Imperfect also Deserves Reward: Multi-Level and Sequential Reward Modeling for Better Dialog Management

Zhengxu Hou[1], **Bang Liu**[2]*, **Ruihui Zhao**[1], **Zijing Ou**[3],
Yafei Liu[1], **Xi Chen**[1], **Yefeng Zheng**[1]
[1] Tencent Jarvis Lab
[2] RALI & Mila, Université de Montréal [3]Sun Yat-sen University
holmes.hzx@gmail.com bang.liu@umontreal.ca ouzj@mail2.sysu.edu.cn
{zacharyzhao,davenliu,jasonxchen,yefengzheng}@tencent.com

Abstract

For task-oriented dialog systems, training a Reinforcement Learning (RL) based Dialog Management module suffers from low sample efficiency and slow convergence speed due to the sparse rewards in RL. To solve this problem, many strategies have been proposed to give proper rewards when training RL, but their rewards lack interpretability and cannot accurately estimate the distribution of state-action pairs in real dialogs. In this paper, we propose a multi-level reward modeling approach that factorizes a reward into a three-level hierarchy: domain, act, and slot. Based on inverse adversarial reinforcement learning, our designed reward model can provide more accurate and explainable reward signals for state-action pairs. Extensive evaluations show that our approach can be applied to a wide range of reinforcement learning-based dialog systems and significantly improves both the performance and the speed of convergence.

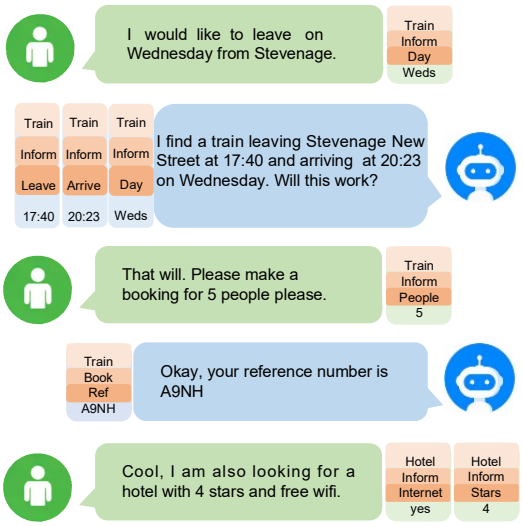


Figure 1: An example of multi-domain task-oriented dialog about train ticket booking and hotel reservation. For each utterence, we show the fine-grained actions in the form of [domain-act-slot:value].

1 Introduction

Task-oriented dialog systems have become a focal point in both academic and industrial research and have been playing a key role in conversational assistants such as Amazon Alexa and Apple's Siri. (Budzianowski et al., 2018; Wei et al., 2018; Chen et al., 2019b) Existing research on task-oriented dialog systems mainly includes pipeline and end-to-end methods (Zhang et al., 2020). For pipeline-based systems, usually could be divided into four components: Natural Language Understanding (NLU), Dialog State Tracking (DST), Dialog Management (DM), and Natural Language Generation (NLG). The modular structure makes the systems more interpretable and stable than end-to-end systems, which directly take natural language context as input and output a response.

In pipeline-based dialog systems, DM is a core component that is responsible for modeling the current belief state (structured information about the current situation) and deciding the next action. Due to its non-differentiable nature, many researchers resort to Reinforcement Learning (RL) to learn a DM (Peng et al., 2017; Casanueva et al., 2018; Chen et al., 2019a; Vu, 2019). However, RL suffers from the problems of low data efficiency and slow convergence speed in dialog management due to reward sparsity and huge action space (Takanobu et al., 2019; Liu and Lane, 2018; Fang et al., 2019). To solve these problems, existing research designs reward models to estimate how similar a state-action pair is to an expert trajectory. Liu and Lane (2018) and Takanobu et al. (2019) combines a Generative Adversarial Network (GAN) with RL to acquire a reward model which could give rewards in turn/dialog level. However, introducing GAN will bring other problems like model instability. To solve this, Li et al. (2020) proposed to train a discriminator and directly use it as a fixed reward

*Corresponding author.

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2993–3001
June 6–11, 2021.

estimator during RL training. However, there is no thorough evaluation to analyze the performance of their reward estimator. Besides, researchers ignore the huge action space of RL that has a huge impact on RL's converging speed.

In this paper, we propose to interpret a state-action pair from a multi-level and sequential perspective, instead of simply classifying them as "right" or "wrong". Fig. 1 shows an example of multi-domain ("Train" and "Hotel") task-oriented dialog to illustrate our idea. For each utterance, we infer the domain, act, and slot of it to form a three-level hierarchy, leading to more accurate and interpretable modeling of the dialog agent's actions. For example, for the utterance "Okay, your reference number is A9NH", the RL agent books a train ticket of A9NH. We infer the domain of the action "book" is "train", and the slot is "Ref" (reference number) with slot value "A9NH". An RL agent will get rewards from the action only if it belongs to an appropriate domain, and will get slot rewards only if it takes suitable action. For example, if the RL agent in Fig. 1 chooses the wrong action "Train-Book-Ref: A9NH" at the first turn (i.e., direct booking without confirmation from the user), it will only receive rewards for domain, since it should take the action "inform" instead of "book".

To construct a multi-level reward model, we propose the following designs. First, we utilize a disentangled autoencoder to factorize and encode a dialog state into three independent latent sub-states, characterizing domain, act, and slot, respectively. Correspondingly, we also design an action decomposer to decompose an action into three sub-actions, taking the first user action in Figure 1 as an example, the decomposer will decompose action "Train-Inform-Day:Weds" into "Train", "Inform" and "Day". Second, we learn a multi-level generator-discriminator framework. The generator generates sub-states and sub-actions from noises, and the discriminator learns to classify whether a sub state-action pair is real or generated. In this way, the learned discriminator can give rewards to a state-action pair in terms of domain, act, and slot. Lastly, we impose Markov property to our multi-level rewards by only rewarding an act/slot if the prior domain/act is appropriate. Such design also alleviates the problem of huge action decision space in RL, as the "domain-act-slot" hierarchy restricts the choice of act/slot when the domain/act has been decided.

We run extensive evaluations to test our multi-level sequential reward model by incorporating it into a variety of RL-based agents. The experimental results demonstrate that our reward model can significantly improve the performance of RL agents and accelerate the speed of convergence.

2 Related Work

Dialog reward modeling aims to give a proper reward to the action made by an RL agent. Traditional hand-crafted rule-based reward modeling requires expert knowledge and cannot handle unseen actions or situations. Su et al. (2015) proposes a reward shaping method to speed up online policy learning, which models the sum of all rewards in turn level.

After that, most researchers tend to exploit GAN by considering an RL agent as a generator and a reward function as a discriminator. Liu and Lane (2018) first introduces the adversarial method for reward computation. It learns a discriminator which can give the probability of authenticity in the dialog level. Takanobu et al. (2019) further expands the adversarial method by inferring a user's goal and giving a proper reward in turn level. However, adding adversarial training to RL will bring potential drawbacks, as training RL is different from normal GAN training whose dataset is fixed, which needs training with the environment and simulated user which is changing all the time. Thus RL and discriminator are training with a moving target rather than a fixed object. It is hard to supervise the adversarial training of generator and discriminator due to no solid feedback. Besides, as claimed in (Li et al., 2020), such adversarial training is only suitable for policy gradient-based methods like Proximal Policy Optimization (PPO), but not working for value-based RL algorithm like Deep Q-Network (DQN).

Recently, Li et al. (2020) utilizes a generator to approximate the distribution of expert state-action pairs and trains a discriminator to distinguish them from expert state-action pairs. By introducing a pretraining method, this approach can be extended to both on-policy and off-policy RL methods. However, it is still confused that whether this reward model could give correct rewards. Different from the aforementioned methods, in this paper, our model generates rewards in a more accurate sequential and multi-level manner.

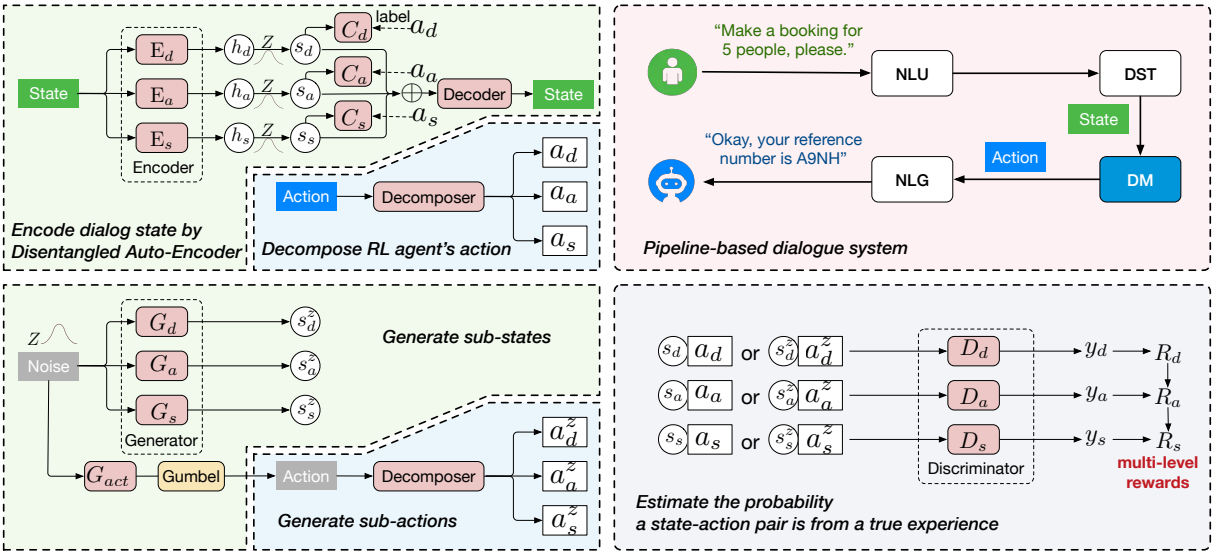


Figure 2: An overview of our multi-level and sequential reward modeling mechanism.

3 Method

We propose a novel multi-level sequential reward estimator and learn it under an Inverse Reinforcement Learning (IRL) framework. IRL aims to learn a reward estimator based on expert data, which are state-action pairs $(\mathcal{S}_e, \mathcal{A}_e)$ from expert policy. IRL could be formally defined as:

$$\mathbb{E}_{\pi^*_\theta(\mathcal{A}_e|\mathcal{S})} R^*(\mathcal{A}_e, \mathcal{S}) \geq \mathbb{E}_{\pi_\theta(\hat{\mathcal{A}}|\mathcal{S})} R^*(\hat{\mathcal{A}}, \mathcal{S}) \quad (1)$$

where the goal is to find an optimal reward function R^*, such that based on the same states $\mathcal{S}$, expert dialog policy π^*_θ will obtain equal or higher rewards than the agent's policy π_θ. We denote expert action and agent action as A_e and $\hat{A}$, respectively.

Our objective is approximating R^* by capturing the distribution of expert dialog f_e and estimating how likely a state-action pair is from f_e as the reward. To accurately model the expert distribution f_e, we disentangle f_e into three levels: domain distribution f^e_d, action distribution f^e_a, and slot distribution f^e_s. Fig. 2 shows the framework of our multi-level reward estimator. Given a state-action pair from the input and output of a DM module in a pipeline-based system, we combine three components to estimate its quality. First, we acquire sub-states and sub-actions by utilizing a Disentangled Auto-Encoder (DAE) to encode states and a rule-based decomposer to decompose actions. Second, we learn different sub-generators to generate sub-states and sub-actions from noises. Third, we train different sub-discriminators to classify whether a state-action pair is from expert data or agent policy. Besides, we sequentially connect the three discriminators, imposing Markov property to the multi-level rewards, as well as alleviating the problem of huge action space in RL. Finally, the discriminators can serve as reward estimators for domian, action, and slot during inference. Algorithm 1 summarizes the training process of our model components. We introduce more details in the following.

3.1 State-Action Decomposition and Representation

We first decompose an action into sub-actions and learn to decompose and encode a state into sub-states from domain, act, and slot level.

For action decomposition, we decompose an action $\mathcal{A}$ by rules based on how the action vector is defined. Such a rule-based decomposer can be easily implemented by first defining an assignment matrix M, then multiply M with $\mathcal{A}$ and select three sub-spans of $\mathcal{A}$ to form three sub-actions a_d, a_a and a_s, which are all one-hot vectors.

For state decomposition and representation, we decompose a discrete state $\mathcal{S}$ into sub-states of domain, act, and slot, and learn a continuous representation of them by DAE. As shown in Fig. 2, the DAE contains three encoders E_d, E_a and E_s to extract and encode the sub-states from $\mathcal{S}$:

$$[h_d; h_a; h_s] = Encoder(\mathcal{S}). \quad (2)$$

To enforce each encoder learn the sub-state corresponding to domain, act and slot respectively, we adopt three auxilary classifiers (C_d, C_a and C_s) which classify each sub-state representation (s_d, s_a

and s_s) with the corresponding sub-action (a_d, a_a and a_s) as label. To enhance model generalization, we inject data-dependent noises into latent variables h_d, h_a and h_s. In particular, a noise variance σ^2 is obtained via the Multilayer Perceptron (MLP) transformation from state $\mathcal{S}$: $\log \sigma^2 = MLP(\mathcal{S})$. Then we sample noise z from a Gaussian distribution $\mathcal{N}(h, \sigma^2 I)$. The reparameterization trick (Kingma and Welling, 2013) is further exploited to achieve end-to-end training:

$$[s_d; s_a; s_s] = h + \sigma\epsilon, \quad \epsilon \sim N(0, I). \tag{3}$$

In this way, the sub-state representations are different for every training time of input state $\mathcal{S}$, and thus the model is provided with additional flexibility to explore various trade-offs between noise and environment.

Next, we reconstruct the state via a decoder:

$$\hat{\mathcal{S}} = Decoder([s_d; s_a; s_s]). \tag{4}$$

After that, since the state $\mathcal{S}$ is a discrete vector, we can learn DAE by minimizing a binary cross entropy loss:

$$\mathcal{L}_{dec} = \sum_{i=1}^{|\mathcal{S}|} \left[\mathcal{S}_i \log(\hat{\mathcal{S}}_i) + (1 - \mathcal{S}_i) \log(1 - \hat{\mathcal{S}}_i) \right]. \tag{5}$$

The loss for the auxilary classifiers are:

$$\mathcal{L}^i_{enc} = -\frac{exp(s_i^T W_i a_i)}{\sum\limits_{a_j \in \mathcal{A}_i} exp(s_i^T W_i a_j)}, \; i \in [d, a, s], \tag{6}$$

where $W_i \in \{W_d, W_a, W_s\}$ is the learnable parameters of the classifiers and $\mathcal{A}_i$ is the action space of the corresponding action level. Therefore, the overall loss for training DAE is given by:

$$\mathcal{L}_{DAE} = \mathcal{L}_{dec} + \sum_{i \in [d,a,s]} \mathcal{L}^i_{enc}. \tag{7}$$

3.2 Adversarial Learning of State-Action Distribution

Different from previous adversarial training methods in which generator (policy) and discriminator (reward estimator) are trained alternatively when interacting with a simulated user, our GAN network (Goodfellow et al., 2014) is trained offline without the need of simulated users. As shown in Fig. 2, our discriminator $\mathcal{D}$ is composed of three sub-discriminators$\{D_d, D_a, D_s\}$, our generator $\mathcal{G}$ consists a set of parallel sub-generators $\{G_d, G_a, G_s, G_{act}\}$ with the same Gaussian noise Z as input to generate sub-states $\{s^z_d, s^z_a, s^z_s\}$ and actioin $\hat{\mathcal{A}}$. Then $\hat{\mathcal{A}}$ is decomposed into $\{a^z_d, a^z_a, a^z_s\}$ by the same rule-based decomposer we described in Sec. 3.1. As a true action $\mathcal{A}$ is discrete, we use Straight-Through Gumbel Softmax (Jang et al., 2016) to approximate the sampling process. The generators aim to approximate the distribution of expert dialog $(\mathcal{S}_e, \mathcal{A}_e)$ by learning distributions $\{f^e_d, f^e_a, f^e_s\}$ of sub state-actions with $\{G_d, G_a, G_s\}$ and G_{act}. We train the generators by the following loss:

$$L_{\mathcal{G}}(\theta) = \mathbb{E}_{z \sim p(z)}(\log(1 - \mathcal{D}(\mathcal{G}(z)))), \tag{8}$$

where θ represents the parameters of generator G.

For discriminator, it consists of three paralleled and independent MLP networks with a sigmoid output layer. The discriminator outputs three scores $\{y_d, y_a, y_s\}$ that respectively denote the probability a sub state-action pair is from a true expert distribution. The traininig loss could be written as:

$$L_{D_i} = -[\mathbb{E}_{(s_i,a_i) \sim f^e_i} \log D_{\phi_i}(s_i, a_i) + \mathbb{E}_{z \sim p(z)}(1 - \log D_{\phi_i}(s^z_i, a^z_i))], i \in [d, a, s]. \tag{9}$$

Algorithm 1 Reward Estimator Training

Require: Expert dialog $[\mathcal{S}_e : \mathcal{A}_e]$
  repeat
    Training DAE by Eq. 7
  until DAE converge
  Initialize $\mathcal{G}_\theta$, $\mathcal{D}_\phi$ with random weights θ, ϕ
  repeat
    for g-steps **do**
      Sample noise samples z from Gaussian prior $p(z)$
      Update θ by Eq. 8
    end for
    for d-steps **do**
      Generate $s^z_d, s^z_a, s^z_s, \hat{\mathcal{A}}_z$ by $\mathcal{G}_\theta(z)$
      Decompose $\hat{\mathcal{A}}$ into a^z_d, a^z_a, a^z_s
      Sample $s_d, s_a, s_s, \mathcal{A}$ from DAE
      Decompose $\mathcal{A}$ into a_d, a_a, a_s
      Update ϕ by Eq. 9
    end for
  until GAN converges

3.3 Reward Shaping and Combination

Reward shaping provides an RL agent with extra rewards in addition to the original sparse rewards r_{ori} in a task to alleviate the problem of reward sparsity. We follow the same assumption of (Liu and Lane, 2018; Paek and Pieraccini, 2008), in which state-action pairs similar to expert data will receive higher rewards.

The rewards from our discriminators are calculated as:

$$\begin{aligned} R_d &= y_d, \\ R_a &= y_a \cdot \text{Sigmoid}(\tau(R_d + b)), \\ R_s &= y_s \cdot \text{Sigmoid}(\tau(R_a + b)), \end{aligned} \tag{10}$$

where $\{y_d, y_a, y_s\}$ are the outputs of discriminators $\{D_d, D_a, D_s\}$ in Fig. 2. Note that we impose Markov property into multi-level reward calculation by taking the reward of domain/act level into account when calculating the reward of act/slot level. An agent will receive a low reward when it chooses a wrong domain even if y_a or y_s is high. We accomplish this by the $sigmoid$ functions in Eq. 10. τ and b are two hyper-parameters controlling the shape of the $sigmoid$ function. A smaller τ will introduce a softer penalty given by prior-level reward.

After getting the three-level rewards, we propose two reward integration strategies. The first strategy we denote as R_{SeqPrd} is simply using R_S from Eq. 10 as the combined reward. This strategy will bring reward to nearly 1 or 0. The second strategy we denote as R_{SeqAvg} is computing the mean of the three rewards $\{R_d, R_a, R_s\}$ as the final reward. Finally, we augment the original reward r_{ori} by adding R_{SeqPrd} or R_{SeqAvg} for reward shaping.

3.4 Details of Modeling and Training

For the Disentangled Auto-Encoder, the input of its encoder is binary states $\mathcal{S}$. We use three paralleled MLP layers with same hidden size 64 as the sub-encoders to get hidden states $\{h_d, h_a, h_s\}$, which are the same with the architecture of the MLP network for generating noise variance σ^2. We train the encoder, decoder, and classifier network simultaneously.

For the generator part, we utilize four independent and parallel MLP layers. All layers share the same gaussian noise as input. The first three aim to capture the distribution in the field of d, a, s with output size = 64. The output size of G_4 is 300 with an output layer of ST-Gumbel Softmax. To make the output of generators be similar to the encoding representation of DAE and bring noise to the discriminator as well, we further add two MLP networks separately after generation layer to simulate the sampling process of mean and variance. We add weight regularization in a form of $l2$ norm to avoid overfitting. In our experiments, the generator is weaker compared to the discriminator, therefore we set the training frequency ratio of generator and discriminator to be 5:1.

For the discriminator part, we utilize three parallel MLP layers followed by a sigmoid function as the output layer. Training a multi adversarial network is not easy. Three discriminators will be insensitive to their own field if training all $\mathcal{G}$ and $\mathcal{D}$ jointly. Thus, we train $\mathcal{G}$ and $\mathcal{D}$ in the following way. $\mathcal{D}$ takes all outputs from $\mathcal{G}$ as input, but only chosen sub-generator and sub-discriminator pairs have gradient backpropagation, and others are frozen. During the experiment, we found start training from one pair to two pairs than to all pairs brings good results.

4 Experiments

4.1 Experimental Setup

Dataset We run evaluations based on the MultiWOZ dataset (Budzianowski et al., 2018)[1] . It is a multi-domain dialog dataset that constructed from human dialog records, mainly ranging from restaurant booking to hotel recommending scenarios. There are 3,406 single-domain dialogs and 7,032 multi-domain dialogs in total. The average number of turns is 8.93 and 15.39 for single and multi-domain dialogs, respectively.

Platform We implement our methods and baselines based on the Convlab platform (Lee et al., 2019)[2]. It is a multi-domain dialog system platform supporting end-to-end system evaluation, which integrates several RL algorithms.

Implementation Details For fair comparisons, we follow the same experiment settings in (Li et al., 2020). Specifically, an agenda-based user simulator (Schatzmann et al., 2007) is embedded and exploited to interact with dialog agent. We set the training environment to a "dialog-act to dialog-act (DA-to-DA)" level, where the agent interacts with a simulated user in a dialog act way rather than an utterance way. We use a rule-based dialog state

[1] https://github.com/budzianowski/multiwoz
[2] https://github.com/sherlock1987/SeqReward

tracker (DST) to track 100% of the user goals. We train on millions of frames (user-system turn pairs) with 392-dimensional state vectors as inputs and 300-dimensional action vectors as outputs. For all the RL networks, we use a hidden layer of 100 dimensions and ReLU activation function.

Evaluation metrics During the evaluation, the simulated user will generate a random goal first for each conversation and then complete the session successfully if the dialog agent has accomplished all user requirements. We exploit *average turn*, *success rate* and *reward score* to evaluate the efficiency of proposed reward model. In particular, the *reward score* metric is defined as

$$\textit{reward score} = \begin{cases} -T + 80, & \text{if success} \\ -T - 40, & \text{if fail} \end{cases} \quad (11)$$

where T denotes the number of system-user turns in each conversation session. The performances averaged over 10 times with different random seeds are reported as the final results. Besides, we evaluate our RL models in every $1,000$ frame (system-user turn) by using $1,000$ dialogs interacting with a simulated user.

4.2 Baselines

We evaluate the proposed reward estimator via two classical RL algorithms: i) Deep Q-Network (DQN) (Mnih et al., 2015), which is a value-based RL algorithm; ii) Proximal Policy Optimization (PPO) (Mnih et al., 2015), which is a policy-based RL algorithm.

In terms of the DQN-based methods, we compare our method DQN$_{SeqAvg}$ and DQN$_{SeqPrd}$ (corresponding to R_{SeqAvg} and R_{SeqPrd}, respectively) with DQN$_{vanilla}$, whose reward function is defined in Eq. 11, and DQN$_{offgan}$ (Li et al., 2020), which also pretrains an reward function to achieve performance gains. Similarly, we also evaluate on Warm-up DQN (WDQN) with different reward function, named WDQN$_{vanilla}$, WDQN$_{offgan}$, WDQN$_{SeqAvg}$ and WDQN$_{SeqPrd}$, respectively.

For the implementation details of DQN-based agents, we use $\epsilon-$greedy action exploration and set a linear decay from 0.1 in the beginning to 0.01 after $500k$ frames. We train DQN on 500 batches of size 16 every 200 frames. Besides, we use a relay buffer of size $50,000$ to stabilize training.

In terms of the PPO-based methods, we pick up two adversarial methods: i) Guided Dialog Policy Learning (AIRL) (Takanobu et al., 2019); and ii) Generative Adversarial Imitation Learning (GAIL) (Ho and Ermon, 2016). AIRL works on turn level and gives reward scores based on state-action-state triple (s_t, a_t, s_{t+1}). For GAIL, it works on dialog level and gives rewards after dialog ends.

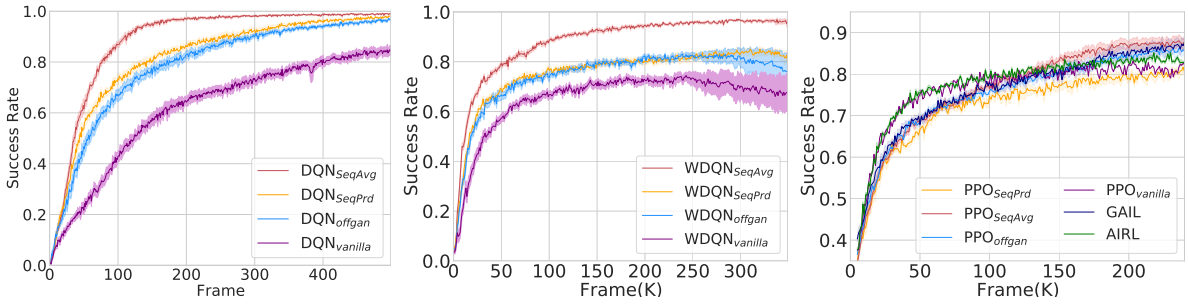
Similar to DQNs, we also compare our methods with PPO$_{vanilla}$ and PPO$_{offgan}$ (Li et al., 2020). There is one extra hyperparameter named training epoch for GAIL and AIRL, which represents the training ratio of discriminator and PPO models. Here we set it to 4. Apart from these, all the other hyperparameters stay the same. Different from the settings for DQN, the $\epsilon-$greedy stays 0.001 without decay. Besides, we set val-loss-coef to be 1, meaning no discount for value loss. We also set the training frequency to be 500 frames.

4.3 Results with DQN-based Agents

From Fig. 3 (a), DQN$_{SeqPrd}$ achieves the best performance with a success rate of 0.990 and converges after 130K, which speeds up the training process by almost 300% compared to DQN$_{vanilla}$. Compared with DQN$_{vanilla}$, the methods using pre-trained reward functions R_{offgan}, R_{SeqArg}, R_{SeqPrd} are better than vanilla in terms of both convergence speed and success rate. This phenomenon suggests that these three reward estimators could speed up dialog policy training.

Different from DQN$_{offgan}$, whose reward function is also learned by adversarial training, we further apply disentangled learning and multi-view discriminator to obtain fine-grained rewards. The performance of DQN$_{SeqPrd}$ and WDQN$_{SeqPrd}$ gains received in convergence speed and final performance of our methods confirm the superiority of the hierarchical reward.

For WDQN agent, since first warmed up with human dialogs, the WDQN-based methods share a similar success rate (around 6%) before training and consistently converge faster than DQN-based models. However, the usage of warm-up operation will mislead the model to local optimum and deteriorate the final success rate. This phenomenon can be found in the last 100 frames, the performances of WDQN$_{vanilla}$ and WDQN$_{offgan}$ drop significantly. Another attractive property of our method, compared with WDQN$_{vanilla}$ and WDQN$_{offgan}$, is the variance of success rate is obviously small, which strongly supports the remarkable benefit of exploiting disentangled representation to learn prof-



(a) DQN agents results (b) WDQN agents results (c) PPO agents results

Figure 3: The training process of dialog agents based on different reinforcement learning algorithms.

Model	SRate	RScore	ATurn
DQN$_{vanilla}$	0.843	55.18	8.02
DQN$_{offgan}$	0.964	71.87	5.86
DQN$_{SeqAvg}$	0.981	74.21	5.57
DQN$_{SeqPrd}$	**0.990**	**75.43**	**5.40**
WDQN$_{vanilla}$	0.678	32.53	10.84
WDQN$_{offgan}$	0.760	43.88	9.33
WDQN$_{SeqAvg}$	0.798	40.01	8.79
WDQN$_{SeqPrd}$	**0.960**	**71.05**	**6.09**
AIRL	0.793	50.63	10.14
GAIL	0.832	51.65	10.01
PPO$_{vanilla}$	0.861	56.93	8.30
PPO$_{offgan}$	0.860	56.96	8.20
PPO$_{SeqPrd}$	0.806	49.86	9.34
PPO$_{SeqAvg}$	**0.879**	**59.04**	**8.07**

Table 1: The performance of dialog agents with different reward functions. SRate, Rscore and ATurn represent *success rate*, *reward score* and *average turn*.

Model	**Acc**	**Prec**	**Rec**	**F1**	**JS**
R$_{offgan}$	0.79	0.84	0.76	0.80	1.39
R$_{d}$	0.86	**0.97**	0.80	0.88	0.69
R$_{a}$	0.71	0.91	0.65	0.76	0.14
R$_{s}$	0.77	0.95	0.69	0.80	0.33
R$_{SeqAvg}$	**0.87**	0.91	0.85	**0.88**	1.00
R$_{SeqPrd}$	0.87	0.87	**0.87**	0.87	**3.73**

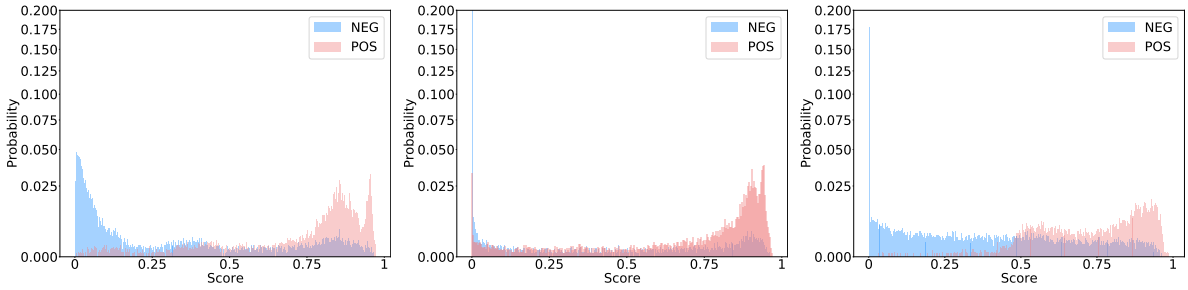
Table 2: The accuracy, precision, recall, F1 and JS divergence scores on test dataset with equal number of positive and negative samples.

itable sequential reward in dialog management.

4.4 Results with PPO-based Agents

For the policy gradient-based agents, we compare our models with two other strong baselines, *i.e.,* GAIL and AIRL, whose reward functions are updated during RL training. Similar to DQN-based methods, we employ PPO algorithms to train dialog agents with different reward functions. Before training a PPO agent, we perform imitation learning with human dialogs to warm-up PPO agents, achieving around 33% success rate. For fair comparisons, we also pretrain the discriminator in GAIL and reward model in AIRL by feeding positive samples and negative samples from pretrain process of dialog agents.

As demonstrated in Fig. 3 (c), although AIRL rises faster than others during the first 50 frames, it converges to a worse result, compared with PPO$_{SeqAvg}$. An interesting observation is that PPO$_{vanilla}$ even performs better than AIRL. This may be due to the fact that adversarial learning is extremely unstable in RL. Therefore, we aim to learn an off-line reward function to guide the evolution of agents, as we motivate in the introduction. In the comparison between PPO$_{offgan}$ and PPO$_{SeqAvg}$, the performance gains obtained by our model verifies the advantage of exploiting multi-level reward signals. Moreover, it can be seen that, in the PPO-based RL algorithm, the performance of the agent with the reward function R_{SeqPrd} is worse than that of R_{SeqAvg}, but the opposite is true in the DQN and WDQN-based methods. This may be caused by that the multiplicative reward (*i.e.,* R_{SeqPrd}) may cause the gradient to be very steep, which makes the training of the policy gradient-based model unstable. However, in the value-based RL method, an average reward (*i.e.,* R_{SeqAvg}) might degenerate the performance, as a hierarchical reward is more general and intuitive, which has access to precise intermediate reward signals. The performances of the last frame in terms of *success rate*, *reward score* and *average turn* are shown in Table 1, in which we could claim again that our method PPO$_{SeqAvg}$ outperforms all baseline models by a substantial margin.



(a) R_{SeqAvg} (b) R_{SeqPrd} (c) R_{offgan}

Figure 4: Histogram of distribution about fake/real state-action pairs in different rewards setting. Horizontal axis means the reward score, and vertical axis means frequency.

4.5 Analysis of Different Rewards

To visualize the model performance and what benefits a sequential reward will bring, we view the evaluation as a binary classification and distribution distance problem. we use accuracy, precision, recall, and F1 to find out how good this binary model is, and use JS divergence to evaluate the ability of the reward model to divide positive and negative distributions, the larger the better. We construct a test dataset with equal numbers $(7,372)$ of positive and negative samples from the test dataset. All positive samples are original state-action pairs. For negative samples, we fix states and randomly pick actions from those with different domains. We evaluate three reward models separately in Table 2. R_d is the best one among the three with the highest five scores. This is pretty straightforward since domain is the first identity to divide action space into groups. And for R_a and R_s, the JS divergence is lower, this is because some actions could have different domains with the same action-slot. For example, action "Train-Inform-Arrive" and "Hotel-Inform-Arrive" have the same action-slot with different domains. Thus, R_a and R_s will only give an ambiguous decision boundary. But from a sequential view, we make a new combination of R_{SeqAvg} and R_{SeqPrd}, which gives good results.

R_{offgan} could give the right rewards to some extent, but from Fig. 4 (c), there is a large intersection between fake and real distributions among three, which means it wrongly classifies fake action as right. And this is the reason why its F1 score is lower. Besides, this reward model is a biased model, its ratio of true negative and true positive samples is 0.89 thus it tends to give fake results. For both of our model, there is little bias, R_{SeqPrd} is 0.99 and R_{SeqAvg} is 0.98, which benefits from sequential combination.

For R_{SeqPrd}, R_{SeqAvg} and R_{offgan}, R_{SeqPrd} perform the best, no matter from the view of binary classification or JS divergence. And the distribution is much sharper than R_{SeqAvg} with prediction score centering at 0 or 1. For R_{SeqAvg}, the distribution is softer than R_{SeqPrd} as shown in Fig. 4. Although there is no exact evaluation to say how bad one action is, from the good results of PPO$_{SeqAvg}$, nearly the same binary classification score with R_{SeqPrd} as well as lower JS divergence, we could get the conclusion that it is the most accurate rewards among the three.

5 Conclusion

We propose a multi-level and sequential reward modeling mechanism that models expert state-action pairs in terms of domain, act, and slot. Our approach combines a disentangled auto-encoder and a generator-discriminator framework to model the distribution of expert state-action pairs. The learned discriminators can thereby serve as a multi-level reward estimator. Experimental results show that our three-level modeling mechanism gives more accurate and explainable reward estimations and significantly boosts the performance of a variety of RL-based dialog agents, as well as accelerating the convergence speed of training.

6 Acknowledgements

This work was supported by the Tencent Jarvis Lab. We thank Ziming Li, Zhanpeng Huang for the helpful discussions and insightful comments. Thank Kallirroi for leading me into the field of dialogue systems.

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.

Inigo Casanueva, Paweł Budzianowski, Pei-Hao Su, Stefan Ultes, Lina Rojas-Barahona, Bo-Hsiang Tseng, and Milica Gašić. 2018. Feudal reinforcement learning for dialogue management in large domains. *arXiv preprint arXiv:1803.03232*.

Lu Chen, Zhi Chen, Bowen Tan, Sishan Long, Milica Gašić, and Kai Yu. 2019a. Agentgraph: Toward universal dialogue management with structured deep reinforcement learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(9):1378–1391.

Meng Chen, Ruixue Liu, Lei Shen, Shaozu Yuan, Jingyan Zhou, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2019b. The jddc corpus: A large-scale multi-turn chinese dialogue dataset for e-commerce customer service. *arXiv preprint arXiv:1911.09969*.

Ting Fang, Tingting Qiao, and Duanqing Xu. 2019. Hagan: Hierarchical attentive adversarial learning for task-oriented dialogue system. In *International Conference on Neural Information Processing*, pages 98–109. Springer.

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.

Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *Advances in neural information processing systems*, 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*.

Ziming Li, Sungjin Lee, Baolin Peng, Jinchao Li, Shahin Shayandeh, and Jianfeng Gao. 2020. Guided dialog policy learning without adversarial learning in the loop. *arXiv preprint arXiv:2004.03267*.

Bing Liu and Ian Lane. 2018. Adversarial learning of task-oriented neural dialog models. *arXiv preprint arXiv:1805.11762*.

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):529–533.

Tim Paek and Roberto Pieraccini. 2008. Automating spoken dialogue management design using machine learning: An industry perspective. *Speech communication*, 50(8-9):716–729.

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.

Pei-Hao Su, David Vandyke, Milica Gasic, Nikola Mrksic, Tsung-Hsien Wen, and Steve Young. 2015. Reward shaping with recurrent neural networks for speeding up on-line policy learning in spoken dialogue systems. *arXiv preprint arXiv:1508.03391*.

Ryuichi Takanobu, Hanlin Zhu, and Minlie Huang. 2019. Guided dialog policy learning: Reward estimation for multi-domain task-oriented dialog. *arXiv preprint arXiv:1908.10719*.

Dirk Väth Ngoc Thang Vu. 2019. To combine or not to combine? a rainbow deep reinforcement learning agent for dialog policy. In *20th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 62.

Zhongyu Wei, Qianlong Liu, Baolin Peng, Huaixiao Tou, Ting Chen, Xuan-Jing Huang, Kam-Fai Wong, and Xiang Dai. 2018. Task-oriented dialogue system for automatic diagnosis. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–207.

Zheng Zhang, Ryuichi Takanobu, Qi Zhu, MinLie Huang, and XiaoYan Zhu. 2020. Recent advances and challenges in task-oriented dialog systems. *Science China Technological Sciences*, pages 1–17.