Reward Estimation with Scheduled Knowledge Distillation for Dialogue Policy Learning

Junyan Qiu[a], Haidong Zhang [b] and Yiping Yang[c]

[a]University of Chinese Academy of Sciences, Zhongguancun Road, Haidian District, Beijing, China; [b,c]Institute of Automation, Chinese Academy of Sciences, Zhongguancun Road, Haidian District, Beijing, China

ARTICLE HISTORY
Compiled January 8, 2023

ABSTRACT
Formulating dialogue policy as a reinforcement learning (RL) task enables a dialogue system to act optimally by interacting with humans. However, typical RL-based methods normally suffer from challenges such as sparse and delayed reward problems. Besides, with user goal unavailable in real scenarios, the reward estimator is unable to generate reward reflecting action validity and task completion. Those issues may slow down and degrade the policy learning significantly. In this paper, we present a novel scheduled knowledge distillation framework for dialogue policy learning, which trains a compact student reward estimator by distilling the prior knowledge of user goals from a large teacher model. To further improve the stability of dialogue policy learning, we propose to leverage self-paced learning to arrange meaningful training order for the student reward estimator. Comprehensive experiments on Microsoft Dialogue Challenge and MultiWOZ datasets indicate that our approach significantly accelerates the learning speed, and the task-completion success rate can be improved from 0.47%∼9.01% compared with several strong baselines.

KEYWORDS
Reinforcement learning; dialogue policy learning; curriculum learning; knowledge distillation

1. Introduction

Task-oriented dialogue (TOD) systems are designed to assist users to accomplish specific goals (e.g., booking flight tickets or querying the weather) with a natural interaction. It has been widely used to build commercial virtual voice assistants, such as Apple Siri, Microsoft Cortana and Amazon Echo. As one of the key components of TOD, dialogue policy determines the next action that agents should take under the dialogue context at each turn (Zhang, Wang, Zheng, Zhao, & Huang, 2021; Zhao, Wang, Zhu, et al., 2021).

Traditional researches on dialogue policy learning (DPL) mainly focus on non-statistical methods with expensive expert handcrafted rule-making, which is non-extensible and highly complex in nature (Chen, Liu, Yin, & Tang, 2017; Zhao, Wang, Zhu, et al., 2021). Applying machine learning approaches, especially supervised learn-

CONTACT Junyan Qiu. Email: qiujunyan2018@ia.ac.cn

Table 1. An dialogue example with different user goals and the evaluation results. The user goal changes from Italian restaurant to Thai one at turn 6. *agt* stands for agent while *usr* stands for user.

User goal 1: Find a restaurant, ask its phone number. User goal 2: Find a Chinese restaurant.
agt_1: What can I help you? usr_2: I want to find a place to eat. agt_3: What about La Pergola, it's a nice Italian restaurant. usr_4: OK, what's the phone-number? agt_5: 1456-2311. usr_6: Sorry, I change my mind, is there a restaurant that serves Thai food? agt_7: Sure, Above eleven, its number is 1334-2587. usr_8: Thank you. agt_9: Thank you.
For user goal 1: Succeed, all requirements satisfied. For user goal 2: Failed, not a Chinese restaurant.

ing (SL), enables dialogue systems to optimize policy statistically (Hosseini-Asl, McCann, Wu, Yavuz, & Socher, 2020; X. Li, Chen, Li, Gao, & Celikyilmaz, 2017; Z. Li, Kiseleva, & de Rijke, 2020). However, the fixed training corpus does not necessarily present all optimal strategies, thus the system can be biased by the annotations (Jeon & Lee, 2022). Recently, DPL is formulated as a reinforcement learning (RL) problem, where the agent learns dialogue policy by interacting with users so as to maximize expected cumulative discounted reward (Diakoloukas, Lygerakis, Lagoudakis, & Kotti, 2020; C. Tian, Yin, & Moens, 2022; Zhang et al., 2021; Zhao, Wang, Zhu, et al., 2021; Zhou, Zhu, & Zhao, 2022). Thus, the performance of dialogue system depends much on the designing of reward functions.

Typical reward functions merely provide rewards at the end of conversation depending on whether the task is successfully accomplished. Such a scheme does not reflect the validity of system action at intermediate turns, which may lead to insufficient exploration in large action space and slow down DPL considerably. Some researchers proposed to get intrinsic reward after each action and learn to optimize its action choices in time (Peng, Li, Gao, Liu, Chen, & Wong, 2018; S.-Y. Su, Li, Gao, Liu, & Chen, 2018; Zhang et al., 2021). For example, (Peng, Li, Gao, Liu, Chen, & Wong, 2018; S.-Y. Su et al., 2018) employed a discriminator to distinguish responses from agents and users, and used its output as the intrinsic reward. (P.-H. Su, Gašić, & Young, 2018; Zhang et al., 2021; Zhao et al., 2020) divided predefined user goals into multiple subgoals and generate intrinsic reward by estimating subgoals completion. However, most of them ignored the fact that user goals have a crucial impact on evaluating task success, and they can not deal with shifty user goals, as exemplified by dialogue example in Table 1

Knowledge distillation can transfer privileged information encoded in a complex model (i.e., the teacher) to a compact one (i.e., the student) (Gou, Yu, Maybank, & Tao, 2021; Hinton, Vinyals, Dean, et al., 2015). Considering that user goals are unavailable in real-life conversations, we regard them as auxiliary information and incorporate them into a teacher reward estimator at the training stage. The teacher is then used to guide the training of a compact student reward estimator. Instead of

generating rewards based on predefined user goals, the proposed method can dynamically adjust its strategy of rewarding by learning prior distributions of user goals. As a result, it requires fewer human interactions to make the policy satisfactory and can deal with shifty user goals more intelligently.

The standard training approach based on randomly sample shuffling, might suffer from local optima, data noise and unstable problems (X. Wang, Chen, & Zhu, 2021; Zhao, Wang, & Huang, 2021). Curriculum learning, which trains models from easy samples to hard ones, can improve the generality and convergence rate of models in dialogue policy optimization (S. Liu, Zhang, He, Xu, & Zhou, 2021; Zhao, Qin, Zhenyu, Zhu, & Wang, 2022; Zhao, Wang, & Huang, 2021). Hence, we incorporate it into the knowledge distillation process, and design a complexity measurer and training scheduler to arrange reasonable training order for the student reward estimator.

To sum up, we propose a scheduled knowledge distillation framework (SKD), which distills user goals to automatically generate proper reward for efficient DPL. It is mainly composed of two processes: knowledge distillation transferring the internal knowledge of user goals from the teacher reward estimator to the student, and curriculum learning scheduling the order of sampled data throughout the training process of the student. Specifically, we leverage BERT (Devlin, Chang, Lee, & Toutanova, 2018) to construct a teacher reward estimator, which is accessible to user goals, and pre-train it via deep Q-learning (Mnih et al., 2015). Besides, we also devise an annotator to generate turn-level supervised reward label based on sub-goals completion for the teacher and train it by weakly supervised learning. Then we train the student reward estimator to mimic outputs of the teacher. Since the teacher is trained with access to user goals, we assume that the student can anticipate user goals while assigning rewards. The *self-paced learning*, a typical kind of curriculum learning, arranges scheduled learning process for the student reward estimator. The basic idea is to gradually increase the difficulty of knowledge transferred to the student reward estimator. It mainly consists of two components: 1) Complexity measurer that evaluates the difficulty of a conversation by distinguishing outputs of the teacher and student reward estimator. The conversation is deemed complicated if it indicates large divergence between them, and vice versa. 2) Training scheduler controlling the knowledge distillation process based on the conversation complexity given by the complexity measurer.

Extensive experiments demonstrate that our method significantly improves the efficiency and effectiveness of dialogue policy learning by combining knowledge distillation and curriculum learning. Moreover, our method can be easily applied to value-based reinforcement learning algorithms, e.g., deep Q-network (DQN) and deep dyna-Q (DDQ) Peng, Li, Gao, Liu, and Wong (2018). To conclude, our contributions can be summarized as follows:

- We propose a novel scheduled knowledge distillation (SKD) framework that incorporates user goals information to automatically generate immediate rewards and learn dialogue policy efficiently. To the best of our knowledge, this is the first work to leverage knowledge distillation of user goals for turn-level reward prediction.
- We introduce a new curriculum learning method that combines knowledge distillation to automatically schedule the learning process of the student reward estimator. As far as we know, we are the first to apply curriculum learning to learn a reward estimator for efficient DPL.
- We conduct comprehensive experiments by constructing a dialogue agent for the movie-ticket booking task. The results indicate that our method accelerates the

learning speed of dialogue policy and improves performance significantly.

The rest of the paper is organized as follows. In Section 2, we introduce existing works that are mostly related to this paper. In Section 3, we briefly present the background of DQN and DDQ algorithms. In Section 4, we depict the detailed architecture of our proposed method. The performance evaluation is given in Section 5, and the conclusion and future work are presented in Section 6.

2. Related work

In this section, we will introduce existing works that are closely related to this paper, including RL-based dialogue policy in 2.1, knowledge distillation in Section 2.2 and curriculum learning in Section 2.3. The related works are summarized in Table 2.

2.1. RL-based dialogue policy

Reinforcement learning (RL) has been widely used to optimize dialogue policy due to its ability to automatically fine-tune dialogue strategy based on user feedback (Geishauser et al., 2022; G. Wu et al., 2021; Zhao et al., 2022; Zhao, Wang, & Huang, 2021). These methods regard real users as the environment and the agent learns its policy by interacting with them. How to improve the efficiency of DPL to reduce human-machine interactions remains to be the biggest challenge.

A feasible solution is to employ user simulators as substitution of real users. Technically, the simulator can generate infinite dialogue experience without any real-world cost. Unfortunately, the quality of dialogue policy can not be guaranteed due to the discrepancy between real and simulated users (Peng, Li, Gao, Liu, & Wong, 2018). Thus, (Peng, Li, Gao, Liu, & Wong, 2018) proposed a Deep Dyna-Q (DDQ) framework that introduced an trainable environment call *world model* to mimic real user behaviors. Based on DDQ, some researchers attempted to improve the quality of simulated experience by using GANs (Creswell et al., 2018) or employing Gaussian process based model (G. Wu et al., 2021). Despite the outstanding achievements of these methods, the reward function that governs the learning process is normally designed as providing reward signals at the end of a conversation, which suffers from the *reward sparsity* issue and decreases the efficiency of DPL significantly.

Another line of research proposed to improve the training efficiency by constructing dense reward functions (Zhang et al., 2021; Zhao et al., 2020). For example, (Peng, Li, Gao, Liu, Chen, & Wong, 2018; S.-Y. Su et al., 2018) applied adversarial training by employing a discriminator to distinguish responses generated by agents from those by users, and using its output as the intrinsic reward (Peng, Li, Gao, Liu, Chen, & Wong, 2018). However, the notorious instability of adversarial training suffers badly from non-convergence and highly sensitive to the hyperparameters selection (G. Wu et al., 2021). (Khandelwal, 2021; Peng, Li, Gao, Liu, Chen, & Wong, 2018) leveraged human dialogues or domain text as prior knowledge to pre-train the reward estimator, which requires extra high-quality annotated data to pre-train the reward functions.

Furthermore, existing approaches have proved that randomly sampling user goals for dialogue agent to train on may lead to the learned policy inefficient and unstable (Zhao, Wang, & Huang, 2021). To address that issue, many researchers proposed to incorporate curriculum learning into DPL. For example, (Zhao, Wang, & Huang, 2021; H. Zhu, Zhao, & Qin, 2021) replaced random user goal sampling with a teacher

model to arrange meaningful training order for the policy. Similarly, (S. Liu et al., 2021) evaluated user goal complexity based on the dialogue state differential space that the agent can explore for current training samples. (Zhao et al., 2022) leveraged the learning experiences of dialogue policy and skip-level selection according to the learning needs to maximize the efficiency.

However, all these methods ignored the crucial impact of user goals on evaluating task success and could not deal with shifty user goals. In this paper, we propose a novel framework that incorporates user goal to generate dynamic and accurate reward for efficient DPL. Our method has three attractive features that combines the advantages of the aforementioned three approaches: (1) It can be easily applied to value-based approaches, including DQN and DDQ; (2) It provides accurate reward after each action to explore the action space more efficiently; (3) It arranges meaningful learning schedule for the reward estimator and policy that improves stability of dialogue policy learning.

2.2. Knowledge distillation

Knowledge distillation (KD) was first proposed by (Hinton et al., 2015) to compress the knowledge in a cumbersome model into a small one. It is generally composed of a teacher model, a student model and a distillation approach: the teacher model holds rich knowledge from annotated data or prescient extra inputs, while the student model learns to match with it. The distillation approach aims to transfer the knowledge from teacher to the student. It is widely used in computer vision (Greco, Saggese, Vento, & Vigilante, 2021; Ren et al., 2021; X. Wang et al., 2020), natural language processing (Haidar, Rezagholizadeh, et al., 2019; B. Li et al., 2021; J. Liu, Chen, & Liu, 2019) and other applications (Ren et al., 2021; Shen, Xu, & Cao, 2020).

Moreover, KD can be also used to enhance the student model when the teacher is equipped with privileged information (Z. Tian et al., 2020; W. Wu et al., 2019). As stated before, the user goal is a crucial factor to build a reliable reward estimator, while it is unavailable when communicating with real users. To this end, we construct a teacher-student framework to transfer goal-anticipated information from the teacher reward estimator, employing large-scaled language model and pre-trained with user goals, to the student. This framework enables the reward estimator to keep user goals in mind when generating rewards during inference.

2.3. Curriculum learning

Curriculum learning (Bengio, Louradour, Collobert, & Weston, 2009) (CL) is a strategy that trains a machine learning model by learning from easy samples and gradually increasing the complexity. It normally consists of two components, i.e., a difficulty measurer (DM) that evaluates the complexity of each data and a training scheduler (TS), which selects data with desired complexity degree for the model to train on. Generally, all training data is sorted by the DM and passed to the TS, which samples relatively simple examples and sends them to the model for training. The TS also decides when to sample harder examples as the training progresses until the whole dataset is included for training. A plenty of researches have exploited its power in a wide range of application scopes, such as supervised learning tasks (El-Bouri, Eyre, Watkinson, Zhu, & Clifton, 2020; Guo et al., 2018; Platanios, Stretcu, Neubig, Póczos, & Mitchell, 2019), reinforcement learning tasks (Narvekar & Stone, 2019; Qu, Tang,

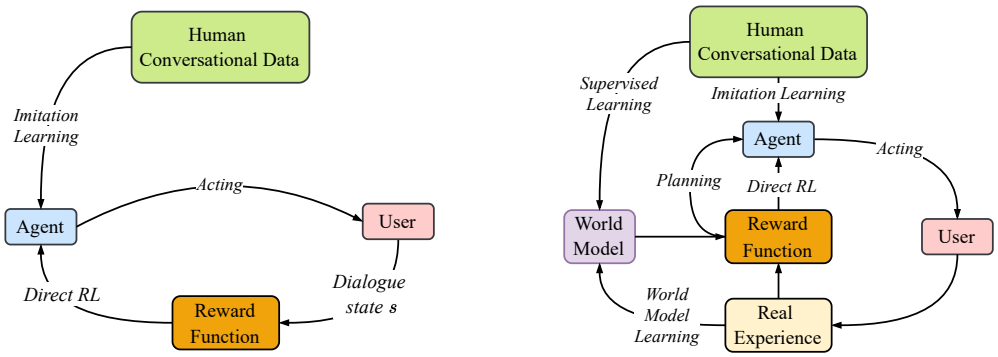
& Han, 2018) and graph learning tasks (Chu, Wang, Shi, & Jiang, 2021; Dong, Long, Xu, & Xiao, 2021; Y. Wang, Wang, Liang, Cai, & Hooi, 2021). Recent works employed curriculum learning to arrange reasonable schedule for efficient DPL (S. Liu et al., 2021; Zhao, Wang, & Huang, 2021; H. Zhu et al., 2021). In this paper, we incorporate a special kind of curriculum learning, self-paced learning, into distillation process to train the student reward estimator more stable and efficiently.

Table 2. Summarization of related work.

RL-based dialogue policy			
Categories	Drawbacks		References
DDQ-based framwork	Reward sparsity	Ignore user goals	(Creswell et al., 2018; Peng, Li, Gao, Liu, & Wong, 2018) (G. Wu et al., 2021)
Reward shaping	Requiring extra resources Unstable	Uncapable of dealing with shifty user goals	(Zhang et al., 2021; Zhao et al., 2020) (Peng, Li, Gao, Liu, Chen, & Wong, 2018; S.-Y. Su et al., 2018) (Khandelwal, 2021)
Curriculum learning	Reward sparsity		(Zhao, Wang, & Huang, 2021; H. Zhu et al., 2021) (S. Liu et al., 2021; Zhao et al., 2022)
Knowledge distillation			
Applications	References		
Natural Language processing	(Greco et al., 2021; Ren et al., 2021; X. Wang et al., 2020)		
Computer vision	(Haidar et al., 2019; B. Li et al., 2021; J. Liu et al., 2019)		
Other applications	(Ren et al., 2021; Shen et al., 2020)		
Curriculum learning			
Application scopes	References		
Supervised learning	(El-Bouri et al., 2020; Guo et al., 2018; Platanios et al., 2019)		
Reinforcement learning	(Narvekar & Stone, 2019; Qu et al., 2018)		
Graph learning	(Chu et al., 2021; Dong et al., 2021; Y. Wang et al., 2021)		

3. Background

Before we describe our method in detail, we briefly introduce the background of two RL-based dialogue policy learning strategies, i.e., deep Q-network in Section 3.1 and deep dyna-Q in Section 3.2.



(a) DQN (b) DDQ

Figure 1. Different strategies of learning dialogue policy.

3.1. Deep Q-network

Dialogue policy determines which action the agent should take at next turn and is normally formulated as a Markov Decision Process. It is normally optimized by applying reinforcement learning algorithms, consisting of an agent, a user and a reward function, as shown in Fig. 1a. The agent is defined as a deep Q-network (DQN)

(Mnih et al., 2015) $Q(s_t, a_t; \theta_Q)$ parameterized by θ_Q. It is initialized on human conversational data via imitation learning, and optimized by interacting with the user, which is referred to as *direct reinforcement learning*. Particularly, at each turn t, The agent observes the dialogue state s_t, selects an action a_t from the action set $\mathcal{A}$ in a $\epsilon - greedy$ way, i.e., randomly selecting an action with a small probability ϵ or according to $a_t = \arg\max_{a \in \mathcal{A}} Q(s_t, a; \theta_Q)$. The the reward function assigns a reward r_t for the agent and the user updates the dialogue states to s_{t+1}. The generated experience (s_t, a_t, s_{t+1}, r_t) is stored to a replay buffer called the real experience pool ($\mathcal{D}^u$) during direct reinforcement learning. The objective for DQN optimization is defined as:

$$\mathcal{L} = \mathbb{E}_{s_t, a_t, s_{t+1}, r_t}[(Q(s_t, a_t; \theta_Q) - y)^2] \tag{1}$$

$$y = r_t + \gamma \max_{a_{t+1} \in \mathcal{A}} Q'(s_{t+1}, a_{t+1}; \theta_{Q'}) \tag{2}$$

where $\gamma \in [0, 1]$ is a discount factor, $Q'(s, a; \theta_{Q'})$ is the target work that is updated with the Q-network parameters θ_Q periodically and fixed between individual iterations Mnih et al. (2015).

3.2. Deep Dyna-Q

Despite the effectiveness of learning dialogue policy via DQN, it is requisite to interact with real users for policy optimization, which is expensive to train dialogue agents from scratch. Previous researches proposed to build simulators as substitutions of real users X. Li et al. (2016), which can generate unlimited simulated experiences and does not cost any human forces. However, user simulators have difficulties in modeling the conversational complexity of human interlocutors (Peng, Li, Gao, Liu, & Wong, 2018). (Dhingra et al., 2017) pointed out that dialogue agents trained with user simulators exhibit significant discrepancy compared with those trained with real users. To tackle these problems, deep Dyna-Q (DDQ) (Sutton, 1990), a typical RL method, has been introduced to dialogue policy learning (Peng, Li, Gao, Liu, & Wong, 2018; S.-Y. Su et al., 2018; Zhao et al., 2020) to bridge the gap between real users and user simulators. It employed a trainable environment called *world model* to mimic real users. As shown in Fig. 1b, the world model is implemented by a neural network $W(s_t, a_t; \theta_W)$ parameterized by θ_W. It is initialized on human conversational data via supervised learning, and trained through multi-task learning, referred to as the *world model learning*. The dialogue policy is optimized by interacting with simulated users, referred to as *planning*, in most cases, and requires only a small number of real user interactions. The simulated experience (s_t, a_t, s_{t+1}, r_t) generated during planning is stored to another replay buffer called simulated experience pool $\mathcal{D}^s$.

4. Method

In this section, we will introduce the detailed architecture of our proposed method in three subsections, including knowledge distillation in Section 4.1, self-paced learning in Section 4.2 and dialogue policy learning in Section 4.3.

Fig. 2 illustrates the proposed Scheduled Knowledge Distillation (SKD) for Dialogue Policy Learning, which can be divided into two processes: 1) **Knowledge distillation.** In this process, we designed two reward estimators. One of them serves as the teacher, which has access to user goals and is pre-trained via deep Q-learning and

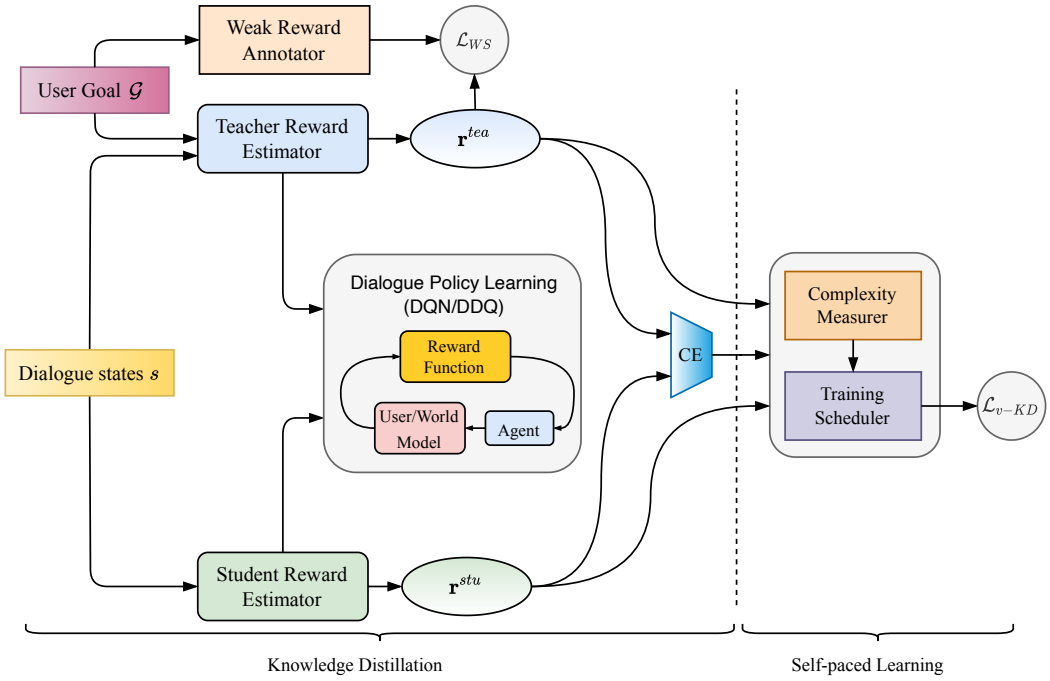


Figure 2. Architecture of the proposed scheduled knowledge distillation for dialogue policy learning.

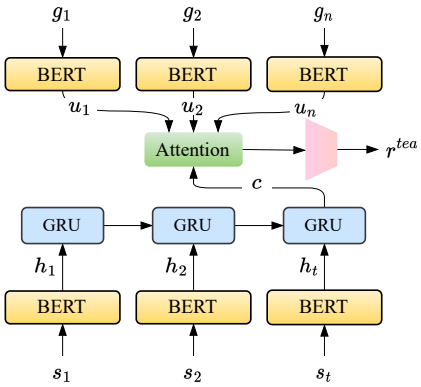
weakly supervised learning. The other one serving as the student is trained to mimic outputs of the teacher. The dialogue policy is optimized and parameters of teacher reward estimator are frozen while training the student model. 2) **Self-paced learning**. We design two modules to arrange scheduled distillation for the student reward estimator, i.e., complexity measurer and training scheduler. The former one evaluates the difficulty degree of each conversation by calculating the divergence of outputs between the teacher and student reward estimator. The latter one decides whether the current example is too hard for the student to learn at current stage based on the evaluation of the complexity measurer.

4.1. Knowledge distillation

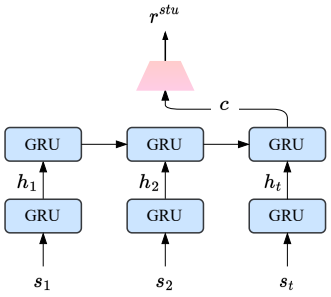
Knowledge distillation is applied to transfer user goal anticipated information from the teacher reward estimator to the student model. In this part, we first present the architecture of the teacher and student reward estimators in subsection 4.1.1. Then we introduce the pre-training procedure of the teacher model in subsection 4.1.2. To the end, we describe the process of transferring knowledge to the student reward estimator in subsection 4.1.3

4.1.1. Architecture of teacher and student reward estimators

4.1.1.1. Teacher reward estimator. Note that we expect to generate rewards for each action based on historical dialogue states and user goals. Unfortunately, user goals, which plays a crucial role in reward estimation, are normally unavailable in real-life conversations. Thus, we design a teacher reward estimator to learn prior distribution of user goals.

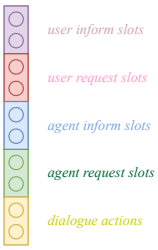


(a) Teacher reward estimator

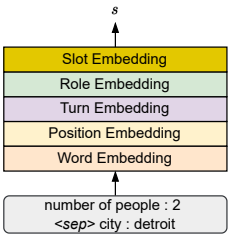


(b) Student reward estimator

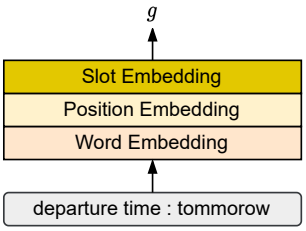
Figure 3. Architecture of teacher and student reward estimators.



(a) Multi-hot state representations



(b) Embedded state representations



(c) Embedded user goal representations

Figure 4. Dialogue state and user goal representations

The teacher reward estimator is presented in Fig. 3a. It takes as input the dialogue states and user goals, and outputs reward for the agent. Traditional methods represent dialogue states with multi-hot vectors, which are concatenated by several feature categories, including dialogue actions, inform and request slots from users and agents respectively Peng, Li, Gao, Liu, and Wong (2018); S.-Y. Su et al. (2018), as shown in Fig. 4a. However, such mechanism viewing each dialogue state as individual unit can not fully explore the semantic meaning shared across states and actions, e.g., both *city* and *theater* convey information regarding locations, which should elicit similar representations. To this end, we propose to map these states into dense embeddings to capture semantic information contained in each token. As illustrated in Fig. 4b, we designed five types of embeddings to represent the dialogue states including: (1) word embedding Φ_w converting each token in the states to dense vectors; (2) position embedding Φ_p indicating sequence order; (3) turn embedding Φ_t indicating turn order; (4) role embedding Φ_r identifying the interlocutor, i.e., user and agent; (5) slot embedding Φ_s describing types of slots, e.g., *inform* and *request*. The state representation s_t is obtained by adding these embeddings. Notably, if there exist more than one slots at one turn, they will be concatenated as a single sequence with a special token $\langle$sep$\rangle$ to separate them.

The user goal $\mathcal{G}$ is the objective of the conversation, which is composed of a set of request slots $\mathcal{R}$ and constraint (inform) slots $\mathcal{C}$ Lu, Zhang, and Chen (2019). Request slots specify the desired information, such as the location and time in movie-tickets booking task. While constraint slots pinpoint the requirements, e.g, a cheap restaurant

that servers Chinese food. For example, when a user wants to *"book two flight tickets from Shanghai to Beijing tomorrow"*, the user goal is formulated as:

$$\mathcal{G} = \left(\begin{array}{l} \mathcal{C} = \left[\begin{array}{l} NumberOfTickets : two \\ DepartureDate : tomorrow \\ DeparturePlace : Shanghai \\ Destination : Beijing \end{array} \right] \\ \mathcal{R} = [DepartureTime] \end{array} \right) \tag{3}$$

The user goal can be divided into several subgoals, where each subgoal g contains exactly one slot. For example, user goal Equation 3 can be divided into five subgoals and two of them can be formulated as:

$$g_1 = \left(\ \mathcal{C} = [NumberOfTickets : two] \ \right) \tag{4}$$

and

$$g_2 = \left(\ \mathcal{R} = [DepartureTime] \ \right) \tag{5}$$

Similar to the dialogue state, each subgoal g is also mapped into dense embedding. It is calculated by adding three types of embeddings, i.e., word embedding, position embedding and slot embedding, the brief illustration is depicted in Fig. 4c.

After obtaining dialogue state and user goal representations, we design a teacher reward estimator to generate reward for each action. Concretely, we employ a large scale pre-trained language model BERT (Devlin et al., 2018) to encode the historical dialogue states $\{\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_t\}$. It first leverages BERT to convert the dialogue state to hidden vectors. In order to get the fixed-size representation, we take the output of BERT corresponding to the first token $< cls >$, which is a special token added before each sequence following (Devlin et al., 2018), $\mathbf{h}_i = \text{BERT}(\mathbf{s}_i)$. Then, these vectors are fed into Gated Recurrent Units (GRU) (Chung, Gulcehre, Cho, & Bengio, 2014) to capture semantic dependencies across turns, $\mathbf{c} = \text{GRU}([\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_t])$.

Similarly, the embedded sub-goals are separately converted into fixed-size vectors by BERT, $\mathbf{u}_j = \text{BERT}(\mathbf{g}_j)$. The dialogue state and user goals are then aggregated by computing attention distributions between them.

$$\alpha_i = \frac{\exp(\mathbf{c} \cdot \mathbf{u}_i)}{\sum_{j=1 \sim t} \exp(\mathbf{c} \cdot \mathbf{u}_j)}$$

$$\mathbf{e} = \sum_{i=1}^{t} \alpha_i \cdot \mathbf{u}_i$$

Finally, the aggregation representation is fed into a feedforward neural network to generate reward for each action:

$$\mathbf{p}^{tea} = sigmoid(\mathbf{e} \cdot \mathbf{W} + \mathbf{b}) \tag{6}$$

$$\mathbf{r}^{tea} = R_a \cdot \mathbf{p}^{tea} \tag{7}$$

where $\mathbf{W} \in \mathbb{R}^{d_m \times |\mathcal{A}|}$ and $\mathbf{b} \in \mathbb{R}^{|\mathcal{A}|}$ are trainable parameters (d_m represents the number of hidden nodes), R_a is a hyper-parameter referred to as the *anchor reward*. $\mathbf{p}^{tea}$ and $\mathbf{r}^{tea}$ are $|\mathcal{A}|$-dimensional vectors, where each dimension i represents the probability of the i_{th} action deemed effective and the reward of taking the i_{th} action respectively.

4.1.1.2. Student reward estimator. The architecture of student reward estimator is depicted in Fig. 3b. It differs from the teacher model in two aspects: (1) It utilizes GRU instead of BERT to encode the dialogue states to compress the model size and reduce the cost of computing resources during inference; (2) It generates rewards merely depending on the historical dialogue states without access to user goals, which conforms to the real-world situations.

4.1.2. Pre-training of teacher reward estimator

The teacher reward estimator is pre-trained on two parts. The first part is deep Q-learning, which is stated in section 3.1 and defined as follows:

$$\mathcal{L}_Q^{tea} = \mathbb{E}_{s_t, a_t, s_{t+1}, r_t}[(Q(s_t, a_t; \theta_Q) - y)^2] \tag{8}$$

$$y = r + r_i^{tea} + \gamma \max_{a_{t+1} \in \mathcal{A}} Q'(s_{t+1}, a_{t+1}; \theta'_Q) \tag{9}$$

where r_i^{tea} indicates the reward of taking the i_{th} action in the action set $\mathcal{A}$ generated by the teacher reward estimator[1], a_t is the action selected from action set $\mathcal{A}$ at turn t, which is converted to action representations through an action embedding $\Phi_a \in \mathbb{R}^{|\mathcal{A}| \times d_m}$. Furthermore, we also introduce weakly supervised learning to train the teacher model more effectively with supervised guidance. To this end, we devise a *weak reward annotator* in a non-parametric way by checking whether the dialogue action fulfills one of the sub-goals. We assign a reward for each of them in the action set $\mathcal{A}$ which constitutes the reward label $\mathbf{r} = \{r_1, r_2, \cdots, r_{|\mathcal{A}|}\}$, where r_i denotes the reward label of taking the i_{th} action a_i. Its value depends on whether the action can successfully complete user sub-goals, i.e., $r_i = \begin{cases} R_a & \text{if } a_i \text{ completes one user sub-goal} \\ 0 & \text{otherwise} \end{cases}$[2].

Particularly, one sub-goal is deemed to be satisfied under two circumstances: (1) constraint slots are requested or (2) the information of request slots are informed. For example, querying about the *number of tickets* and informing the *departure time* are identified as effective actions which fulfill sub-goals stated in Equation 4 and 5 respectively. The weakly supervised learning loss then capitalizes on L_2 norm to compute the distance between the reward label $\mathbf{r}$ and reward predicted by the teacher $\mathbf{r}^{tea}$.

$$\mathcal{L}_{WS} = \mathbb{E}(\|\mathbf{r} - \mathbf{r}^{tea}\|_2) = \frac{1}{|\mathcal{A}|} \sum_{i=1}^{|\mathcal{A}|} (r_i - r_i^{tea})^2 \tag{10}$$

The final loss is defined as the summation of the aforementioned two parts:

$$\mathcal{L}^{tea} = \mathcal{L}_Q^{tea} + \mathcal{L}_{WS} \tag{11}$$

[1] $r = 2T$ if the dialogue succeeds and $r = -T$ if it fails. T is the maximize number of dialogue turns. In this paper, $T = 40$. $r = -1$ if the dialogue is not finished yet.

[2] R_a is the same with that in Equation 7

The gradient with respect to the parameters of teacher reward estimator θ_T is

$$\nabla_{\theta_T}\mathcal{L}^{tea} = \mathbb{E}_{s_t,a_t,s_{t+1},r_t}[-2(Q(s_t,a_t;\theta_Q)-y)\cdot\nabla_{\theta_T}r_i^{tea}] + \frac{1}{|\mathcal{A}|}\sum_{i=1}^{|\mathcal{A}|}[-2(r_i - r_i^{tea})\cdot\nabla_{\theta_T}r_i^{tea}] \tag{12}$$

4.1.3. Transferring knowledge to the student reward estimator

Knowledge distillation was proposed to compress knowledge in a cumbersome model, i.e., the teacher, to a compact one, i.e., the student Hinton et al. (2015). In this paper, the knowledge is transferred by training the student network to mimic the soft label generated by the teacher reward estimator. Following Romero et al. (2014), we leverage cross entropy loss to measure the similarity between them. While training the student network, we freeze parameters of the pre-trained teacher.

$$\mathcal{L}_{KD} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{H}(\mathbf{p}_i^{tea}, \mathbf{p}_i^{stu}) \tag{13}$$

where $\mathcal{H}$ indicates the cross entropy loss, N is the number of examples in the training set, i is the index of the i_{th} example.

4.2. Self-paced learning

Self-paced learning, a typical kind of curriculum learning, is designed to control the order of the student learning from the teacher in the distillation process. It consists of two components, including complexity measurer and training scheduler.

4.2.1. Complexity measurer

In order to arrange scheduled learning process for the student reward estimator, we apply adversarial training to measure the conversation complexity. To achieve that, we employ a discriminator $\mathcal{D}$ to distinguish outputs of the teacher and student reward estimator. It is defined as the Kullback-Leibler (KL) loss between the rewards predicted by the teacher and student reward estimator, and generates a real value indicating the differentiation between them.

$$score_i = KL(\mathbf{p}_i^{tea} \| \mathbf{p}_i^{stu}) \tag{14}$$

If the discriminator outputs a small $score_i$, it suggests that the student reward estimator is able to mimic the teacher regarding the current example i, which can be deemed as a simple case and selected as a training instance in the early stage. Otherwise it is a hard one.

4.2.2. Training scheduler

As shown in Equation 13, the student learns from the teacher in all samples equally. To gradually increase the complexity of samples that the student model trained on, we introduce example weights $\mathbf{v} = \{v_1, v_2, \cdots, v_N\} \in \{0,1\}^N$ into the knowledge

distillation loss stated in Equation 13:

$$\mathcal{L}_{v-KD} = \frac{1}{N}\sum_{i=1}^{N} v_i \mathcal{H}(\mathbf{p}_i^{tea}, \mathbf{p}_i^{stu}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{|\mathcal{A}|} -v_i[p_{i,j}^{tea} \cdot \log(p_{i,j}^{stu})] \tag{15}$$

where N is the number of training samples. We use self-paced learning to estimate $\mathbf{v}$ as follows:

$$\min_{\mathbf{v}} \sum_{i=1}^{N} v_i score_i + g(v_i; \lambda) \tag{16}$$

where $g(v_i; \lambda)$ is a SP-regularizer defined as a negative ℓ_1 norm X. Wang et al. (2021):

$$g(v_i; \lambda) = -\lambda \cdot v_i \tag{17}$$

where λ is a hyperparameter controlling the learning pace. The optima value of $\mathbf{v}$ can be calculated by:

$$v_i^* = \begin{cases} 1 & score_i < \lambda \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

The solution is intuitive: if $score_i$ is smaller than the threshold λ, then it is a easy example and should be used for knowledge distillation ($v_i = 1$ in Equation 15) at the current step. Otherwise it is not selected as a training example ($v_i = 0$). To select examples with desired complexity, we gradually increase λ as the training processes. Concretely,

$$\lambda(t) = \lambda_0 + \frac{1 - \lambda_0}{T} \cdot t \tag{19}$$

where t is the current training step, T is the total training step, $\lambda_0 \in [0, 1]$ is a initial value to ensure the easiest examples are selected when the training starts.

4.3. Dialogue policy learning

The dialogue policy network $Q(s, a; \theta_Q)$ is trained by using experience stored in the real or simulated experience buffer[3], i.e., $\mathcal{D}^u$ and $\mathcal{D}^s$. To facilitate policy learning, we construct a new action-value function by combing the original $Q-$function and the student reward estimator $F(s, a; \theta_S)$, Mathematically:

$$\mathcal{L}_Q^{stu} = \mathbb{E}_{s_t, a_t, s_{t+1}, r_t}[(Q_s(s_t, a_t; \theta_Q, \theta_S) - y)^2] \tag{20}$$

$$y = r + \gamma \max_{a_{t+1} \in \mathcal{A}} Q_s'(s_{t+1}, a_{t+1}; \theta_Q', \theta_S') \tag{21}$$

where $Q_s(s_t, a_t; \theta_Q, \theta_S) = Q(s_t, a_t; \theta_Q) + F(s_t, a_t; \theta_S)$. During interaction, the action is decided by the combined action-value function, i.e., $a_t = \arg\max_{a_t \in \mathcal{A}} Q_s(s_t, a_t; \theta_Q, \theta_S)$.

[3]Simulated experience buffer is only used in DDQ based agents.

Table 3. The statistics of datasets.

Dataset		#Intents	#Slots	#Dialogues
MDC	Movie-Ticket Booking	11	29	2890
	Restaurant Reservation	11	30	4103
	Taxi Ordering	11	29	3094
MultiWOZ		13	30	10438

The policy network $Q(s, a; \theta_Q)$ can be optimized by back-propagation and mini-batch gradient descent with respect to θ_Q:

$$\nabla_{\theta_Q} \mathcal{L}_Q^{stu} = \mathbb{E}_{s_t, a_t, s_{t+1}, r_t}[2(Q_S(s_t, a_t; \theta_Q, \theta_S) - y) \cdot \nabla_{\theta_Q} Q(s_t, a_t; \theta_Q)] \tag{22}$$

Notably, the parameters of student reward estimator is updated together with the dialogue policy network. The final loss of training the student reward estimator and gradient is calculated by:

$$\mathcal{L}^{stu} = \mathcal{L}_{v-KD} + \mathcal{L}_Q^{stu} \tag{23}$$

$$\begin{aligned} \nabla_{\theta_S} \mathcal{L}^{stu} = & \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{|\mathcal{A}|} -v_i [p_{i,j}^{tea} \cdot \frac{1}{p_{i,j}^{stu}} \cdot \nabla_{\theta_S} p_{i,j}^{stu}] \\ & + \mathbb{E}_{s_t, a_t, s_{t+1}, r_t}[2(Q_s(s_t, a_t; \theta_Q, \theta_S) - y) \cdot \nabla_{\theta_S} F(s_t, a_t; \theta_S)] \end{aligned} \tag{24}$$

5. Experiment

In this section, we will introduce the datasets used in this paper in Section 5.1, baselines in Section 5.2, implementation details in Section 5.3, simulation evaluation in Section 5.4 and human evaluation in Section 5.5.

5.1. Datasets

We conduct experiments on two public dialogue datasets, i.e., Microsoft Dialogue Challenge (MDC) (X. Li et al., 2016, 2018) and MultiWOZ (Budzianowski et al., 2018). Microsoft Dialogue Challenge contains three tasks including *movie-ticket booking, restaurant reservation* and *taxi-ordering*. Each task has a built-in user simulator for experimentation purpose. The datasets are collected via Amazon Mechanical Turk (X. Li et al., 2016) and annotated using schema defined by domain experts. MultiWOZ is a task-oriented dialogue dataset with more than 10k dialogues involving 5 domains (*restaurant, hotel, attraction, taxi* and *train*). We use rule-based dialogue state tracker and user simulator provided by ConvLab-2 (Q. Zhu et al., 2020) for simulated interaction. Agenda-based user simulators (SCHATZMANN & YOUNG, 2009), providing stack-like representation of user states, are employed to reduce human intervention. The detailed statistics of three datasets are presented in Table 3.

5.2. Baselines

To validate the effectiveness of our proposed method, we carry out comprehensive experiments and compare with 4 DQN-based agents and 6 DDQ-based agents. Besides, we also add a **Rule-based** model constructed by manually annotated rules and various forms of logic[4], and evaluate its performance on MDC dataset.

5.2.1. DQN-based agents

- The **DQN** agent (Peng, Li, Gao, Liu, & Wong, 2018) is learned by standard DQN algorithm with only direct reinforcement learning in each epoch.
- **VACL** (Zhao et al., 2022) presents a versatile adaptive curriculum learning framework that applies automatic curriculum learning on dialogue policy tasks.
- The **ACL-DQN** agent (Zhao, Wang, & Huang, 2021) takes user goal sampled by a teacher model for automatic curriculum learning.
- SDPL (S. Liu et al., 2021) proposes a scheduled dialogue policy learning framework that models dialogue complexity by estimating the dialogue state differential space, and arranges reasonable learning schedule. **DQN-SDPL** is a DQN agent that incorporates SDPL framework.
- The **DQN-SKD** is the method proposed in this paper, which incorporates KD and CL into the DQN agent.

5.2.2. DDQ-based agents

- The **DDQ** agent (Peng, Li, Gao, Liu, & Wong, 2018) are learned by interacting with real users as well as a trainable environment called world model to improve data efficiency.
- **DR-D3Q** (Zhao et al., 2020) proposes dueling direct reinforcement learning and planning with dynamic reward based on the DDQ framework.
- **ES-DDQ** (Zhang et al., 2021) incorporates emotional features into DDQ agents.
- **D3Q** (S.-Y. Su et al., 2018) combines DDQ framework with generative adversarial network (GAN), which incorporates a discriminator into the *planning* process to control the quality of generated experience.
- **GP-DDQ** (G. Wu et al., 2021) designs a novel DDQ-based framework that builds the world model as a Gaussian Process model to generate high-quality simulated experiences.
- **DDQ-SDPL** (S. Liu et al., 2021) is a DDQ agent that incorporates SDPL framework.
- **DDQ-SKD** is the method proposed in this paper, which incorporates KD and CL into the DDQ agent.

5.3. Implementation details

For all the agents implemented in this paper, the policy networks $Q(s, a; \theta_Q)$ are two-layer MLPs with 80 hidden nodes and tanh activation functions following (Peng, Li, Gao, Liu, & Wong, 2018). The world model $W(s, a; \theta_W)$ in DDQ-based agents contains two shared hidden layers and three task-specific layers. We use base uncased version of BERT, with 12 layers, 768 hidden nodes and 12 attention heads, to encode dialogue states and user goals in the teacher model. The hidden nodes of GRU in the teacher

[4]Readers can refer to the E2Edialog (X. Li et al., 2018) repository for implementation details.

and student model are set to 768[5] and 80 respectively. $\epsilon-$greedy exploration is applied with $\epsilon = 0.05$ during training. The buffer size of real and simulated experience pool, i.e., $\mathcal{D}^u$ and $\mathcal{D}^s$ is set to 5000 following Peng, Li, Gao, Liu, and Wong (2018). The batch size is set to 16. The learning rates for optimizing the policy and reward estimator are set to $5e-4$ and $1e-5$ respectively. The maximum length of conversation L is set to 40. Besides, Replay Buffer Spiking (RBS) (Lipton et al., 2016), a variant of imitation learning, is employed to generate initial experiences. We apply rule-based agent implemented by (X. Li et al., 2018) and pre-fill $\mathcal{D}^u$ with 100 dialogues before training. The anchor reward R_a is set to 5, λ_0 in Equation 19 is set to 0.2.

[5]The reason for setting the hidden nodes of GRU in teacher model to 768 is to make it compatible with the outputs of BERT model.

Table 4. Experimental results on Microsoft Dialogue Challenge dataset. Best results are marked in **bold**.

Agent		Epoch=100			Epoch=200			Epoch=300		
		Success	Reward	Turns	Success	Reward	Turns	Success	Reward	Turns
Movie-Ticket Booking										
Rule-based		0.2135	-10.42	27.14	0.2367	-11.45	29.01	0.2256	-11.01	28.91
DQN-based	DQN	0.4676	-4.54	32.91	0.5586	10.98	25.91	0.6978	28.91	24.24
	VACL	0.4079	-4.79	29.49	0.4331	-1.42	28.81	0.4339	-1.33	28.79
	ACL-DQN	0.5717	15.92	27.36	0.7523	37.39	21.30	0.7573	45.28	18.57
	DQN-SKD	**0.6418**	**18.56**	**20.42**	**0.7716**	**39.81**	**20.14**	**0.8128**	**48.10**	**15.30**
DDQ-based	DDQ(10)	0.5918	23.14	27.12	0.6321	36.19	19.17	0.7316	40.19	18.45
	DR-D3Q	0.6183	15.94	21.42	0.6812	22.67	19.27	0.7899	33.75	16.68
	ES-DDQ	0.6243	22.74	22.78	0.7232	35.76	20.78	0.7581	40.10	18.55
	D3Q(10)	0.6333	28.99	16.01	0.7000	37.24	15.52	0.6667	33.09	15.83
	GP-DDQ	0.7069	35.09	21.48	0.7706	43.65	19.60	0.7874	45.72	19.54
	DDQ-SDPL	0.7600	**44.75**	14.90	0.8000	51.15	**13.29**	0.8300	54.63	12.73
	DDQ-SKD	**0.7645**	43.31	**14.78**	**0.8144**	**53.12**	13.91	**0.8347**	**57.01**	**12.56**
Restaurant Reservation										
Rule-based		0.1784	-19.56	30.09	0.1891	-20.11	31.52	0.1894	-20.20	30.91
DQN-based	DQN	0.3567	2.63	26.24	0.4098	8.13	22.19	0.4132	6.78	23.15
	VACL	0.3664	-8.49	24.93	0.4352	-1.81	23.96	0.4321	-2.17	24.05
	DQN-SKD	**0.4134**	**10.44**	**23.14**	**0.5143**	**15.14**	**18.09**	**0.5222**	**17.78**	**17.01**
DDQ-based	DDQ(10)	0.4356	12.56	21.45	0.5723	22.07	15.43	0.5678	19.01	14.23
	DR-D3Q	0.4233	14.43	20.23	0.5924	25.13	14.67	0.5762	24.13	13.91
	D3Q(10)	0.4566	15.33	**18.34**	0.5531	19.01	**13.91**	0.5626	23.15	17.35
	DDQ-SKD	**0.4716**	**16.01**	20.24	**0.6145**	**28.77**	14.89	**0.6241**	**25.16**	**12.33**
Taxi Ordering										
Rule-based		0.2791	-8.40	27.81	0.2809	-9.12	28.09	0.2894	-9.01	26.56
DQN-based	DQN	0.5133	15.32	20.13	0.6784	24.12	17.49	0.7136	27.43	16.31
	VACL	0.4814	3.75	21.96	0.6460	19.20	19.89	0.6556	20.23	19.56
	DQN-SKD	**0.5732**	**18.43**	**18.90**	**0.7123**	**27.31**	**16.76**	**0.7321**	**28.94**	**16.01**
DDQ-based	DDQ(10)	0.5567	16.71	19.12	0.6533	25.88	18.32	0.7451	30.13	15.67
	DR-D3Q	0.5331	16.43	18.23	0.7198	28.19	16.71	0.7261	25.78	15.94
	D3Q(10)	0.5812	20.44	16.12	0.7316	31.04	15.57	0.7819	34.41	14.13
	DDQ-SKD	**0.6256**	**23.19**	**15.79**	**0.7644**	**35.28**	**16.47**	**0.8201**	**36.37**	**15.01**

5.4. Simulation Evaluation

In this setting, the agents are optimized by conversing with rule-based user simulators (or world model trained to mimic real users in DDQ-based agents) following (Peng, Li, Gao, Liu, & Wong, 2018). Despite the gap between real users and user simulators, it allows us to perform detailed analysis of different agents without much cost and reproduce the experimental results easily.

5.4.1. Main results

The performances of agents at different epochs on the Microsoft Dialogue Challenge and MultiWOZ datasets are presented in Table 4 and Table 5 respectively. We evaluate the quality of all these agents in terms of success rate, average reward and average turns. Among them, success rate is the main metric indicating the proportion of successful conversations. Particularly, the conversation is deemed as successful when all the requests are filled in and the constraints are satisfied. The final results are averaged over 5 repetitions of experiments. We record the results at training epoch 100, 200 and 300. Normally, better performance at certain epoch means higher efficiency.

As shown in Table 4, it can be clearly observed that the RL-based agent (including DQN and DDQ-based agents) consistently outperforms rule-based ones significantly. Notably, the performance of RL-based agent barely exhibits evident promotion as the training epoch increases. This is because rule-based agent does not learn to optimize its policy from dialogue experiences. The advantage is that it normally achieves better results at the beginning of training when few dialogue experience is available, which is suitable for pre-filling the experience pool $\mathcal{D}^u$ before training.

When our method is combined with DQN agent, it outperforms the original DQN with a considerable margin in all training epochs in terms of three metrics. The results demonstrate that SKD can provide accurate and prompt rewards for the agent to explore the action space more efficiently. Particularly, when compared with the ACL-DQN and VACL agents, which also applied curriculum learning, DQN-SKD still boosts the success rate significantly. This is because the teacher policy model applied in the ACL-DQN agent is trained from scratch during policy optimization, which is naive and may arrange immature curriculum for the student policy model in the early training stage. VACL measures user goal difficulty by calculating the average cumulative return (reward) of corresponding sampled trajectories. However, the sparse and delayed reward does not necessarily reflect the user goal's difficulty. Comparatively, our method can address those aforementioned two problems. On the one hand, SKD pre-trains the teacher reward estimator in advance, so that the quality of curriculum arranged for the student model can be guaranteed during the whole training process. On the other hand, the teacher and student reward estimators in SKD, both providing meaningful and dynamic reward, are able to evaluate the dialogue difficulty more accurately and sufficiently.

When combined with DDQ agent, our method still achieves the best or competitive results among the DDQ-based agents. Specifically, DDQ-SKD promotes the final success rate by 5.53% compared with the original DDQ agent. We conclude that our method can improve the learning speed and performance of the dialogue agent with informative reward generated by the student reward estimator. Considering those agents that devise enriched reward function to address *reward sparsity* issue, e.g., DR-D3Q, ES-DDQ, our method outperforms them with a considerable margin. We attribute it to the knowledge distillation, where the student reward estimator benefits a lot from the teacher model, which is embedded with rich user goal and historical dialogue state information. Moreover, DDQ-SKD outperforms DDQ-SDPL in Movie-Ticket Booking task, which joints curriculum learning and dialogue policy optimization, by 0.45%, 1.44% and 0.47% at 100, 200 and 300 epochs. This is because DDQ-SDPL sorts all training samples from easy to hard in advance regardless of the learning process of dialogue policy. By contrast, our method defines the complexity of each sample by monitoring the feedback of the student reward estimator, which can adjust the training schedule more flexible and efficient.

Table 5. Experimental results on MultiWOZ dataset. Best results are marked in bold.

Agent		Epoch=100			Epoch=200			Epoch=300		
		Success	Reward	Turns	Success	Reward	Turns	Success	Reward	Turns
DQN-based	DQN	0.4321	13.57	15.61	0.4789	19.15	16.14	0.5577	26.42	14.16
	DQN-SDPL	0.4500	2.51	**13.49**	0.5200	11.04	13.36	0.5700	12.22	13.81
	DQN-SKD	**0.4912**	**21.66**	13.51	**0.5581**	**26.17**	**12.42**	**0.5914**	**28.19**	**12.45**
DDQ-based	DDQ(10)	0.5132	23.19	16.12	0.5714	27.35	15.33	0.5963	28.27	16.24
	DR-D3Q	0.6177	25.66	15.67	0.6231	31.64	16.79	0.6201	30.56	15.66
	D3Q(10)	0.5735	23.91	14.85	0.6047	29.98	14.16	0.6316	**32.61**	13.91
	DDQ-SKD	**0.6317**	**27.15**	**14.19**	**0.6434**	**34.01**	**13.67**	**0.6536**	31.98	**13.41**

Table 5 presents the main results on MultiWOZ dataset, which is more complicated since it contains more domains and dialogue turns. Experimental results indicate that SKD achieves state-of-the-art performance compared with both DQN and DDQ-based agents. Concretely, DQN-SKD outperforms DQN-SDPL by 2.14% in terms of success rate, which is even higher than the improvement of 0.47% in the Movie-Ticket Booking task. The results proved quantitative evidence that SKD consistently outperforms SDPL when combined with DQN and DDQ agents. Besides, it suggests that our method is superior in arranging more reasonable schedules under complicated situations. On the contrary, for the DDQ-based agents, mostly addressing reward sparsity and hysteresis problems (i.e., DR-D3Q and D3Q), the improvements are generally lower in MultiWOZ than those in MDC dataset. It suggests that the reward estimator works better in simpler conversations.

5.4.2. Ablation study

To evaluate the impact of each module on the final performance and dialogue learning speed, we conduct several ablation experiments by removing each component respectively.

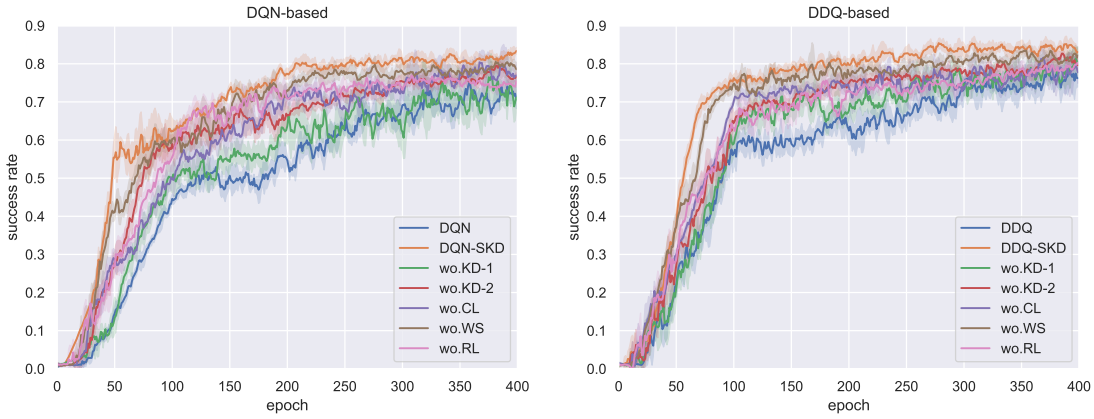
- **knowledge distillation:** train the student reward estimator without knowledge distillation (wo.KD-1). Furthermore, we implement another baseline that employs an auxiliary weakly supervised learning objective to train the student: $\mathcal{L}_{WS}^{stu} = \mathbb{E}(\|\mathbf{r} - \mathbf{r}^{stu}\|_2)$ (wo.KD-2).
- **curriculum learning:** remove the example weights $\mathbf{v}$ in Equation 15, i.e., utilize Equation 13 to transfer knowledge to the student (wo.CL).
- **weak supervision:** pre-train the teacher reward estimator without weak supervision (wo.WS).
- **reinforcement learning:** pre-train the teacher reward estimator with out reinforcement learning. (wo. RL)

The final results are presented in Table 6 and Fig. 5. Generally, performance degradation can be observed after removing part of the modules in both DQN and DDQ-based agents. Among them, directly removing the knowledge distillation (wo.KD-1) results in the most severe performance drop with 7.84% and 6.35% success rate degradation compared with DQN-SKD and DDQ-SKD respectively. The results suggest that it is insufficient to properly reward actions without the guidance of teacher reward estimator, which may even misguide the dialogue policy and make it stuck into local optima. When the student model is trained with weakly supervised learning

Table 6. Success rate of removing the curriculum learning (CL), knowledge distillation (KD), weak supervision (WS) and reinforcement learning (RL) on Movie-Ticket Booking dataset.

Model	Epoch=100	Epoch=200	Epoch=300	Epoch=400
DQN-SKD	0.6418	0.7716	0.8128	0.8145
wo.KD-1	0.5218	0.6168	0.6841	0.7361
wo.KD-2	0.5584	0.6916	0.7616	0.7841
wo.CL	0.5091	0.7329	0.7960	0.7946
wo.WS	0.6010	0.7563	0.7746	0.8012
wo.RL	0.5557	0.7081	0.7314	0.7535
DDQ-SKD	0.7645	0.8144	0.8447	0.8367
wo.KD-1	0.6045	0.6995	0.7631	0.7732
wo.KD-2	0.6433	0.7513	0.7654	0.8110
wo.CL	0.6896	0.7266	0.7789	0.7981
wo.WS	0.7292	0.8012	0.8141	0.8257
wo.RL	0.6320	0.7288	0.7548	0.8032

(i.e., wo.KD-2), the performance degradation can be greatly alleviated. This can be explained by the fact that user goal plays an essential role in evaluating the effective of actions and task success. Surprisingly, it performs even better to train the student model on the teacher (i.e., DQN-SKD and DDQ-SKD) than directly on the annotated data (wo.KD-2). This is because the teacher model has superiority in learning the internal features contained in the annotated data with the help of BERT and user goal information. And training on the soft labels predicted by the teacher makes learning easier for the student model by focusing the relationships learned by the teacher across all the actions.



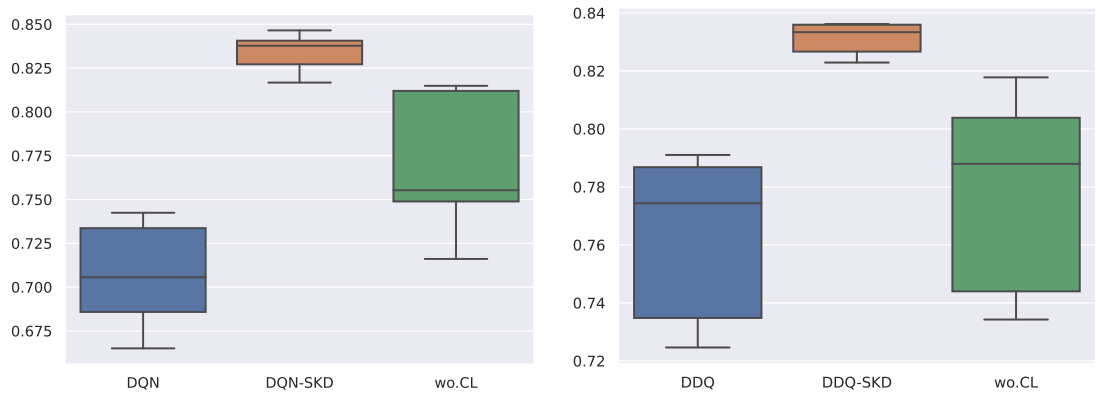
(a) DQN-based agents

(b) DDQ-based agents

Figure 5. Learning curves of agents discarding different components on Movie Ticket Booking dataset.

When curriculum learning is discarded, the success rate drops 1.99% and 3.86% compared with DQN-SKD and DDQ-SKD respectively. Learning curves depicted in Fig. 5 also shows significant learning speed degradation. Fig. 6 is the boxplots of different agents at 400 training epochs with 5 repetitive experiment results. Obviously, DQN-SKD and DDQ-SKD agents are more stable than those trained without curriculum learning. The results demonstrate that curriculum learning used to train the student reward estimator can not only promote the final performance but also enhance

the stability of policy learning.



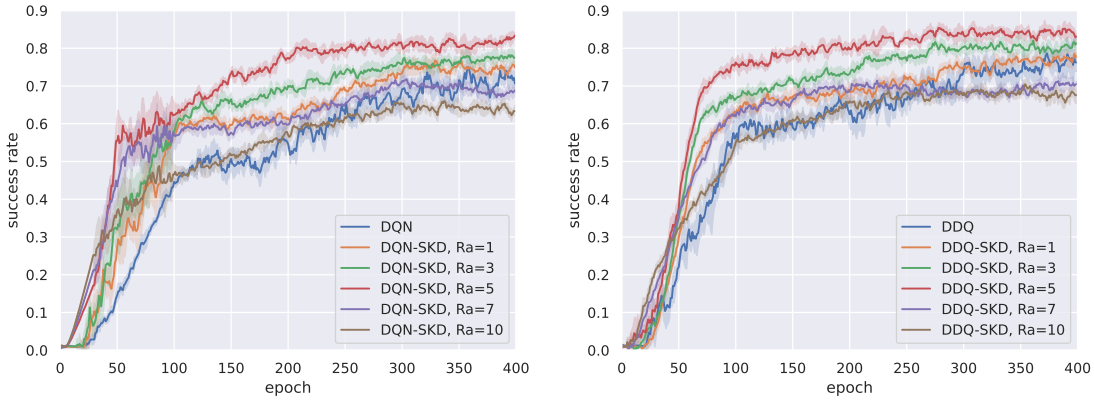
(a) DQN-based agents (b) DDQ-based agents

Figure 6. Box plots of DQN and DDQ based agents on Movie-Ticket Booking dataset. The center, bottom and top line of the box show the median, 25^{th} and 75^{th} percentiles respectively. The lines that extend from the box represent the expected variation of the data. The shorter the box, the more stable it is.

When the teacher reward estimator is pre-trained without weak supervision, the learning curves depict relatively mild performance decay. This is because the teacher model benefiting from the user goal information, which serves as the input of the teacher model, can easily learn knowledge of task completion and thus make up the absence of weakly supervised guidance. Pre-training teacher reward estimator without RL degrades the final performance by 6.10% and 3.35% on compared to DQN-SKD and DDQ-SKD respectively. We believe that human-machine interactions in RL framework enables the reward estimator to perceive more dialogue actions that are not presented in the annotated data, and thus improves the quality of generated reward.

5.4.3. Different choices of Anchor reward

To explore how different choices of the *anchor reward* R_a influence the final performance and learning speed of the agents, we present learning curves of agents with different anchor rewards in Fig. 7. We observe that weak supervision generally promotes the learning speed at the early training stage as R_a gets larger, as can be observed from the first 50 epochs of DDQ-based agents. However, as the training progresses, agents with extremely large R_a gradually lags behind those with relatively small R_a. For example, DDQ-SKD achieves about 50% success rate with $R_a = 10$ at epoch 100, where it achieves more than 75% with $R_a = 5$. In fact, the performance is gradually improved as R_a increases until $R_a = 5$ and declined as it continues to grow. We conclude those observations as follows: while weak supervision can accelerate the convergence of dialogue policy, the optimal anchor reward R_a should be determined seriously to balance the trade-off between local guidance and global reward. More specifically, When R_a is too small, the weak supervision may have little impact on guiding the teacher model to generate proper reward reflecting the action validity. When R_a is too large, the agent tends to focus on short term reward rather than global optimum (i.e., accomplishing the task). Overemphasis on fitting the weakly annotated data may harm the final performance.



(a) DQN-based

(b) DDQ-based

Figure 7. Learning curves of DQN-SKD and DDQ-SKD with different anchor reward R_a.

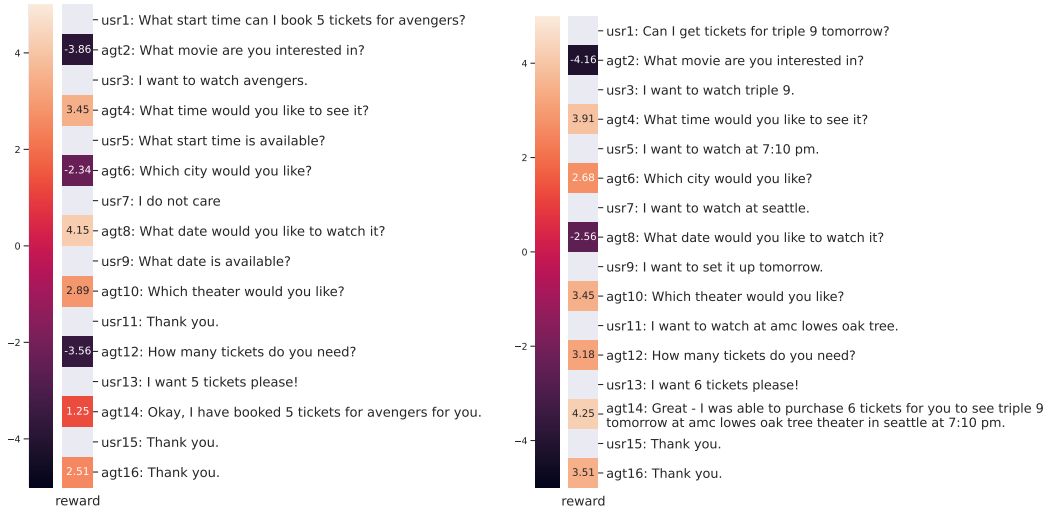
5.4.4. Reward visualization

In order to demonstrate the strength of our reward estimator in providing proper reward reflecting action validity and task completion, we present Fig. 8 that illustrates the reward provided by the student reward estimator at the end of each agent response. To better show the ability of the reward estimator to evaluate the action, we employ a rule-based agent, which requests information in a fixed order, to interact with the user simulator.

As shown in Fig. 8, it is clear the reward can generally reflect the dialogue state and action validity of the agent. To be specific, the reward estimator is capable of assigning penalty to discouraging duplicated actions at turn 2 of both dialogues, where the agent requests the movie name identified at the last turn by the user. Similar observations can be seen from turn 12 of dialogue 8a. Besides, even if the agent takes the same action, the agent is able to provide reward according to the dialogue states and agent actions. For example, when the agent asks for the location at turn 6, the reward estimator assigns a negative reward to this action in dialogue 8a, which does not answer the inquiry of the user about start time. In comparison, the reward assigned to the action at turn 6 is positive in dialogue 8b, which is reasonable. This is because the reward estimator taking historical dialogue states as input can avoid rewarding duplicated or invalid actions. Another interesting phenomenon can be observed at turn 14, where the agent books tickets for the user in both dialogues. The reward is relatively small (1.25) since it does not meet all requirements of the user in dialogue 8a, while it is rather large (4.25) when the agent successfully fulfill the task in dialogue 8b. This is attributable to the knowledge distillation that makes the student reward estimator anticipate user goal and task completion information.

5.4.5. Performance of dealing with shifty user goals

To illustrate the superiority of our method in dealing with shifty user goals, we develop a user simulator that randomly changes part of user goals during conversation and let it interact with different agents that are trained with 300 epochs on MultiWOZ dataset. We test them on 100 dialogues for each agent. The results are shown in Fig. 9. Not surprisingly, our method still achieves the best result among all DDQ-based agents. Compared with D3Q and DR-D3Q, which degrade the success rate by 13.34% and



(a) (b)

Figure 8. Reward visualization of a dialogue simulation between a rule based agent and a user simulator. The reward is provided by a trained student reward estimator.

9.49% on shifty user goals respectively, our method exhibits the mildest performance decay by 2.36%. This is because the teacher reward estimator learns prior distribution of user goal information and thus enables the system to handle user goals more flexible. Another potential reason may be that the system can act more properly when the policy is combined with reward functions during interaction.

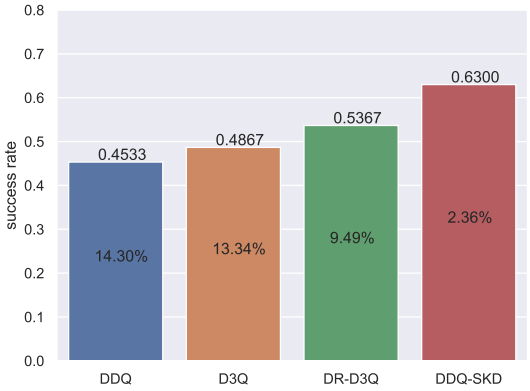


Figure 9. Success rate of different agents dealing with shifty user goals on MultiWOZ. The number in each bar represents success rate degradation compared to that in Table 6 at Epoch=300.

5.4.6. The effect of different complexity measuring strategies

In this section, we further explore the effects of different criterion of complexity measuring for self-paced learning. In addition to the KL loss between the output of teacher and student reward estimator (**Stu-Tea**) stated in Equation 14, we consider other two different metrics to evaluate the dialogue complexity, which is defined as the KL loss between the true label $\mathbf{p}$ and (1) the output of teacher $\mathbf{p}^{tea}$ (**Gold-Tea**), (2) the output of the student $\mathbf{p}^{stu}$ (**Gold-Stu**), where $\mathbf{p} = \mathbf{r}/R_a$. Besides, we also measure the

Table 7. Experimental results of DDQ-SKD with different complexity measurer on MultiWOZ dataset. wo. CL means without curriculum learning.

Complexity measurer	Epoch=100			Epoch=200			Epoch=300		
	Success	Reward	Turns	Success	Reward	Turns	Success	Reward	Turns
Stu-Tea	**0.6317**	**27.51**	**14.19**	**0.6434**	**34.01**	**13.67**	**0.6536**	31.98	**13.41**
Gold-Tea	0.6178	26.71	16.43	0.6281	29.76	14.57	0.6458	**32.51**	13.62
Gold-Stu	0.5315	22.47	15.44	0.5754	26.52	16.78	0.5999	28.14	15.79
Rule	0.5589	21.01	17.43	0.5671	25.51	17.10	0.6031	27.42	15.22
wo. CL	0.4918	20.43	18.91	0.5549	23.91	17.49	0.5838	24.76	16.44

dialogue difficulty based on the number of slots N contained in the corresponding user goal (**Rule**). Specifically, $score = N/N_{max}$, N_{max} is a constant representing the maximum number of slots of user goal in the dataset. Intuitively, the more slots, the more difficult it is. We conduct experiments on MultiWOZ dataset based on DDQ-SKD agent.

The results are presented in Table 7. Generally, Stu-Tea outperforms others on most metrics. More concretely, even though the golden labels are involved in complexity measuring, Stu-Tea still shows better performances. The results suggest that evaluating the difference between outputs of teacher and student models can better reflect the student's learning process. Comparatively, Rule achieves the worst results among all complexity measurer variants, while still outperforms wo. CL. We conclude that the number of slots in a user goal can only partially judge the complexity compared with other measurements.

5.4.7. Case study

Table 8 presents dialogue examples of simulated user interacting with different agents. It is obvious that the DQN and DDQ agents can accomplish the task more efficiently when they are combined with our method. More specifically, the DQN agent tends to perform like a naive rule-based agent that requests information in predefined order. And it asks for the theater information repeatedly (*Which theater would you like?*) when the user does not answer it properly, which fails to accomplish the task within the maximum turns. As for the DDQ agent, although it books two tickets for the user, several important information, including date, theater and start time, are neglected. Furthermore, it may request information that has already been identified in the context. For example, the DDQ agent asks the user for the preferred movie at turn 2 and the number of tickets at turn 8, which are both identified when the conversation starts, i.e., at turn 1. Such problems may decrease the efficiency of completing tasks, and even make real users impatient and terminate the conversation early.

In comparison, our methods avoid these problems and accomplish the tasks perfectly. We attribute this to the following factors: (1) Reward annotations are capable of preventing the agent from performing redundantly, since the weak reward annotator allocates a negative reward to discourage such reduplicate behaviors by evaluating sub-goals completion. (2) The student reward estimator, learning rich contextual and user goal anticipated information from the teacher model, is able to guide the agent to explore the action space more effectively.

Table 8. Dialogue examples of simulated user interacting with different agents trained at 100 epochs. *usr* denotes user, while *agt* denotes agent.

$\mathcal{G} = \left(\begin{array}{l} \mathcal{C} = \left[\begin{array}{l} Date : tomorrow \\ City : detroit \\ MoiveName : deadpool \\ StartTime : 7pm \end{array} \right] \\ \mathcal{R} = \left[\begin{array}{l} Theater \end{array} \right] \end{array} \right)$	$\mathcal{G} = \left(\begin{array}{l} \mathcal{C} = \left[\begin{array}{l} City : duQuoin \\ NumberofPeople : 2 \\ MovieName : StarWars \end{array} \right] \\ \mathcal{R} = \left[\begin{array}{l} Date \\ Thether \\ StartTime \end{array} \right] \end{array} \right).$
DQN	**DDQ**
usr_1: What theater will play the deadpool at 7pm tomorrow? agt_2: What movie are you interested in? usr_3: I want to watch deadpool. agt_4: Which city would you like? usr_5: I want to watch at detroit. agt_6: What date would you like to watch it? usr_7: I want to set it up tomorrow. agt_8: Which theater would you like? usr_9: Which theater is available? agt_{10}: Which theater would you like? ...	usr_1: What start time can I book 2 tickets for star wars? agt_2: What movie are you interested in? usr_3: I want to watch star wars. agt_4: What time would you like to see it? usr_5: Any time is fine. agt_6: Which city would you like? usr_7: I want to watch at du Quoin. agt_8: How many tickets do you need? usr_9: I want 2 tickets please. agt_{10}: Ok, your purchase of 2 tickets for star wars at du Quoin is confirmed. usr_{11}: Oh, sorry this is wrong ticket. agt_{12}: Thank you.
Failure: the conversation is repeated and the agent is not able to book the ticket within the maximum turns.	Failure: the agent is not able to book the tickets with all requests identified.
DQN-SKD	**DDQ-SKD**
usr_1: What theater will play the dead pool at 7pm tomorrow? agt_2: Which city would you like? usr_3: I want to watch at detroit. agt_4: How many tickets do you want? usr_5: I want 3 tickets please! agt_6: Great, I was able to book 3 tickets for you to watch dead pool at *theater* in detroit at 7pm tomorrow! usr_7: Thank you. agt_8: Thank you.	usr_1: What start time can I book 2 tickets for star wars? agt_2: Which city would you like? usr_3: I want to watch at du Quoin. agt_4: What date do you want? usr_5: Any day will do. agt_6: How about 2 tickets for star wars at du Quoin for tomorrow at *theater*? usr_7: What time will it start? agt_8: It starts at 7pm. usr_9: Ok, thank you. usr_{10}: Thank you.
Success	Success

5.5. Human Evaluation

We recruit real users to evaluate the performance of four agents, including DQN, DDQ, DQN-SKD and DDQ-SKD, by conversing with them. All evaluated agents are trained with 100 epochs. In each dialogue session, the user is asked to select one of the agents without knowing the identification of the selected agent. At the beginning of each conversation, the user is presented with a user goal randomly sampled from the corpus and interacts with the agent to accomplish the task. The user can choose to abandon the task and terminate the dialogue if the conversation is deemed too duplicated or the user believes the task is unlikely to be completed. Such dialogue sessions are then considered to be failed. The user is required to provide feedback on each conversation regarding task success. We evaluate these agents in 400 dialogues in total.

As shown in Fig. 10, we can observe that the success rate of all the agents are dropped with a significant margin compared with those in simulation experiments. It is not surprising since the real user, who is not as patient as the simulated user, may terminate the conversation within a small number of turns before the task is

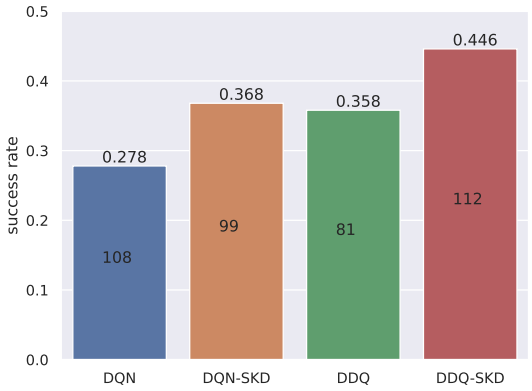


Figure 10. Human evaluation results of DQN, DQN-SKD, DDQ and DDQ-SKD agents. The number of test dialogues are presented in on each bar.

accomplished. Besides, it is clear that the success rate of DQN and DDQ agents are improved by 9.0% and 8.8% when combined with our method, which accords with the observations in the simulation experiments.

6. Conclusion and future work

In this paper, we propose a novel method called scheduled knowledge distillation (SKD), which learns dynamic reward function by combining knowledge distillation and curriculum learning for efficient dialogue policy learning. In knowledge distillation, we first pre-train a teacher reward estimator that benefits from the deep semantic information contained in BERT and user goal information via deep Q-learning. Besides, we also construct weak annotated data based on the sub-goals completion and train the teacher model by weakly supervised learning. Then, the knowledge is transferred to the student reward estimator by training it to mimic the outputs of the teacher model. In curriculum learning, we leverage the divergence between outputs of the teacher and student reward estimators to evaluate the dialogue complexity, and train the student model from easy to hard samples based on the evaluation results. We conduct experiments on two popular task-oriented datasets and results demonstrate the efficiency and effectiveness of our method.

Although our method is combined with DQN and DDQ algorithms, the main contribution of our work SKD can also be easily applied to other value-based approaches, e.g., Q-learning, DDQN, we will explore it in the future. Besides, our method causes extra training cost due to the pre-training of teacher reward estimator, and we will try to reduce the computational complexity in our future work.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Data availability

The Movie-ticket booking dataset analyzed during the current study is available in the TC-BOT repository (https://github.com/MiuLab/TC-Bot).

References

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning* (pp. 41–48).

Budzianowski, P., Wen, T.-H., Tseng, B.-H., Casanueva, I., Ultes, S., Ramadan, O., & Gasic, M. (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* (pp. 5016–5026).

Chen, H., Liu, X., Yin, D., & Tang, J. (2017). A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, *19*(2), 25–35.

Chu, G., Wang, X., Shi, C., & Jiang, X. (2021). Cuco: Graph representation with curriculum contrastive learning. In *Ijcai* (pp. 2300–2306).

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Nips 2014 workshop on deep learning, december 2014*.

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018). Generative adversarial networks: An overview. *IEEE signal processing magazine*, *35*(1), 53–65.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dhingra, B., Li, L., Li, X., Gao, J., Chen, Y.-N., Ahmad, F., & Deng, L. (2017). Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 484–495).

Diakoloukas, V., Lygerakis, F., Lagoudakis, M. G., & Kotti, M. (2020). Variational denoising autoencoders and least-squares policy iteration for statistical dialogue managers. *IEEE Signal Processing Letters*, *27*, 960–964.

Dong, X., Long, C., Xu, W., & Xiao, C. (2021). Dual graph convolutional networks with transformer and curriculum learning for image captioning. In *Proceedings of the 29th acm international conference on multimedia* (pp. 2615–2624).

El-Bouri, R., Eyre, D., Watkinson, P., Zhu, T., & Clifton, D. (2020). Student-teacher curriculum learning via reinforcement learning: predicting hospital inpatient admission location. In *International conference on machine learning* (pp. 2848–2857).

Geishauser, C., van Niekerk, C., Lin, H.-C., Lubis, N., Heck, M., Feng, S., & Gasic, M. (2022). Dynamic dialogue policy for continual reinforcement learning. In *Proceedings of the 29th international conference on computational linguistics* (pp. 266–284).

Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge distillation: A survey. *International Journal of Computer Vision*, *129*(6), 1789–1819.

Greco, A., Saggese, A., Vento, M., & Vigilante, V. (2021). Effective training of convolutional neural networks for age estimation based on knowledge distillation. *Neural Computing and Applications*, 1–16.

Guo, S., Huang, W., Zhang, H., Zhuang, C., Dong, D., Scott, M. R., & Huang, D. (2018). Curriculumnet: Weakly supervised learning from large-scale web images. In *Proceedings of the european conference on computer vision (eccv)* (pp. 135–150).

Haidar, M., Rezagholizadeh, M., et al. (2019). Textkd-gan: Text generation using knowledge distillation and generative adversarial networks. In *Canadian conference on artificial intelligence* (pp. 107–118).

Hinton, G., Vinyals, O., Dean, J., et al. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, *2*(7).

Hosseini-Asl, E., McCann, B., Wu, C.-S., Yavuz, S., & Socher, R. (2020). A simple language model for task-oriented dialogue. *Advances in Neural Information Processing Systems*, *33*, 20179–20191.

Jeon, H., & Lee, G. G. (2022). Dora: Towards policy optimization for task-oriented dialogue system with efficient context. *Computer Speech & Language*, *72*, 101310.

Khandelwal, A. (2021). Weasul: Weakly supervised dialogue policy learning: Reward estimation for multi-turn dialogue. In *Proceedings of the 14th international conference on natural language generation* (pp. 64–75).

Li, B., Wang, Z., Liu, H., Du, Q., Xiao, T., Zhang, C., & Zhu, J. (2021). Learning light-weight translation models from deep transformer. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 35, pp. 13217–13225).

Li, X., Chen, Y.-N., Li, L., Gao, J., & Celikyilmaz, A. (2017). End-to-end task-completion neural dialogue systems. In *Proceedings of the eighth international joint conference on natural language processing (volume 1: Long papers)* (pp. 733–743).

Li, X., Lipton, Z. C., Dhingra, B., Li, L., Gao, J., & Chen, Y.-N. (2016). A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*.

Li, X., Wang, Y., Sun, S., Panda, S., Liu, J., & Gao, J. (2018). Microsoft dialogue challenge: Building end-to-end task-completion dialogue systems. *arXiv preprint arXiv:1807.11125*.

Li, Z., Kiseleva, J., & de Rijke, M. (2020). Rethinking supervised learning and reinforcement learning in task-oriented dialogue systems. *arXiv preprint arXiv:2009.09781*.

Lipton, Z. C., Gao, J., Li, L., Li, X., Ahmed, F., & Deng, L. (2016). Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking. *arXiv preprint arXiv:1608.05081*, *3*.

Liu, J., Chen, Y., & Liu, K. (2019). Exploiting the ground-truth: An adversarial imitation based knowledge distillation approach for event detection. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, pp. 6754–6761).

Liu, S., Zhang, J., He, K., Xu, W., & Zhou, J. (2021). Scheduled dialog policy learning: An automatic curriculum learning framework for task-oriented dialog system. In *Findings of the association for computational linguistics: Acl-ijcnlp 2021* (pp. 1091–1102).

Lu, K., Zhang, S., & Chen, X. (2019). Goal-oriented dialogue policy learning from failures. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, pp. 2596–2603).

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... others (2015). Human-level control through deep reinforcement learning. *nature*, *518*(7540), 529–533.

Narvekar, S., & Stone, P. (2019). Learning curriculum policies for reinforcement learning. In *Proceedings of the 18th international conference on autonomous agents and multiagent systems* (pp. 25–33).

Peng, B., Li, X., Gao, J., Liu, J., Chen, Y.-N., & Wong, K.-F. (2018). Adversarial advantage actor-critic model for task-completion dialogue policy learning. In *2018 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 6149–6153).

Peng, B., Li, X., Gao, J., Liu, J., & Wong, K.-F. (2018). Deep dyna-q: Integrating planning for task-completion dialogue policy learning. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 2182–2192).

Platanios, E. A., Stretcu, O., Neubig, G., Póczos, B., & Mitchell, T. (2019). Competence-based curriculum learning for neural machine translation. In *Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)* (pp. 1162–1172).

Qu, M., Tang, J., & Han, J. (2018). Curriculum learning for heterogeneous star network embedding via deep reinforcement learning. In *Proceedings of the eleventh acm international conference on web search and data mining* (pp. 468–476).

Ren, S., Guo, K., Ma, J., Zhu, F., Hu, B., & Zhou, H. (2021). Realistic medical image super-resolution with pyramidal feature multi-distillation networks for intelligent healthcare

systems. *Neural Computing and Applications*, 1–16.
Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., & Bengio, Y. (2014). Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.
SCHATZMANN, J., & YOUNG, S. (2009). The hidden agenda user simulation model. *IEEE transactions on audio, speech, and language processing*, *17*(4), 733–747.
Shen, Y., Xu, X., & Cao, J. (2020). Reconciling predictive and interpretable performance in repeat buyer prediction via model distillation and heterogeneous classifiers fusion. *Neural Computing and Applications*, *32*(13), 9495–9508.
Su, P.-H., Gašić, M., & Young, S. (2018). Reward estimation for dialogue policy optimisation. *Computer Speech & Language*, *51*, 24–43.
Su, S.-Y., Li, X., Gao, J., Liu, J., & Chen, Y.-N. (2018). Discriminative deep dyna-q: Robust planning for dialogue policy learning. In *Emnlp.*
Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990* (pp. 216–224). Elsevier.
Tian, C., Yin, W., & Moens, M. F. (2022). Anti-overestimation dialogue policy learning for task-completion dialogue system. In *Findings of the association for computational linguistics: Naacl 2022* (pp. 565–577).
Tian, Z., Bi, W., Lee, D., Xue, L., Song, Y., Liu, X., & Zhang, N. L. (2020). Response-anticipated memory for on-demand knowledge integration in response generation. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 650–659).
Wang, X., Chen, Y., & Zhu, W. (2021). A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
Wang, X., Fu, T., Liao, S., Wang, S., Lei, Z., & Mei, T. (2020). Exclusivity-consistency regularized knowledge distillation for face recognition. In *European conference on computer vision* (pp. 325–342).
Wang, Y., Wang, W., Liang, Y., Cai, Y., & Hooi, B. (2021). Curgraph: Curriculum learning for graph classification. In *Proceedings of the web conference 2021* (pp. 1238–1248).
Wu, G., Fang, W., Wang, J., Cao, J., Bao, W., Ping, Y., ... Wang, Z. (2021). Gaussian process based deep dyna-q approach for dialogue policy learning. In *Findings of the association for computational linguistics: Acl-ijcnlp 2021* (pp. 1786–1795).
Wu, W., Guo, Z., Zhou, X., Wu, H., Zhang, X., Lian, R., & Wang, H. (2019). Proactive human-machine conversation with explicit conversation goal. In *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 3794–3804).
Zhang, R., Wang, Z., Zheng, M., Zhao, Y., & Huang, Z. (2021). Emotion-sensitive deep dyna-q learning for task-completion dialogue policy learning. *Neurocomputing*, *459*, 122–130.
Zhao, Y., Qin, H., Zhenyu, W., Zhu, C., & Wang, S. (2022). A versatile adaptive curriculum learning framework for task-oriented dialogue policy learning. In *Findings of the association for computational linguistics: Naacl 2022* (pp. 711–723).
Zhao, Y., Wang, Z., & Huang, Z. (2021). Automatic curriculum learning with over-repetition penalty for dialogue policy learning.
Zhao, Y., Wang, Z., Yin, K., Zhang, R., Huang, Z., & Wang, P. (2020). Dynamic reward-based dueling deep dyna-q: Robust policy learning in noisy environments. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 34, pp. 9676–9684).
Zhao, Y., Wang, Z., Zhu, C., Wang, S., Moens, M.-F., Huang, X., ... others (2021). Efficient dialogue complementary policy learning via deep q-network policy and episodic memory policy. In *Proceedings of the 2021 conference on empirical methods in natural language processing* (pp. 4311–4323).
Zhou, X., Zhu, F., & Zhao, P. (2022). Predicting before acting: improving policy quality by taking a vision of consequence. *Connection Science*, *34*(1), 608–629.
Zhu, H., Zhao, Y., & Qin, H. (2021). Cold-started curriculum learning for task-oriented dialogue policy. In *2021 ieee international conference on e-business engineering (icebe)* (pp. 100–105).

Zhu, Q., Zhang, Z., Fang, Y., Li, X., Takanobu, R., Li, J., ... Huang, M. (2020). Convlab-2: An open-source toolkit for building, evaluating, and diagnosing dialogue systems. *arXiv preprint arXiv:2002.04793*.