Agent-Aware Dropout DQN for Safe and Efficient On-line Dialogue Policy Learning

Lu Chen and **Xiang Zhou** and **Cheng Chang** and **Runzhe Yang** and **Kai Yu**
Key Lab. of Shanghai Education Commission for Intelligent Interaction and Cognitive Eng.
SpeechLab, Department of Computer Science and Engineering
Brain Science and Technology Research Center
Shanghai Jiao Tong University, Shanghai, China
{chenlusz,owenzx,cheng.chang,yang_runzhe,kai.yu}@sjtu.edu.cn

Abstract

Hand-crafted rules and reinforcement learning (RL) are two popular choices to obtain dialogue policy. The rule-based policy is often reliable within predefined scope but not self-adaptable, whereas RL is evolvable with data but often suffers from a bad initial performance. We employ a *companion learning* framework to integrate the two approaches for *on-line* dialogue policy learning, in which a predefined rule-based policy acts as a teacher and guides a data-driven RL system by giving example actions as well as additional rewards. A novel *agent-aware dropout* Deep Q-Network (AAD-DQN) is proposed to address the problem of when to consult the teacher and how to learn from the teacher's experiences. AAD-DQN, as a data-driven student policy, provides (1) two separate experience memories for student and teacher, (2) an uncertainty estimated by dropout to control the timing of consultation and learning. Simulation experiments showed that the proposed approach can significantly improve both *safety* and *efficiency* of on-line policy optimization compared to other companion learning approaches as well as supervised pre-training using static dialogue corpus.

1 Introduction

A task-oriented spoken dialogue system (SDS) is a system that can continuously interact with a human to accomplish a predefined task through speech. Dialogue manager, which maintains the dialogue state and decides how to respond, is the core of an SDS. In this paper, we focus on the dialogue policy.

At the early research, the spoken dialogue systems assume observable dialogue states. Dialogue policy is simply a set of hand-crafted mapping rules from state to machine action. This is referred to as rule-based policy, which often has acceptable performance but has no ability of self-adaption. Nowadays rule-based policy is popular in commercial dialogue systems.

However, in real world scenarios, unpredictable user behavior, inevitable automatic speech recognition, and spoken language understanding errors make it difficult to maintain the true dialogue state and make the decision. Hence, in recent years, there is a research trend towards statistical dialogue management. A well-founded theory for this is the partially observable Markov decision process (POMDP) (Kaelbling et al., 1998), which can provide robustness to errors from the input module and automatic policy optimization by reinforcement learning. Most POMDP based policy learning research is usually carried out using either user simulator or employed users (Williams and Young, 2007; Young et al., 2010). The trained policy is not guaranteed to work well in real world scenarios. Therefore, on-line policy training has been of great interest (Gašić et al., 2011). Recently, Chen et al. (2017) proposed two qualitative metrics [1] to measure on-line policy learning: *safety* and *efficiency*. Safety reflects whether the initial policy can satisfy the quality-of-service requirement in real-world scenarios during the on-line policy learning period. Efficiency reflects how long it takes for the on-line policy training algorithm to reach a satisfactory performance level.

Most traditional RL-based policy training suf-

[1] The quantitative evaluation metrics of safety and efficiency are proposed in section 4.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2454–2464
Copenhagen, Denmark, September 7–11, 2017.

fers poor initial performance, i.e. causes the safety problem. In light of above, Chen et al. (2017) proposed a safe and efficient on-line policy optimization framework, i.e. *companion teaching* (CT), in which a human teacher is added in the classic POMDP. The teacher has two missions: one is to show example actions, another is to act as a critic to give the student extra reward which can make the learning of policy more efficient. The example actions not only make the learning safer but also can be directly used by the training of the student policy. However, there are costs to the teaching of a human teacher.

Based on CT, *companion learning* (CL) framework is proposed to integrate rule-based policy and RL-based policy, resulting in *safe* and *efficient* on-line policy learning. Here, the rule-based policy acts as a virtual teacher which replaces the human teacher in CT. There are a few differences between these two kinds of teachers. First, because it has no marginal cost when it's deployed, the rule teacher can be consulted at any time if needed. On the other hand, the rule policy is not as good as the human teacher, therefore it's important to determine when and how much the student policy depends on the rule teacher. Here, we propose an *agent-aware dropout* Deep Q-Network (AAD-DQN) as the student statistical policy, which provides (1) two separate experience replay pools for student and teacher, (2) an uncertainty estimated by dropout which can be used to control the timing of consultation and learning.

In summary, our main contributions are three-folds: (1) *Companion learning* (CL) framework was proposed to integrate rule-based policy and RL-based policy. (2) An *agent-aware dropout* Deep Q-Network (AAD-DQN) was proposed as the statistical student policy. (3) Compared with other companion teaching approaches (Chen et al., 2017) as well as supervised pre-training using static dialogue corpus (Fatemi et al., 2016), CL with AAD-DQN can achieve better performance.

2 Related Work

Most previous studies of on-line policy learning have been focused on the *efficiency* issue, such as Gaussian Process Reinforcement Learning (GPRL) (Gašić et al., 2010). In GPRL, the kernel function defines prior correlations of the objective function given different belief states, which can significantly speed up the policy learning (Gašić and Young, 2014). Alternative methods include Kalman temporal difference reinforcement learning (Pietquin et al., 2011).

More recently, deep reinforcement learning (DRL) (Mnih et al., 2015) is applied in dialogue policy optimization, including deep Q-Network (DQN) (Cuayáhuitl et al., 2015; Fatemi et al., 2016; Zhao and Eskenazi, 2016; Lipton et al., 2016) and policy gradient (PG) methods, e.g. REINFORCE (Williams and Zweig, 2016; Su et al., 2016; Williams et al., 2017), Advantage Actor-Critic (A2C) (Fatemi et al., 2016). In order to speed up the learning of DQN, Lipton et al. (2016) proposed an efficient exploration technique based on Thompson sample from a Bayesian neural network. Furthermore, they showed that using a few successful dialogues generated by a rule-based policy to pre-fill the replay buffer can benefit the learning at the beginning. To improve the efficiency of PG methods, policy network is initialized with supervised learning (SL) before RL training (Williams and Zweig, 2016; Williams et al., 2017; Su et al., 2016, 2017; Fatemi et al., 2016), which is similar to the idea in (Silver et al., 2016). However, combining RL with SL for dialogue policy optimization is not new. Henderson et al. (2008) were among the first to prove the benefits of combining supervised and reinforcement learning. In the experiments, we will compare CL with these pre-training methods.

Although the improvement of efficiency can benefit the safety of learning process, no matter how efficient the algorithm is, an unsafe on-line learned policy can lead to bad user experience at the beginning of learning period and consequently fail to attract sufficient real users to continuously improve the policy. Therefore, it is important to address the *safety* issue. There are few works about the safety issue of on-line dialogue policy optimization. Williams (2008) proposed a method for integrating business rules and POMDPs. The rules act as the action mask, i.e. the rules nominate a set of one or more actions, and the POMDP chooses the optimal action.

3 Proposed Framework

3.1 Companion Learning for On-line Policy Optimization

In the CL framework, there are two agents: one is the student policy, another is the teacher policy. Here, *teacher policy* is the extra part com-

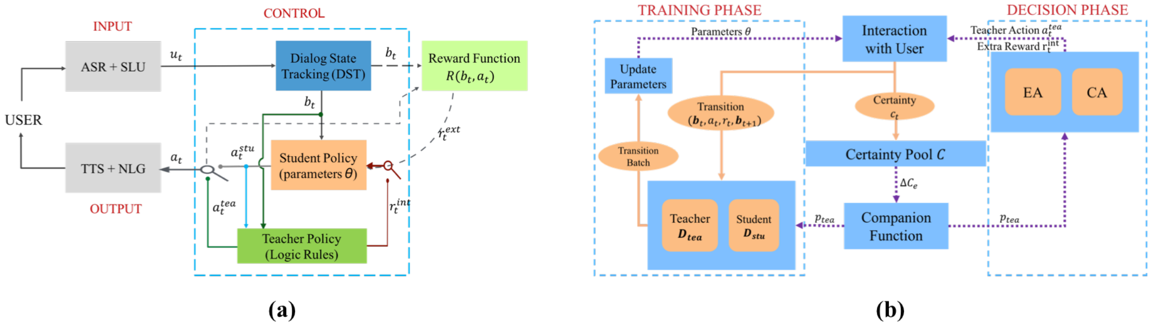

Figure 1: (a) RL-based Companion Learning(CL) Framework with Logic Rules in an SDS. (b) Agent-Aware Dropout DQN (AAD-DQN) for CL.

pared with the classic statistical dialogue manager architecture (Young et al., 2013). The goal of online policy training is to optimize the student policy from data via interaction with users in real scenarios. The teacher guides the policy learning at each turn as a companion of the dialogue policy, hence, referred to as *companion learning* [2]. The CL framework is described in Figure 1(a).

At each turn, the input module (ASR and SLU) receives an acoustic input signal from the human user and the dialogue state tracker keeps the dialogue state up-to-date. The dialogue state is then transmitted to both the student policy and the teacher policy. The student policy first generates a candidate action a_t^{stu} and when it needs help from the teacher policy, it sends a_t^{stu} with some auxiliary information which will be transmitted to the teacher. The teacher policy can then help the student policy with one of the following ways or both:

- **Example Action (EA)**: The teacher generates an action a_t^{tea} instead of a_t^{stu} according to its policy. It corresponds to the left switch in Figure 1(a).

- **Critic Advice (CA)**: The teacher will not explicitly show an action. Instead, it gives an extra reward r_t^{int} to the student policy. It corresponds to the right switch in Figure 1(a).

The action from control module is then transmitted to the output module, which generates the nature text and audio. At each turn, an extrinsic reward signal r_t^{ext} will be given to the student policy by the environment, i.e. the user. The extrinsic reward r_t^{ext} with the extra intrinsic reward r_t^{int} will be used to update the policy parameters θ using reinforcement learning algorithms.

In the CL framework, there are two things that matter: one is when to consult the teacher, another is how to use the teacher's experiences. In this paper, an *agent-aware dropout* DQN (AAD-DQN) is proposed. As shown in Figure 1(b), the certainty information during the interaction is used to define a *companion function*, which controls how often to sample the teacher's experiences for updating parameters during the training phase (left), and when to use EA or CA teaching method during decision phase (right).

The rest of this section is organized as follows. The next subsection introduces the *agent-aware* experience replay in DQN. The definition of certainty in DQN and the companion function are presented in subsection 3.3. The rule-based teacher policy is described in subsection 3.4.

3.2 Agent-Aware Experience Replay in DQN

A Deep Q-Network (DQN) is a multi-layer neural network which maps a belief state $\mathbf{b}_t$ to the Q values of the possible actions a_t at that state, $Q(\mathbf{b}_t, a_t; \theta)$, where θ is the weight vector of the neural network. Neural networks for the approximation of value functions have long been investigated (Lin, 1993). However, these methods were previously quite unstable (Mnih et al., 2013). In DQN, Mnih et al. (2013, 2015) proposed two techniques to overcome this instability, namely experience replay and the use of a target network.

At every turn, the transition including the previous belief state $\mathbf{b}_t$, previous action a_t, corresponding reward r_t and current belief state $\mathbf{b}_{t+1}$ is put in a finite pool (Lin, 1993). In this pa-

[2] The name *companion learning* has another potential meaning that the agents can learn from each other, i.e. the rules guide the RL training, and the optimised RL policy can provide some intuition for the revision of rules. We will give some preliminary discussions about this point in section 5.3.

per, two pools $\mathcal{D}_{stu}$ and $\mathcal{D}_{tea}$ are used to store the student's experiences and the teacher's experiences respectively as shown in Figure 1(b). When the teaching method EA is used in the t-th turn, $a_t = a_t^{tea}$ and the transition is put in $\mathcal{D}_{tea}$, otherwise $a_t = a_t^{stu}$ and the transition is put in $\mathcal{D}_{stu}$. When CA is used, $r_t = r_t^{ext} + r_t^{int}$, otherwise $r_t = r_t^{ext}$. Once any of the pool has reached its predefined maximum size, adding a new transition results in deleting the oldest transition in the pool. During training, a pool is first selected from $\mathcal{D}_{tea}$ and $\mathcal{D}_{stu}$. The probability of selecting $\mathcal{D}_{tea}$ is p_{tea}, i.e. $\mathcal{D} \sim Ber(\mathcal{D}_{tea}, \mathcal{D}_{stu}; p_{tea})$ [3].Then a minibatch of transitions is uniformly sampled from the selected pool, i.e. $(\mathbf{b}_t, a_t, r_t, \mathbf{b}_{t+1}) \sim U(\mathcal{D})$. We call this *agent-aware* experience replay.

Except for the experience replay, a target network with weight vector θ^- is used. This target network is similar to the Q-network except that its weights are only copied every K steps from the Q-network, and remain fixed during all the other steps. The loss function for the Q-network at each iteration takes the following form:

$$L(\theta) = \mathrm{E}_{\mathcal{D}\sim Ber(\mathcal{D}_{tea},\mathcal{D}_{stu};p_{tea}),\ (\mathbf{b}_t,a_t,r_t,\mathbf{b}_{t+1})\sim U(\mathcal{D})} \left[\left(r_t + \gamma \max_{a_{t+1}} Q(\mathbf{b}_{t+1}, a_{t+1}; \theta^-) - Q(\mathbf{b}_t, a_t; \theta)\right)^2\right] \tag{1}$$

where $\gamma \in [0, 1]$ is the discount factor.

The probability p_{tea} controls how often the student learns from the teacher's experiences. As the learning goes on, the probability will decrease. More details will be described in the next section.

3.3 Companion Strategy

It's important for the student to estimate an appropriate point to end the reliance on the teacher. If the reliance is ended too early, the student itself may not reach an acceptable performance, resulting in the sharp drop of performance, which is the *safety* problem. However, if the student always relies on the teacher, it's hard to improve its performance to surpass the teacher's performance, which is the *efficiency* problem.

We get some inspirations from the studying process of a call center service agent. Consider how a new call center service agent gets started. At first, an experienced call center agent tells him some basic *rules* and the new agent works by often consulting these rules. His confidence about how to make decisions gradually increases during the continuous practice. Eventually, he is so confident about his own decisions that he no longer needs any consultation to these rules and even explores some better response ways through interaction with users which are not initially included in the rules. Similarly, we can use the uncertainty/certainty of the Q-network to determine the teaching time.

There are several methods to estimate the uncertainty/certainty in deep neural networks, e.g. Bayesian neural networks (Blundell et al., 2015), dropout (Gal and Ghahramani, 2016), bootstrap (Osband et al., 2016) . Here we use the dropout to estimate the certainty of Q-Network. We call this Q-network DropoutQNetwork. Dropout is a technique used to avoid over-fitting in neural networks. It was introduced several years ago by (Hinton et al., 2012) and studied more extensively in (Srivastava et al., 2014). When dropout is used in training, the elements of the output of each hidden layer $\mathbf{h}$ is randomly set to zero with probability p, i.e. $\mathbf{h}' = \mathbf{h} \odot \mathbf{z}$ [4] where $\mathbf{z}$ is binary vector and each element $z_i \sim Ber(1-p)$. $\mathbf{h}'$ is scaled by $\frac{1}{1-p}$ and then fed to the next layer. At test time the dropout is disabled, i.e. the output of each hidden layer $\mathbf{h}$ is directly fed to the next layer. Although dropout was suggested as an ad-hoc technique, recently it was theoretically proven that the dropout training in deep neural networks is an approximate Bayesian inference in deep Gaussian processes (Gal and Ghahramani, 2016). Therefore, a direct result of this theory gives us tools to model uncertainty with dropout neural networks. To obtain the uncertainty, similar with that at train phrase the dropout is enabled at test phrase. For each input instance (i.e. dialogue belief state) $\mathbf{b}_t$, performing N stochastic forward passes through the network and averaging the output $\mathbf{q}_i \triangleq [q_{i1}, \cdots, q_{iM}]$ to get the mean and the variance. Generally, the variance can be utilized to measure the uncertainty of output. However, it's not a normalized criteria, and it's hard to set a threshold below which we should be confident with the output.

Instead, we proposed a novel method to measure the certainty of the decision of student policy at t-th turn. For each stochastic forward passes, the action $a_{ti} = \arg\max_j q_{ij}$ is regarded as a vote. After N passes [5], there is a committee

[3] Ber is short for Bernoulli.

[4] Here $\odot$ is the element-wise product.

[5] The N forward passes can be done in parallel, e.g. the

$\{a_{t1}, \cdots, a_{tN}\}$ consisting of N votes. The action a_t^{stu} that should be taken in the belief state $\mathbf{b}_t$ is the one with the largest percentage of the votes, and the corresponding percentage is defined as certainty c_t. The process is described in Algorithm 1.

Algorithm 1 The Decision Procedure of Student Policy $\pi^{stu}(\mathbf{b}_t, N)$

```
Require:
    The repeat times $N$ and the belief state $\mathbf{b}_t$
Initial the probability vector $\mathbf{p} = [p_1, \cdots, p_M]$ with zero vector, where $M$ is the number of actions.
for $i = 1, N$ do
    $\mathbf{q}_i \leftarrow$ DropoutQNetwork($\mathbf{b}_t$)
    $a_{ti} \leftarrow \arg\max_j q_{ij}$
    $\mathbf{p}[a_{ti}] \leftarrow \mathbf{p}[a_{ti}] + 1/N$
end for
$c_t \leftarrow \max_j p_j$
$a_t^{stu} \leftarrow \arg\max_j p_j$
return $a_t^{stu}$, $c_t$
```

At the end of e-th dialogue, the average certainty of all turns is computed, i.e. $C_e = \frac{1}{T_e}\sum_{t=0}^{T_e} c_t$, where T_e is the number of turns in e-th dialogue. Generally, the variance of C_e between successive dialogues is high. In order to the smooth the estimation, here we use the moving average of C_e in previous W dialogues to represent the certainty of student at current dialogue, i.e.

$$\overline{C}_e = \frac{1}{W}\sum_{i=e-W}^{e-1} C_i. \qquad (2)$$

As the training goes on, $\overline{C}_e$ grows until it converges. If $\overline{C}_e$ in all successive W dialogues are greater then a threshold C_{th} as shown in Figure 2, it's assumed that the student reaches a point where it is confident enough with its own decision steadily. Therefore, the teaching, both EA and CA, should be ended from now on.

Before the end of the teaching, CA is done in all turns. However, if EA is always done, the disappearance of the teacher may cause a dramatic change in the hybrid decision policy, which results in a sharp drop of performance. To deal with this issue, a monotonically increasing function of the relative certainty $P_{tea}(\Delta C_e)$ is proposed to control the frequency of EA teaching. ΔC_e represents the distance between $\overline{C}_e$ and C_{th}, i.e. $\Delta C_e = \max(0, C_{th} - \overline{C}_e)$. The effect of $P_{tea}(\Delta C_e)$ is that the closer $\overline{C}_e$ is to C_{th}, the more unlikely EA teaching is executed. Besides controlling how often the student directly consult the teacher, another mission of $P_{tea}(\Delta C_e)$ is to control how often the teacher's experiences are replayed, i.e. the probability p_{tea} described in section 3.2. Implementation details of $P_{tea}(\Delta C_e)$ are described in Appendix C.

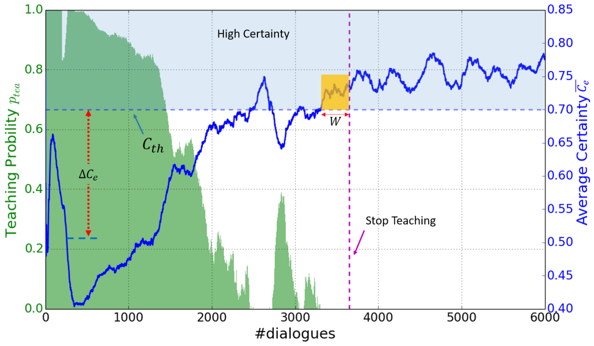

Figure 2: Illustration of average certainty $\overline{C}_e$ and the probability p_{tea}.

The full procedure of companion learning with logic rules is described in Algorithm 2.

3.4 Teacher Policy: Logic Rules

Rule-based policy is popular in commercial dialogue systems (Williams, 2008). The policy, i.e. the dialogue plan/flow, is designed by a domain expert. His knowledge of task domain and business rules is encoded in the rules. There are many methods to represent the decision rules, e.g. propositional logic, first-order logic, decision tree. Here, we use the ordered propositional logic rules, which can be easily translated into IF-THEN rules. When making the decision, these rules are executed in pre-defined order. If the conditions of any rule are satisfied, the decision process will be terminated and the output is the corresponding action. In this paper, three hand-crafted logic rules, R1, R2, and R3 , were used as the teacher:

- R1: confirm the most likely value in slots where the most likely value has probability between 0.1 and 0.6[6];
- R2: offer a restaurant if there is at least one slot in which the belief of most likely value is more than the belief of special value "*none*";

dialogue state can be repeated N times to form a mini-batch, then one forward is executed to get N outputs simultaneously.

[6] This threshold is the best one we have tried.

Algorithm 2 Companion Learning with Logic Rules

```
Require:
    The number of stochastic forward pass $N$, the maximal extra reward $\delta > 0$.
Initialize the parameters $\theta$ of student policy
Initialize replay pools $\mathcal{D}_{tea}$ and $\mathcal{D}_{stu}$ with $\{\}$, certainty memory $\mathcal{C}$ with $\{\}$, $teaching$ with $True$.
for $e = 1, E$ do
    Update the dialogue belief state $\mathbf{b}_0$
    Initialize the average certainty $C_e \leftarrow 0$
    if $teaching$ is $True$ then
        $teaching, p_{tea} \leftarrow \mathsf{Companion}(\mathcal{C})$
    end if
    for $t = 0, T_e$ do
        Set intrinsic reward $r_t^{int} \leftarrow 0$
        Get system action and the corresponding certainty, i.e. $a_t^{stu}, c_t \leftarrow \pi^{stu}(\mathbf{b}_t, N)$
        $C_e \leftarrow C_e + c_t$
        Get action from the rule-based policy, i.e. $a_t^{tea} \leftarrow \pi^{tea}(\mathbf{b}_t)$
        $EA \sim Ber(p_{tea})$
        if $teaching$ is $True$ and $EA$ is $True$ then
            $a_t \leftarrow a_t^{tea}$
        else
            $a_t \leftarrow a_t^{stu}$
        end if
        if $teaching$ is $True$ then
            $r_t^{int} \leftarrow (2 \times \mathbf{1}\{a_t = a_t^{tea}\} - 1)\delta$
        end if
        $C_e \leftarrow \frac{1}{T_e} C_e$, and store $C_e$ in $\mathcal{C}$
        Give the action $a_t$ to the environment, observe the extrinsic reward $r_t^{ext}$ and update the dialogue belief state $\mathbf{b}_{t+1}$
        $r_t \leftarrow r_t^{int} + r_t^{ext}$
        if $EA$ is $True$ then
            Store $\{\mathbf{b}_t, a_t, r_t, \mathbf{b}_{t+1}\}$ in $\mathcal{D}_{tea}$
        else
            Store $\{\mathbf{b}_t, a_t, r_t, \mathbf{b}_{t+1}\}$ in $\mathcal{D}_{stu}$
        end if
        Update the parameters $\theta$ of DropoutQNetwork according to the equation (1).
    end for
end for
return $\theta$
```

Algorithm 3 Companion Function Companion($\mathcal{C}$)

```
Require:
    The average certainty memory $\mathcal{C}$ at $e$-th dialogue and the moving window size $W$.
Initialize $teaching$ with $False$, $p_{tea}$ with 0
for $i = 0, W$ do
    Compute the moving average certainty $\overline{C}_{e-i}$ in ($e$-$i$)-th dialogue with equation (2).
    if $\overline{C}_{e-i} < C_{th}$ then
        $teaching \leftarrow True$
        break
    end if
end for
if $teaching$ is $True$ then
    $\Delta C_e \leftarrow \max(0, C_{th} - \overline{C}_e)$
    $p_{tea} \leftarrow P_{tea}(\Delta C_e)$
end if
return $teaching, p_{tea}$
```

- R3: request values for a slot which is uniformly selected from a pre-defined slot list.

The corresponding pseudo-codes are presented in Appendix B.

4 Evaluation Metrics of On-line Policy Optimization

Most previous work on the evaluation of RL-based dialogue policy optimization focuses on the final performance (FP) when the system converges to a steady level. However, for on-line policy optimization, it's important to measure the learning process. Except for FP, we proposed two quantitative metrics: safety loss and efficiency loss.

4.1 Safety Loss

In the on-line training process, unless the performance of the system reaches the acceptable performance S_a, the interaction between users and the system will be unsafe and causes trouble to continuing training. So the safety of the system is defined to be the system's ability to maintain performance above the acceptable performance S_a.

We quantify the safety loss of the system by summing up the performance gap between the acceptable performance and the system performance S_e in every episode during the on-line learning. Suppose there are E dialogues, then $L_1 = \sum_{e=1}^{E} \max(0, S_a - S_e)$. The safety loss has an intuitive interpretation as the area of the region below the threshold and above training curve. This metric is similar to the integral of absolute error (IAE) (Shinners, 1998) metric commonly adopted in the evaluation of control systems (Gaing, 2004; Jesus and Tenreiro MacHado, 2008).

4.2 Efficiency Loss

Another important issue of on-line learning is efficiency. The efficiency indicates the speed at which the system reaches a specific performance level. In reality, we can tolerate a system to make mistakes at the beginning but it should improve at a significant speed until reaching the ideal performance S_i. Therefore, later failures should weight more than early failures to evaluate efficiency. Similar to the integral of time multiplied by absolute error (ITAE) (Shinners, 1998) metric, we propose a metric efficiency loss. We multiply the performance gap between ideal performance and current performance with the episode index, thus giving later failure greater penalty. Specifically, $L_2 = \sum_{e=1}^{E} \max(0, S_i - S_e)e$.

More illustrations about safety loss and efficiency loss are given in Appendix D.

5 Experiments

Our experiments have three objectives: (1) Comparing our proposed *dropout* DQN in Algorithm 1 with some baselines when there is no teacher. (2) Comparing CL with other two baselines when the teacher gets involved, and investigating the benefits of our proposed *agent-aware* experience replay. (3) Visually analyzing the differences in behaviors between the rule-based teacher policy and the optimized student policy.

An agenda-based user simulator (Schatzmann et al., 2007a) with error model (Schatzmann et al., 2007b) was implemented to emulate the behavior of the human user, and a rule-based policy with 0.695 success rate described in section 3.2 was used as the teacher in our experiments. The purpose of the user's interacting with SDS is to find restaurant information in the Cambridge (UK) area (Henderson and Thomson, 2014). This domain has 7 slots of which 4 can be used by the system to constrain the database search. The summary action space consists of 16 summary actions. More details are described in Appendix A.

For reward, at each turn, an extrinsic reward of -0.05 is given to the student policy. At the end of the dialogue, a reward of $+1$ is given for dialogue success. The maximal extra reward δ is 0.05.

For each set-up, 10000 dialogues are used for training, the moving dialogue success rate is recorded with a window size of 1000. The final results are the average of 40 runs.

5.1 Policy Learning without Teaching

In this section, four policies without teaching are compared:

- DQN: A vanilla deep Q-Network (Mnih et al., 2015) which has two hidden layers, each with 128 nodes.
- A2C: An advantage actor-critic policy which consists of an actor network and a critic network (Fatemi et al., 2016).
- Dropout_DQN_1 and Dropout_DQN_32: They both have a dropout layer after each hidden layer. The dropout rate is 0.2. Their difference is that the number of stochastic forward pass N of Dropout_DQN_32 in Algorithm 1 is 32, while that of Dropout_DQN_1 is 1. Dropout_DQN_1 makes decision according to one output of Q-network similar to that of vanilla DQN. Dropout_DQN_1 was first proposed in (Gal and Ghahramani, 2016), and was confirmed that Dropout_DQN_1 can obtain more efficient exploration.

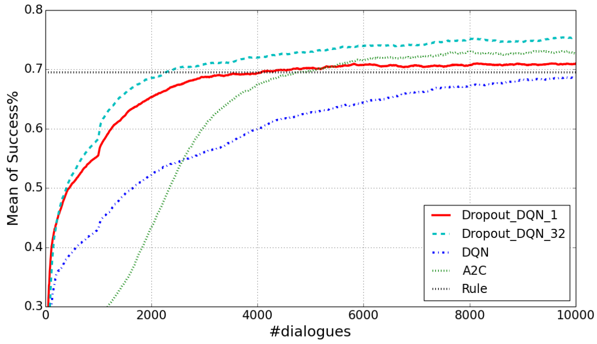

Figure 3: Comparison of four policies without teaching.

The learning curves are described in Figure 3 and the evaluation results are described in Table 1. Comparing Dropout_DQN_1 with DQN in figure 3, the improvement of efficiency caused by

Metrics	No Teaching				Teaching			
	DQN	Dropout_DQN_1	Dropout_DQN_32	A2C	A2C_PreTrain	EA	CL_D	CL_AAD
Safety	1043.3	321.4	258.0	1020.8	176.9	48.8	30.6	**18.6**
Efficiency($\times 10^4$)	534.7	249.1	75.6	299.0	205.4	58.0	62.6	**53.5**
FP	0.684	0.709	0.749	0.727	0.726	**0.751**	0.749	**0.751**

Table 1: The quantitative evaluation results of different methods. Here final performance (FP) is the success rate of last 2000 dialogues. The FP of `Dropout_DQN_32` 0.749 is used as the ideal performance S_i for computing efficiency loss, and the performance of the `rules` 0.695 is used as the acceptable performance S_a for computing safety loss.

dropout can be observed as claimed in (Gal and Ghahramani, 2016). However, `Dropout_DQN_1` seems to suffer premature and sub-optimal convergence, while our proposed `Dropout_DQN_32`, whose decision is based on multi votes (algorithm 1), can result in improvement of efficiency and better final performance. Moreover, `Dropout_DQN_32` also performs much better than the policy gradient method `A2C`.

For the following experiments, the times of stochastic forward pass N in Algorithm 1 is 32.

5.2 Policy Learning with Teaching

In this section, four methods of teaching by the rule-based policy are compared:

- `EA`: 500 dialogues are taught with EA at the beginning (Chen et al., 2017).
- `A2C_PreTrain`: At the beginning, 500 dialogue are collected with rule-based policy. These examples are used to pre-train the actor network with supervised learning. After the pre-training, the policy is continuously optimized with the A2C algorithm (Fatemi et al., 2016).
- `CL_AAD`: Full CL with AAD-DQN described in section 3.
- `CL_D`: CL without agent-aware experience repay, i.e. the teacher's experiences and student's experiences are put in one pool and are uniformly sampled for the experience replay in equation (1).

As can be seen in Figure 4, there is a big dip in the performance of `A2C_PreTrain`. One possible explanation is that because the rule-based policy is sub-optimal, the pre-training makes the student policy reach a local minimum point. The rl-training should first make it escape from the local minimum point, which results in a temporary loss in performance.

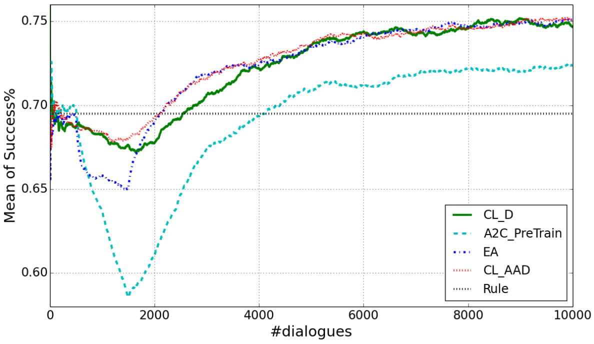

Figure 4: Comparison of four methods with teaching by rule-based teacher.

Comparing CL methods (`CL_D` and `CL_AAD`) with `EA` in Figure 4 and in Table 1, we can conclude that CL can significantly boost the safety of learning process. Moreover, except for safety, `CL_AAD` can boost the efficiency, which benefits from the *agent-aware* experience replay.

5.3 Comparison of Optimized Student Policy and Rule-based Teacher Policy

To interpret what the student has learnt, we further compare the rules and an optimized student policy with 76.7% success rate. The rule-based policy is used to collect 5000 dialogues, while in each turn the decision made by the student policy is also recorded. Figure 5 is a confusion matrix. The x-axis denotes the student's decision and the y-axis denotes the rules' decision. The numbers on the left are the statistics for each action in 5000 dialogues. Each element in the matrix denotes the normalized number of turns when the rule chooses the action in the corresponding line, the student chooses the action in the corresponding column.

As is shown in Figure 5, *offer* and *confirm* are two action types used most frequently. In more than half of turns when the rule-based pol-

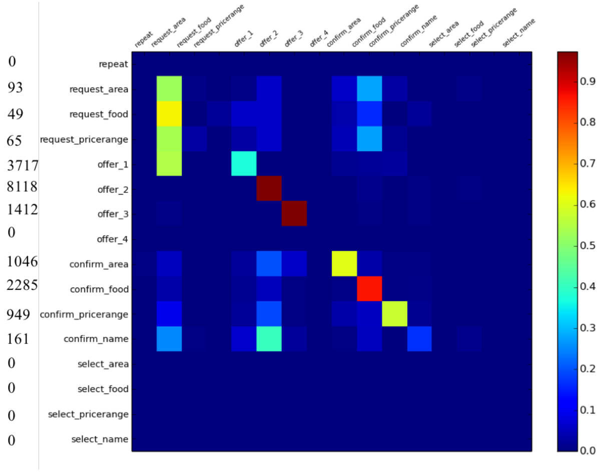

Figure 5: Confusion matrix between the decisions of rule-based policy and the decisions of the optimised student policy. The x-axis denotes the student's decision and the y-axis denotes the rules' decision.

icy chooses *offer_1*, the student policy will choose a different action. Furthermore, from the element in line *offer_1* and column *request_area*, we can find that in this situation the student policy prefers the action *request_area*. Inspired by this disagreement, we designed a new rule:

- R4: request values for slot area when there is only one other slot constraint for the database query.

Similarly, as can be seen in Figure 5, in a considerable proportion of turns when the rule-based policy chooses *confirm_area, confirm_pricerange*, or *confirm_name*, the student policy will choose the action *offer_2*, which may mean that for slots *area, pricerange*, or *name*, when there are values for database query, the system should offer a restaurant instead of confirming the slot-value constraints. Therefore, the rule R1 in section 3.2 was revised as follows:

- R1*: For slot food, confirm the most likely value has the probability between 0.1 and 0.6; For slot area, pricerange and name, confirm the most likely value, the belief of which is smaller than the belief of the special value "*none*" and is larger than 0.1.

Table 2 is the evaluation results of different ordered rules. The rule R4 can significantly boost the success rate (comparing line 2 with line 1), while the rule R1* can both boost the success rate and decrease the dialogue length (comparing line 3 with line 1). The combination of R4 and R1* takes respective advantages (comparing line 4 with line 1, line 2 and line 3). The performance of final order rules is comparable to the performance of optimized student policy.

Ordered Rules	Success Rate	#Turn	Reward
R1, R2, R3	0.695	4.58	0.4657
R1, R4, R2, R3	0.749	5.16	0.4910
R1*, R2, R3	0.705	4.44	0.4824
R1*, R4, R2, R3	**0.753**	4.98	**0.5042**

Table 2: Evaluation results of different ordered rules. As a reference, the performance of optimised student policy is success rate 0.767, #turn 5.10 and reward 0.5124.

It is worth noting that the primary rules R1, R2, and R3 in section 3.2 don't distinguish between different slots. However, the new rules R4 and R1* are all slot-specific, which it is difficult to design at the beginning.

6 Conclusion

This paper has proposed a *companion learning* framework to unify rule-based policy and RL-based policy. Here, the rule-based policy acts as a teacher, which either directly shows example action or gives an extra reward. Based on the uncertainty estimated using a dropout Q-Network, a companion strategy is proposed to control when the student policy directly consults rules and how often the student policy learns from the teacher's experiences. Simulation experiments showed that our proposed framework can significantly improve both *safety* and *efficiency* of on-line policy optimization. Additionally, we visually analyzed the differences in behaviors between the rule-based teacher policy and the optimized student policy, which gave us some inspirations to refine the rules.

Acknowledgments

This work was supported by the Shanghai Sailing Program No. 16YF1405300, the China NSFC projects (No. 61573241 and No. 61603252) and the Interdisciplinary Program (14JCZ03) of Shanghai Jiao Tong University in China. Experiments have been carried out on the PI supercomputer at Shanghai Jiao Tong University.

References

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622.

Lu Chen, Runzhe Yang, Cheng Chang, Zihao Ye, Xiang Zhou, and Kai Yu. 2017. On-line dialogue policy learning with companion teaching. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 198–204.

Heriberto Cuayáhuitl, Simon Keizer, and Oliver Lemon. 2015. Strategic dialogue management via deep reinforcement learning. *NIPS Deep Reinforcement Learning Workshop*.

Mehdi Fatemi, Layla El Asri, Hannes Schulz, Jing He, and Kaheer Suleman. 2016. Policy networks with two-stage training for dialogue systems. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 101–110.

Zwe-Lee L Gaing. 2004. A Particle Swarm Optimization Approach for Optimum Design of PID Controller in AVR System. *IEEE Transactions on Energy Conversion*, 19(2):384–391.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML-16)*.

Milica Gašić, Filip Jurčíček, Simon Keizer, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. 2010. *Gaussian processes for fast policy optimisation of POMDP-based dialogue managers*. Association for Computational Linguistics.

Milica Gašić, Filip Jurčíček, Blaise Thomson, Kai Yu, and Steve Young. 2011. On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 312–317. 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.

James Henderson, Oliver Lemon, and Kallirroi Georgila. 2008. Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Computational Linguistics*, 34(4):487–511.

Matthew Henderson and Blaise Thomson. 2014. The second dialog state tracking challenge. In *SIGDIAL*, volume 263, Stroudsburg, PA, USA. Association for Computational Linguistics.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Isabel S. Jesus and J. A. Tenreiro MacHado. 2008. Fractional control of heat diffusion systems. *Nonlinear Dynamics*, 54(3):263–282.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134.

Long-Ji Lin. 1993. *Reinforcement learning for robots using neural networks*. Ph.D. thesis, Fujitsu Laboratories Ltd.

Zachary C Lipton, Jianfeng Gao, Lihong Li, Xiujun Li, Faisal Ahmed, and Li Deng. 2016. Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking. *arXiv preprint arXiv:1608.05081*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. 2016. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems*, pages 4026–4034.

Olivier Pietquin, Matthieu Geist, and Senthilkumar Chandramohan. 2011. Sample Efficient On-line Learning of Optimal Dialogue Policies with Kalman Temporal Differences. In *IJCAI*, pages 1878–1883.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007a. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *NAACL*, pages 149–152, Morristown, NJ, USA. Association for Computational Linguistics.

Jost Schatzmann, Blaise Thomson, and Steve Young. 2007b. Error simulation for training statistical dialogue systems. In *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*, pages 526–531. IEEE.

Stanley M Shinners. 1998. *Modern control system theory and design*. John Wiley & Sons.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

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. In *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Continuously learning neural dialogue management. *arXiv preprint arXiv:1606.02689*.

Jason D Williams. 2008. The best of both worlds: unifying conventional dialog systems and pomdps. In *INTERSPEECH*, pages 1173–1176.

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. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

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

Jason D Williams and Geoffrey Zweig. 2016. End-to-end LSTM-based dialog control optimized with supervised and reinforcement learning. *CoRR*.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech and Language*, 24(2):150–174.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 1–10.