Rethinking Supervised Learning and Reinforcement Learning in Task-Oriented Dialogue Systems

Ziming Li[1], Julia Kiseleva[2], Maarten de Rijke[1,3]
[1]University of Amsterdam, [2]Microsoft Research, [3]Ahold Delhaize
{z.li, m.derijke@uva.nl}, julia.kiseleva@microsoft.com

Abstract

Dialogue policy learning for Task-oriented Dialogue Systems (TDSs) has enjoyed great progress recently mostly through employing Reinforcement Learning (RL) methods. However, these approaches have become very sophisticated. It is time to re-evaluate it. Are we really making progress developing dialogue agents only based on RL? We demonstrate how (1) traditional supervised learning together with (2) a simulator-free adversarial learning method can be used to achieve performance comparable to state-of-the-art (SOTA) RL-based methods. First, we introduce a simple dialogue action decoder to predict the appropriate actions. Then, the traditional multi-label classification solution for dialogue policy learning is extended by adding dense layers to improve the dialogue agent performance. Finally, we employ the Gumbel-Softmax estimator to alternatively train the dialogue agent and the dialogue reward model without using RL. Based on our extensive experimentation, we can conclude the proposed methods can achieve more stable and higher performance with fewer efforts, such as the domain knowledge required to design a user simulator and the intractable parameter tuning in reinforcement learning. Our main goal is not to beat RL with supervised learning, but to demonstrate the value of rethinking the role of RL and supervised learning in optimizing TDSs.

1 Introduction

The aim of dialogue policies in Task-oriented Dialogue System (TDS) is to select appropriate actions at each time step according to the current context of the conversation and user feedback (Chen et al., 2017). In early work, dialogue policies were manually designed as a set of rules that map the dialogue context to a corresponding system action (Weizenbaum, 1966). The ability of rule-based solutions is limited by the domain complexity and task scalability. Moreover, the design and maintenance of these rules require a lot of effort and domain knowledge.

Due to recent advantages in deep learning and the availability of labeled conversational datasets, *supervised learning* can be employed for dialogue policy training to overcome the disadvantages of rule-based systems. The downside of the supervised learning approach is that the dialogues observed in the datasets are unlikely to represent all possible conversation scenarios; in some extreme cases, the required conversational dataset cannot be collected or acquiring it might cost-prohibitive.

The success of RL in other areas holds promises for dialogue Policy Learning (PL) (Williams and Young, 2007). Using RL techniques, we can train dialogue policies and optimize automatically, from scratch and utilizing interactions with users (Gašić and Young, 2014; Su et al., 2017). In RL-based solutions, the dialogue system takes actions that are controlled by the dialogue policy, and user feedback (the *reward signal*), which is provided when the dialogue is finished, is utilized to adjust the initial policy (Peng et al., 2018b; Williams et al., 2017; Dhingra et al., 2016). In practice, reward signals are not always available and may be inconsistent (Su et al., 2016). As it is not practical to ask for explicit user feedback for each dialogue during policy training, different strategies have been proposed to design a rule-based user simulator along with a reward function that can approximate the real *reward function* which exists only in each user's mind. Designing an appropriate user simulator and accurate reward function requires strong domain knowledge. This process has the same disadvantages as rule-based dialog systems (Walker et al., 1997). The difference is that rule-based approaches to system design meet this problem at the dialogue agent side while rule-based user simulators need to solve it at the environment side.

If the task is simple and easy to solve, why not just build a rule-based system rather than a user-simulator that is then used with RL techniques to train the dialogue system, where more uncontrollable factors are involved? And if the task domain is complex and hard to solve, is it easier to design and maintain a complicated rule-based user simulator than to build a rule-based dialogue agent? Supervised learning methods do not suffer from these issues but require labeled conversational data; in some exceptional cases, if the data cannot be collected for privacy reasons, RL is the solution. However, collecting labeled data is feasible for many applications (Williams et al., 2014; Weston et al., 2015; Budzianowski et al., 2018). Therefore in this work seek to answer the following research question: *Are we really making progress in TDSs focusing purely on advancing RL-based methods?*

To address this question, we introduce three dialogue PL methods which do not require a user simulator. The proposed methods can achieve comparable or even higher performance compared to SOTA RL methods. The first method utilizes an action decoder to predict dialogue combinations. The second method regards the dialogue PL task as a multi-label classification problem. Unlike previous work, we assign a dense layer to each action label in the action space. Based on the second method, we propose an adversarial learning method for dialogue PL without utilizing RL. To backpropagate the loss from the reward model to the policy model, we utilize the Gumbel-Softmax to connect the policy model and the reward model in our third method. We compare our methods with RL and adversarial RL based dialogue training solutions to show how we can achieve comparable performance without a utilizing costly user simulator.

To summarize, our contributions are:

- A dialogue action decoder to learn the dialogue policy with supervised learning.
- A multi-label classification solution to learn the dialogue policy.
- A simulation-free adversarial learning method to improve the performance of dialogue agents.
- Achieving SOTA performance in dialogue PL with fewer efforts and costs compare to existing RL-based solutions.

2 Related Work

A number of RL methods, including Q-learning (Peng et al., 2017; Lipton et al., 2018; Li et al., 2017; Su et al., 2018; Li et al., 2020) and policy gradient methods (Dhingra et al., 2016; Williams et al., 2017), have been applied to optimize dialogue policies by interacting with real users or user simulators. RL methods help the dialogue agent is able to explore contexts that may not exist in previously observed data. A key component in RL is the quality of the reward signal used to update the agent policy. Most existing RL-based methods require access to a reward signal based on user feedback or a pre-defined one if feedback loop is not possible. Besides, designing a good reward function and a realistic user simulator is not easy as it typically requires strong domain knowledge, which is similar to the problem that rule-base methods meet. Peng et al. (2018a) propose to utilize adversarial loss as an extra critic in addition to the main reward function based on task completion. Inspired by the success of adversarial training in other NLP tasks, Liu and Lane (2018) propose to learn dialogue rewards directly from dialogue samples, where a dialogue agent and a dialogue discriminator are trained jointly. Following the success of inverse reinforcement learning (IRL) in different domains, Takanobu et al. (2019) employ *adversarial IRL* to train the dialogue agent. They replace the discriminator in GAIL (Ho and Ermon, 2016) with a reward function with a specific architecture. The learned reward function can provide a stable reward signal and adversarial training can benefit from high quality feedback.

Compared to existing RL based methods, we propose strategy that can eliminate designing a user simulator and sensitive parameter-tuning process while bringing a significant performance improvement with respect to a number of metrics. The absence of user simulators involved will largely reduce the required domain knowledge and supervised learning can lead to robust agent performance.

3 Multi-Domain Dialogue Agent

Dialogue State Tracker (DST) In a standard TDS pipeline, the rule-based DST is deployed to keep track of information emerging in interactions between users and the dialogue agent. The output from the Natural Language Understanding (NLU) module is fed to the DST to extract information, including informable slots about the constraints from users and requestable slots that indicate what users inquire about. In our setup, the dialogue agents and

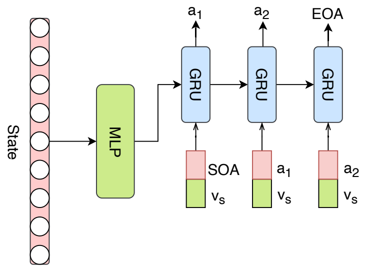

Figure 1: Architecture to approximate a dialogue policy with an action decoder.[2]

user-simulators are interacting through predefined dialogue actions therefore no NLU is involved. Besides, a belief vector is maintained and updated for each slot in every domain.

Dialogue state We formulate a structured state representation s_t according to the information resulting from the DST at time step t. There are 4 main types of information in the final representation: (1) corresponding to the embedded results of returned entities for a query, (2) the last user action, (3) the last system action, and (4) the belief state from the rule-based state tracker. The final state representation s is a vector of 553 bits.

Dialogue action We regard the dialogue response problem as a multi-label prediction task, where in the same dialogue turn, several atomic dialogue actions can be covered and combined at the same moment. In the action space, each action is a concatenation of domain name, action type and slot name, e.g. *'attraction-inform-address'*, which we call an *atomic action*.[1] Lee et al. (2019) proposes that the action space covers both the atomic action space and the top-k most frequent atomic action combinations in the dataset and then the dialogue PL task can be regarded as a single label classification task. However, the expressive power of the dialogue agent is limited and it is beneficial if the agent can learn the action structure from the data and this could lead to more flexible and powerful system responses.

4 Dialogue Policy Learning (PL)

4.1 PL as a sequential decision process

Different atomic dialogue actions contained in the same response are usually related to each other. To fully make use of information contained in co-occurrence dependencies, we decompose the multi-label classification task in dialogue PL as follows. Assuming the system response consists of two atomic actions, *'hotel-inform-address'* and *'hotel-inform-phone'*, the model takes the dialogue state as input and predict the atomic actions sequentially. The path could be described as either *'hotel-inform-address'* $\rightarrow$ *'hotel-inform-phone'* or *'hotel-inform-phone'* $\rightarrow$ *'hotel-inform-address'*. Before the training stage, the relative order of all the atomic actions will be predefined and fixed. Following this solution, we apply a GRU-based (Cho et al., 2014) decoder to model the conditional dependency between the actions in one single turn; see Fig. 1.

The proposed model first extracts state features v_s by feeding the raw state input s to an Multilayer Perceptron (MLP). In the next state, the state representation v_s will be used as the initial hidden state h_0 of action decoder GRU. To avoid information loss during decoding, the input to the action decoder is:

$$input_t = embedding(a_{t-1}) \oplus v_s. \qquad (1)$$

The starting input $input_0$ is the concatenation of starting action *SOA* and state representation v_s. a_{t-1} denotes the dialogue action in the prediction path at time step $t-1$ and *embedding(a)* returns the action embedding of the given action a. In the next steps, actions will be generated consecutively according to:

$$o_t, h_t = GRU(input_t, h_{t-1}), \qquad (2)$$

where o_t is the output of the action decoder. We use cross-entropy to train the action decoder together with the MLP for feature extracting. We use beam-search to find the most appropriate action path.

4.2 PL with adversarial learning

Next, we introduce an adversarial learning solution, *DiaAdv*, to train the dialogue policy without a user simulator along with a dialogue discriminator. Feedback from the discriminator is used as a reward signal to push the policy model to interact with users in a way that is indistinguishable from how a human agent completes the task. However, since the output of the dialogue policy is a set of discrete dialogue actions, it is difficult to pass the gradient update from the discriminator to the policy model. To cross this barrier, we propose to utilize the Gumbel-Softmax function (Jang et al., 2016) to link the discriminator to the generator. Next, we will give a brief introduction about the dialogue policy model and the dialogue discriminator. Afterwards, we will show how we can utilize Gumbel-Softmax to backpropagate the gradient.

Dialogue policy To generate dialogue actions, we

[1]There are 166 atomic actions in total in the action space.

[2]*SOA* and *EOA* are special actions corresponding to the starting signal and ending, signal respectively.

employ an MLP as the action generator Gen_{sa} followed by a set of Gumbel-Softmax functions, where each function corresponds to a specific action in the atomic action space (Fig. 2) and the output of each function has two dimensions. We first introduce how it works when there is only one Gumbel-Softmax function in the setting and then extend it to multiple function. The Gumbel-Max trick (Gumbel, 1954) is commonly used to draw samples u from a categorical distribution with class probabilities p. The process of Gen_θ can be formulated as follows:

$$p = \text{MLP}(s) \tag{3}$$

$$u = \textit{one\_hot}(\arg\max_i [g_i + \log p_i]), \tag{4}$$

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

$$y_i = \frac{\exp((\log(p_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(p_j) + g_j)/\tau)} \tag{5}$$

for $i = 1, \ldots, k$. In practice, τ should be selected to balance the approximation bias and the magnitude of gradient variance. In our case, p corresponds to the dialogue action status distribution $p(a_l^i|s)$ where $l \in \{0, \ldots, k-1\}$ and $i \in \{1, \ldots, m\}$. In our setting, k is set to 2 and each dimension denotes one specific action status, which could be 1 if selected or 0 if not selected. m is set to the size of in the action space – 166. By taking into account the multiple actions, we rewrite the sampled vector y as y_l^i where l and i denote the corresponding dialogue action status and the i_{th} atomic action in the action space respectively. The final combined action is:[3]

$$a_{fake} = y_0^1 \oplus y_1^1 \oplus \ldots \oplus y_0^{166} \oplus y_1^{166}. \tag{6}$$

Next, the generated action a_{fake} is fed to the reward model D_ω along with the corresponding state s. The dialogue policy Gen_θ aims to get a higher reward signal from the discriminator D; the training loss function for the generator Gen_θ is:

$$L_G(\theta) = -\mathbb{E}_{s, a_{fake} \sim Gen}(D_\omega(s, a_{fake})) \tag{7}$$

Dialogue reward As to the dialogue discriminator, we build a reward model D_ω that takes as input the state-action pair (s, a) and outputs the reward $D(s, a)$. Instead of using a discriminator to

[3] $Dim(a_{fake}) = 166 * 2$.

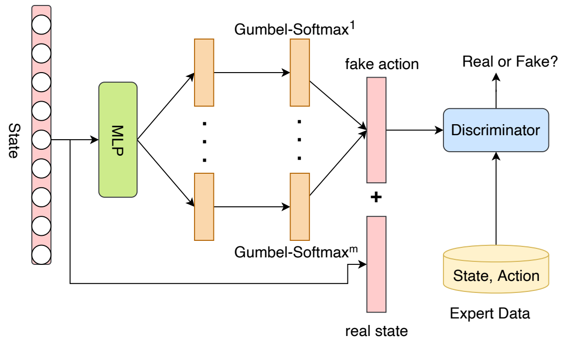

Figure 2: Architecture to approximate the dialogue policy with adversarial learning. The dialogue policy dialogue discriminator is linked to the dialogue policy through a set of Gumbel-Softmax functions.[4]

predict the probability of a generated state-action pair as being real or fake, inspired by Wasserstein GANs (Arjovsky et al., 2017), we replace the discriminator model with a reward model that scores a given pair (s, a). Since the reward model's goal assigns a higher reward to the real data and a lower value to fake data, the objective can be given as the average reward it assigns to the correct classification. Given an equal mixture of real data samples and generated samples from the dialogue policy Gen_θ, the loss function for the reward model D_ω is:

$$L_D(\omega) = -\,\mathbb{E}_{s, a_{fake} \sim Gen_\theta}(D_\omega(s, a_{fake})) \tag{8}$$

$$+\, \mathbb{E}_{s, a \sim data}(D_\omega(s, a))). \tag{9}$$

During training, the policy network and the reward model are be updated alternatively.

4.3 PL as multi-label classification with dense layers

We introduced *DiaAdv*, which can bridge the policy network and the reward model together utilizing Gumbel-Softmax functions. A by-product of this framework is the policy network with dense layers and a set of Gumbel-Softmax functions. If we discard the Gumbel-Softmax functions but keep the dense layers, we obtain a new model, *DiaMultiDense*, to solve the multi-label classification problem. Each dense layer corresponds to a specific dialogue action and the output of the dense layer has two dimensions denoting the two possible values for action status, *selected* and *not selected*. We expect the dense layers can extract informative information particularly for their corresponding actions and discard noisy information. During inference, the two possible values for the status of an action will be compared and the higher one will be the la-

[4] '+' denotes the concatenation operation.

bel for the current dialogue action. *DiaMultiDense* can be regarded as a simple but efficient state denoising method for dialogue PL with multi-label classification.

5 Experimental Setup

MultiWOZ Datasset (Budzianowski et al., 2018) is a multi-domain dialogue dataset with 7 distinct domains,[5] and $10,438$ dialogues. The main used scenario is a dialogue agent is trying to satisfy the tourists' demands such as booking a restaurant or recommending a hotel with specific requirements. Each dialogue trajectory is decomposed into a set of state-action pairs with the same TDS that is used for training. In total, we have $56,700$ dialogue state-action pairs in the training set, with $7,300$ in the validation set, and $7,300$ in the test set.

Baselines Three types of baselines are explored:
(B1): *Supervised Learning*, where the dialogue action selection task is regarded as a multi-label classification problem.
(B2): *Reinforcement Learning (RL)*, where the reward function is handcrafted and defined as follows: at the end of a dialogue, if the dialogue agent accomplishes the task within T turns, it will receive $T*2$ as a reward; otherwise, it will receive $-T$ as a penalty. T is the maximum number of turns in each dialogue; we set it to 40 in all experiments. Furthermore, the dialogue agent will receive -1 as an intermediate reward during the dialogue to encourage shorter interactions. In our experiments, we used three methods, including: *GP-MBCM* (Gašić et al., 2015), *ACER* (Wang et al., 2016), *PPO* (Schulman et al., 2017).
(B3): *Adversarial learning*, where dialogue agent is trained with a user simulator, we conduct comparisons with two methods: *GAIL* (Ho and Ermon, 2016) and *GDPL* (Takanobu et al., 2019). The dialogue agents in *GAIL* and *GDPL* are both PPO agents while these two methods have different reward models. We report the performance of *ALDM* (Liu and Lane, 2018) for completeness.

5.1 Training setup

DiaSeq With respect to *DiaSeq*, we use a two-layer MLP to extract features from the raw state representation. First, we sort the action order according to the action frequency in the training set. All action combinations in the dataset will be transferred to an action path based on the action order. Three special actions – *PAD*, *SOA*, *EOA*, corresponding to padding, start of action decoding and end of action decoding – are added to the action space for action decoder training. We use beam search to predict the action combinations and beam size is set to 6. The action embedding size is set to 30; the hidden size of the GRU is 50.
DiaAdv For the policy network of *DiaAdv*, a two-layer MLP is used to extract state features followed by 166 dense layers and Gumbel-Softmax functions consecutively. To sample a discrete action representation, we implemented the "Straight-Through" Gumbel-Softmax Estimator (Jang et al., 2016); the temperature τ for each function is set to 0.005. As to the discriminator, a three-layer MLP takes as input the concatenation of dialogue state and action, and outputs a real value as the reward for the state-action pair.
DiaMultiDense We reuse the policy network from *DiaAdv* except the Gumbel-Softmax functions.
GDPL (Takanobu et al., 2019) is reused. The policy network and value network are three-layer MLPs.
PPO The policy network in PPO shares the same architecture as *GDPL*. The difference is that the reward model is replaced with a handcrafted one.
GAIL *GAIL* shares the same policy network as *GDPL*. The discriminator is a two-layer MLP taking as input the state-action pair.
DiaMultiClass The policy network is a three-layer MLP and trained with cross entropy. It has the same architecture as the policy network in *GDPL*.
We reuse the reported performance of GP-MBCM, ACER, and ALDM from (Takanobu et al., 2019) since we share the same TDS and user simulator. The methods based on RL or adversarial learning are pre-trained with real human dialogues.[6]

5.2 Evaluation metrics

Before a conversation starts, a user goal will be randomly sampled. The sampled user goal mainly contain two parts of information. The first part is about the constraints of different domain slots or booking requirements, e.g. *'restaurant-inform-food'='Thai'*, *'restaurant-inforarea'='east'*, *'restaurant-book-people'=4* which means the user wants to book a table for 4 per-

[5]Attraction, Hospital, Police, Hotel, Restaurant, Taxi, Train

[6]The code of our work is available at `https://github.com/cszmli/Rethink-RL-Sup`

Dialogue agent	**Turn**	**Match**	**Rec**	**F1**	**Success rate**
GP-MBCM	2.99	0.44	–	0.19	28.9
ACER	10.49	0.62	–	0.78	50.8
PPO (human)	15.56	0.60	0.72	0.77	57.4
ALDM	12.47	0.69	–	0.81	61.2
GDPL	7.80	0.81	0.89	0.87	81.7
GAIL	7.96	0.81	0.87	0.86	80.5
DiaMultiClass	12.66	0.58	0.71	0.79	57.2
DiaMultiDense	9.33	0.85	0.94	0.87	86.3*
DiaSeq	9.03	0.81	0.88	0.85	81.6
DiaAdv	8.80	0.85	0.94	0.85	87.4*

Table 1: The performance of different dialogue agents, which is calculated based on the average results by running each method 5 times. * indicates statistically significant improvements ($p < 0.005$) using a paired t-test over the *GDPL* success rate and the proposed methods.

sons to have Thai food in the east area. The information contained in the second part is about the slot values that the user is looking for, such as *restaurant-request-phone=?*, *'restaurant-request-address'=?*, which means the user wants to know the phone and address of the recommended restaurant. We use *Match*, *Recall*, *F1 score* to check if all the slot constraints and requested slot information have been satisfied. *F1 score* evaluates whether all the requested information has been provided while *Match* evaluates whether the booked entities match the indicated constraints. We use *Average Turn* and *Success rate* to evaluate the efficiency and level of task completion of dialogue agents. If an agent has provided all the requested information and made a booking according to the requirements, the agent completes the task successfully.

6 Results and Discussion

6.1 Performance of different dialogue agents

Tab. 1 shows the performance of different dialogue agents. With respect to success rate, *DiaAdv* manages to achieve the highest performance by 6% compared to the second highest method *GDPL*. However, *DiaAdv* is not able to beat *GDPL* in terms of average turns. A possible reason is that *GDPL* can generate more informative and denser dialogue action combinations. With a user simulator in the training loop, the dialogue agent can explore more unseen dialogue states in the dataset. Furthermore, the same user simulator will be used to test the dialogue agent and the dialogue agent will definitely benefit from what he has explored in the training stage. However, more informative and denser responses will not guarantee all the users' requirements will be satisfied and this will lead to a lower *Match* score as shown in Tab. 1.

Dialogue agent	DiaSeq	DiaMultiClass	DiaMultiDense
#Parameters	251,000	184,000	133,000

Table 2: Total number of parameters for supervised learning models.

As to *DiaSeq*, it can achieve almost the same performance as *GDPL* from different perspectives while *GDPL* has a slightly higher *F1* score. However, the potential cost benefits of *DiaSeq* are huge since it does not require a user simulator in the training loop. The training of *DiaSeq* is well-understood and we can get rid of tuning the sensitive parameters in RL and Adversarial Learning. To sum up, *DiaSeq* is far more cost-efficient solution.

Another supervised learning method, *DiaMultiDense* achieves remarkable performance with respect to different metrics. Compared to the traditional solution *DiaMultiClass*, joining of dense layers as in *DiaMultiDense* brings a huge performance gain; it manages to beat *DiaMultiClass* on all the metrics. And it achieves higher *F1* score than *DiaAdv*. Since the only difference between *DiaMultiDense* and *DiaMultiClass* is that we replace the last layer of *DiaMultiClass* with a stack of dense layers, the change in the number of parameters may lead to the performance gap. We report the number of parameters of three supervised learning methods in Tab. 2. *DiaMultiDense* achieves the highest performance among these three methods while using the fewest parameters. We believe the dense layers have been trained to filter noisy information from the previous module and the final classification can benefit from the high-quality information flow.

6.2 User experience evaluation

Automatic metrics can only capture part of the performance difference between different dialogue agents. For example, we use success rate to reflect the level of task completion and use turn number to represent the efficiency of dialogue agents. However, the final goal of a TDS is to assist real users to complete tasks. To fully evaluate system performance while interacting with real users, we launch an evaluation task on Amazon Mturk to rate the user experience with the proposed dialogue systems. For each evaluation task, we will first present an Mturk worker with a randomly sampled user goal, which contains the constraints about specific domain slots and some slot information that the user is looking for. In the next step, according to the sampled goal, two generated dialogues

Dataset \ Agent	DiaMultiClass		DiaSeq		DiaMultiDense		GDPL		DiaAdv	
	Turn	Success rate	Turn	Success rate	Turn	Success rate	Turn	Success rate	Turn	Success rate
MultiWOZ (0.1)	17.14	31.7	10.77	70.4	18.36	27.0	9.21	21.2	16.80	37.2
MultiWOZ (0.4)	12.56	59.0	9.99	75.5	10.76	79.4	8.49	68.0	9.90	81.6
MultiWOZ (0.7)	13.1	53.6	9.35	77.2	10.02	85.1	8.10	73.3	9.30	87.0

Table 3: The performance of different dialogue agents with different amounts of expert dialogues. We only report *Average Turn* and *Success rate* here due to limited space.

Dialogue pair	**Win**	**Loose**	**Tie**
DiaMultiDense vs. GDPL	42	50	8
DiaSeq vs. GDPL	50	44	6
DiaAdv vs. GDPL	39	51	10

Table 4: Human evaluation results.

from two different dialogue agents are shown to the worker. The worker needs to pick up the dialogue agent that provides a better user experience. Different factors will be taken into account, such as response quality, response naturalness, how similar it is compared to a real human assistant. If the worker thinks two dialogue agents perform equally good/bad or it's hard to distinguish which one is better, the option 'Neutral' can be selected. Four dialogue agents are evaluated: *GDPL*, *DiaSeq*, *DiaMultiDense* and *DiaAdv*, and there are three comparison pairs *DiaMultiDense-GDPL*, *DiaSeq-GDPL*, *DiaAdv-GDPL* since *GDPL* is regarded as the SOTA method. Each comparison pair has 100 dialogue goals sampled and 200 corresponding dialogues from two different dialogue agents. All the dialogue actions in the dialogue turns are translated into human readable utterances with the language generation module from ConvLab (Lee et al., 2019). Each dialogue pair is annotated by three Mturk workers. The final results are shown in Tab. 4.

The method *DiaAdv* can be regarded as an extension of *DiaMultiDense* by adding a classifier to provide a stronger training signal. According to the results from Section 6.1, these two methods do improve the success rate of dialogue agents. However, as shown in Tab. 4, while the success rate improves, the user experience degrades. According to Tab. 1, *GDPL* and *DiaAdv* have similar *F1* scores but the *DiaAdv* has a higher *Recall* value; this means that *DiaAdv* achieves a lower *Precision*. The unnecessary information mixed in the system response annoys users and results in a lower user experience. Given the relatively large difference in terms of success rate, the trade-off between success rate and user experience should be carefully examined. From another perspective, it is understandable that *GDPL* can provide a better user experience because a pre-designed user simulator is involved and the discriminator will encounter more diverse state-action combinations that are not seen in the training data. In contrast, the discriminator in *DiaAdv* only has access to the training data and this limits its judging ability. This does not imply that having a user simulator in the loop is essential to provide high quality user experience: *DiaSeq*, which is a completely supervised learning method, outperforms *GDPL*.

6.3 Discussion

How many expert dialogues are enough to train a dialogue agent with supervised learning? One motivation for dropping supervised learning and employing RL methods in TDS is that building high-quality conversational datasets is expensive and time-consuming. In contrast, training dialogue agents with a user-simulator is cheaper and affordable in many cases. Since we have no control on how much domain knowledge should be involved to build a user-simulator, we are not able to measure the expense of a reliable user-simulator. However, we can conduct an experiment to show how many real human dialogues are required to train a high-quality dialogue agent.

Based on the original MultiWoZ dataset, we build three smaller subsets: *MultiWoZ(0.1)*, *MultiWoZ(0.4)*, *MultiWoZ(0.7)* by only keeping 10%, 40%, and 70% dialogue pairs from the original dataset, respectively. We retrain *DiaMultiClass*, *GDPL*, *DiaAdv*, *DiaMultiDense*, *DiaSeq* and report the performance in Tab. 3. With respect to supervised learning agents, with only 10% expert dialogue pairs, *DiaMultiClass* gets half the success rate compared to the original performance (Tab. 1). By adding 30% more dialogue pairs to the training set, *DiaMultiClass* can achieve the same performance 59% with the original success rate 57.2%. Beyond this, *DiaMultiClass* does not benefit from the increase in expert dialogues and starts to fluctuate between 55% and 59%. In contrast, *DiaSeq* can achieve higher performance when there are only 10% expert dialogue pairs and the success rate increases with the number of available expert dialogues. *DiaMultiDense* achieves the best performance with the same amount of expert dialogues as the other two supervised learning methods. The performance difference among the three supervised learning methods shows that the method itself is the main factor to influence the performance rather than the number of available expert dialogues in the given dialogue environment. Traditional *DiaMultiClass* does not exert the potential of a given dataset to the fullest in dialogue PL.

Can adversarial learning eliminate expert dialogues? As can be concluded from Tab. 3, *GDPL* and *DiaAdv* managed to improve the performance with the increasing number of expert dialogues. *GDPL* and *DiaAdv* have the reward models that are supposed to distinguish real dialogue pairs from the machine-generated ones. By observing more expert dialogues, the reward model can provide a dialogue policy with more reliable and consistent updating signals. Fig. 3 shows the success rate gain by applying adversarial learning methods to the corresponding pre-trained models.[7] When the success rates of *DiaMultiClass* with MultiWoZ(0.4) and MultiWoZ(1.0) are both around 60%, deploying *GDPL* manages to bring 10% performance gain. The performance difference can be caused by the improved quality of the reward model. Conversely, if the reward model has no access to sufficient amount of expert behaviors, it has little clue how the expert dialogues should look like. This can lead to poor reward signals for the policy network. We can see it in the case of *GDPL* that the success rate drops to 21% while the pre-trained model can achieve 31% success rate on MultiWoZ(0.1). The performance gain between *DiaMultiDense* and *DiaAdv* is not so remarkable with respect to success rate compared to the gain between *DiaMultiClass* and *DiaAdv*.

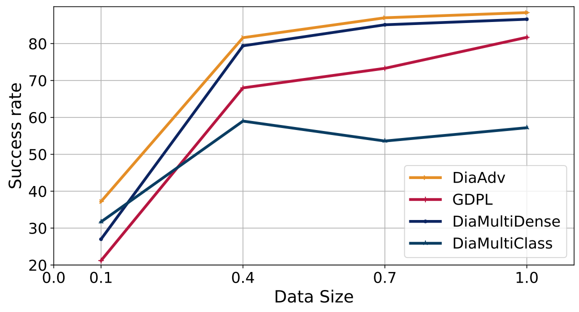

Figure 3: The performance gain between the pre-trained and their corresponding adversarial learning models with different amounts of expert dialogues.

[7]*DiaAdv* is the adversarial extension of *DiaMultiDense* while *GDPL* is the adversarial extension of *DiaMultiClass*.

However, *DiaAdv* does help to reduce the dialogue turns while improving the success rate as shown in Tab. 3. We can regard *DiaAdv* as a promising method to fine-tune the *DiaMultiDense* to explore more potential dialogue states.

How sensitive are adversarial learning to pre-trained dialogue policy? We explore how pre-trained dialogue policies affect the final performance of adversarial learning based dialogue agents. We first use supervised learning to pre-train the dialogue policies of *GDPL* and *DiaAdv* respectively with different training epochs. As shown in Fig. 4, the performance gain between the pre-trained dialogue policy and the corresponding adversarial are limited. With respect to *GDPL*, it even degenerates the original performance of the pre-trained policy when the starting points are relatively low. In other words, the main contributions to the adversarial dialogue agents come from the supervised learning stage; it is challenging for the dialogue agents to achieve the same performance without a promising pre-trained dialogue policy.

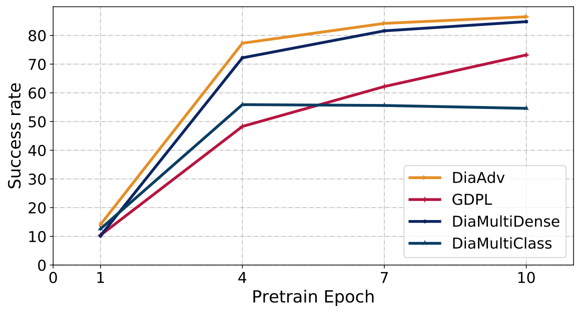

Figure 4: The performance gain between the pre-trained and their corresponding adversarial learning models with different amounts of pre-taining epochs.

7 Conclusion

We proposed two supervised learning approaches and one adversarial learning method to train the dialogue policy for TDSs without building user simulators. The proposed methods can achieve state-of-the-art performance suggested by existing approaches based on Reinforcement Learning (RL) and adversarial learning. However, we have demonstrated that our methods require fewer training efforts, namely the domain knowledge needed to design a user simulator and the intractable parameter tuning for RL or adversarial learning. Our findings have questioned if the full potential of supervised learning for dialogue Policy Learning (PL) has been exerted and if RL methods have been used in the appropriate TDS scenarios.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

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.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

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*.

M Gašić, N Mrkšić, Pei-hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Policy committee for adaptation in multi-domain spoken dialogue systems. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 806–812. IEEE.

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.

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*.

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*.

Ziming Li, Sungjin Lee, Baolin Peng, Jinchao Li, Julia Kiseleva, Maarten de Rijke, Shahin Shayandeh, and Jianfeng Gao. 2020. Guided dialogue policy learning without adversarial learning in the loop. In *EMNLP 2020*. ACL.

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.

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*.

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.

Marilyn A Walker, Diane J Litman, Candace A Kamm, and Alicia Abella. 1997. Paradise: A framework for evaluating spoken dialogue agents. *arXiv preprint cmp-lg/9704004*.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. 2016. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*.

Joseph Weizenbaum. 1966. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

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, Matthew Henderson, Antoine Raux, Blaise Thomson, Alan Black, and Deepak Ramachandran. 2014. The dialog state tracking challenge series. *AI Magazine*, 35(4):121–124.

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