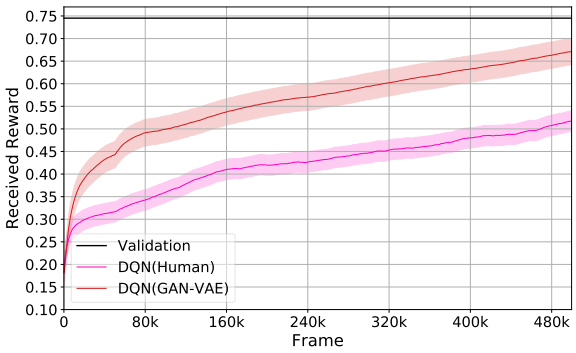

Figure 3: The reward returned by the pre-trained reward function during dialogue policy training.

Figure 3 shows curves presenting the reward changes during the RL training. The curve *Validation* denotes the average reward received based on the real human-human dialogues, which can be regarded as the human performance evaluated by the pre-train reward function $r_{GAN\text{-}VAE}$ and it is ~ 0.74.[3] For *DQN(Human)* and *DQN(GAN-VAE)* training, we feed generated in real-time dialogue batches to reward function $r_{GAN\text{-}VAE}$. We can see that both approaches are getting a high reward, but *DQN(GAN-VAE)* is growing faster, because $r_{GAN\text{-}VAE}$ is used for the training of *DQN(GAN-VAE)*. That is a promising finding since we can suggest that a well-trained reward function can be utilized not only to guide the dialogue policy training but also to judge the quality of different agents.

[3]Ideally, the reward on human dialogues should be equals to 0.5 because the discriminator is not able to distinguish the simulated dialogues from real human-human ones after generator and discriminator converge according to Eq. 7.

5.2 Results with PPO-based agents

As for *GAIL* and AIRL, the reward functions are updated on the fly, and therefore we can only employ policy gradient-based RL algorithms. We use PPO algorithms to train the dialogue agent with different reward functions. Before initiating training, we first warm-up all the dialogue agents with human dialogues via imitation learning. As a result, the warmed-up agents share similar success rates which is $\sim 33\%$. We also pre-train discriminators in *GAIL* and reward models in *AIRL* utilizing positive examples from the training set and negative examples from the pre-trained dialogue agents.

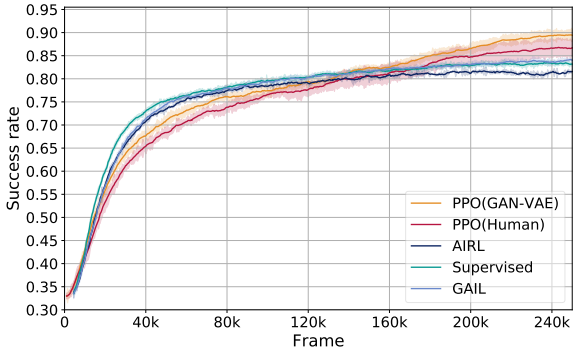

Figure 4: The learning process of PPO-based dialogue agents with different reward functions.

Figure 4 demonstrates that in terms of success rate *GAIL* and *AIRL* rise faster than *PPO(GAN-VAE)* and *PPO(Human)* during first $120k$ frames. Then both methods flattened and converged at $\sim 81\%$. It is important to note, that we utilize teacher-forcing in the adversarial step by feeding human-human dialogues to the agents every several frames while training *GAIL* and *AIRL*. Due to the large task action space, it is nearly impossible to successfully train a high-quality dialogue agent without teaching-forcing steps in adversarial learning methods. The agent called *supervised* represents the setup where we discard the training signals from the discriminators or the reward models in *GAIL* and *AIRL* and only train the policy network using teacher-forcing with the same frequency. We can observe that the adversarial training signal in *GAIL* and *AIRL* degenerates the performance of supervised learning methods.

5.2.1 Discussion

We explored various parameters for *GAIL* and *AIRL* setups, unfortunately unsuccessful. The potential reason is ConvLab has 300 actions, and it

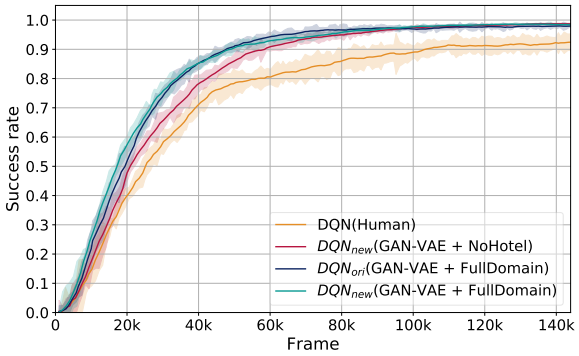

Figure 5: The learning process of dialogue agents in different domains.

is intractable for a dialogue agent to explore the action space relying only on the sparse positive reward signals which can easily lead to a local optimum. Takanobu et al. (2019) successfully applied *AIRL* to learn dialogue policy, but the considered size of action space was only half compared to our setup. More importantly, Takanobu et al. (2019) formulated dialogue policy learning as a multi-label classification task where it is easier to achieve a higher success rate by selecting as many actions as possible in one turn. Moreover, DQN-based RL algorithms are not applicable in their setup. In comparison, our agent *PPO(GAN-VAE)* can achieve higher performance in the more commonly used setup. Comparing *PPO(GAN-VAE)* and *PPO(Human)*, we can verify our claim that the dialogue agent benefits from the pre-trained reward function $r_{GAN\text{-}VAE}$. As shown in Figure 2 and Figure 4, the agents trained using the handcrafted reward function, such as *DQN(Human)* and *PPO(Human)*, share a similar final performance $\sim$ 87%. Another important finding the DQN-based agents benefit more compared to the PPO-based ones from incorporating the reward signals from the same reward function $r_{GAN\text{-}VAE}$.

5.3 Transfer learning with pre-trained reward function

To define the action space, we utilize 300 the most frequent actions from the MultiWoz dataset and use one-hot embedding to represent them. As shown in Figure 1, the action and the state representations are concatenated to form a specific state-action pair. This approach ignores the relations between different actions. For example, *Restaurant_Inform_Price* and *Restaurant_Request_People* should be close for the same conversation since they happen to be in the same domain. However, even for different domains, connections between actions are possible, e.g. *Inform_Price* and *Request_People* can also happen in the *Hotel* domain, corresponding to actions *Hotel_Inform_Price* and *Hotel_Request_People*. We ask ourselves if we can transfer the knowledge learned in existing domains to a new domain, which we have never seen before via the pre-trained reward function. To answer this question, we first reformulate the action representation as a concatenation of three different segments: *Onehot(Domain)*, *Onehot(Diact)*, *Onehot(Slot)*. Following this approach, actions containing similar information will be linked through the corresponding segments in their representation. Utilizing this formulation, we retrained our reward function in selected domains and incorporate it into the training of a dialogue agent in a new unseen domain. Concretely, we train the reward function based on the following domains: *Restaurant, Bus, Attraction, and Train*. As a testing domain, we pick *Hotel* since it has the most slot types and some of them are unique, such as *Internet, Parking, Stars*. DQN_{ori} in Figure 5 corresponds to the dialogue agent trained with all domains and the action is represented with a single one-hot embedding. By replacing the action representation in DQN_{ori} with the new action formulation we get agent – DQN_{new}. Based on the obtained results, we can conclude DQN_{new}*(GAN-VAE + NoHotel)* benefits from the reward function trained in different domains and it outperforms *DQN(Human)*. As expected, the agents DQN_{new}*(GAN-VAE + FullDomain)* and DQN_{ori}*(GAN-VAE + FullDomain)*, which are trained using reward from all domains, have better performance compared to DQN_{new}*(GAN-VAE + NoHotel)*.

6 Conclusion

In this work, we have proposed a guided dialogue policy training method without using adversarial training in the loop. First, we trained the reward model with an auxiliary generator. Then the trained reward model was incorporated into a common reinforcement learning method to guide training of a high-quality dialogue agent. By conducting extensive experimentation, we demonstrated that the proposed methods achieve remarkable performance, in terms of task success, as well as the potential to transfer knowledge from previously utilized task domains to new ones.

References

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026.

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, 19(2):25–35.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2016. Towards end-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777*.

Milica Gašić and Steve Young. 2014. Gaussian processes for pomdp-based dialogue manager optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):28–40.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Emil Julius Gumbel. 1954. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office.

Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *NIPS*, pages 4565–4573.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Sungjin Lee, Qi Zhu, Ryuichi Takanobu, Xiang Li, Yaoqin Zhang, Zheng Zhang, Jinchao Li, Baolin Peng, Xiujun Li, Minlie Huang, et al. 2019. Convlab: Multi-domain end-to-end dialog system platform. *arXiv preprint arXiv:1904.08637*.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*.

Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*.

Ziming Li, Julia Kiseleva, and Maarten de Rijke. 2019. Dialogue generation: From imitation learning to inverse reinforcement learning. In *AAAI 2019: 33rd AAAI Conference on Artificial Intelligence*, 6722–6729. AAAI.

Zachary Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. 2018. Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Bing Liu and Ian Lane. 2018. Adversarial learning of task-oriented neural dialog models. In *Proceedings of the SIGDIAL 2018 Conference*, pages 350–359.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, Yun-Nung Chen, and Kam-Fai Wong. 2018a. Adversarial advantage actor-critic model for task-completion dialogue policy learning. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6149–6153. IEEE.

Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Kam-Fai Wong. 2018b. 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)*, volume 1, pages 2182–2192.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. *arXiv preprint arXiv:1704.03084*.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. 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*, pages 149–152. Association for Computational Linguistics.

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

Pei-Hao Su, Pawel Budzianowski, Stefan Ultes, Milica Gasic, and Steve Young. 2017. Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. *arXiv preprint arXiv:1707.00130*.

Pei-Hao Su, Milica Gasic, Nikola Mrkšić, Lina M Rojas Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2431–2441.

Shang-Yu Su, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Yun-Nung Chen. 2018. Discriminative deep dyna-q: Robust planning for dialogue policy learning. In *EMNLP*.

Ryuichi Takanobu, Hanlin Zhu, and Minlie Huang. 2019. Guided dialog policy learning: Reward estimation for multi-domain task-oriented dialog. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 100–110.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274*.

Jason D Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*.