Clipping Loops for Sample-Efficient Dialogue Policy Optimisation

Yen-Chen Wu
University of Cambridge
ycw30@cam.ac.uk

Carl Edward Rasmussen
University of Cambridge
cer54@cam.ac.uk

Abstract

Training dialogue agents requires a large number of interactions with users: agents have no idea about which responses are bad among a lengthy dialogue. In this paper, we propose loop-clipping policy optimisation (LCPO) to eliminate useless responses. LCPO consists of two stages: loop clipping and advantage clipping. In loop clipping, we clip off useless responses (called loops) from dialogue history (called trajectories). The clipped trajectories are more succinct than the original ones, and the estimation of state-value is more accurate. Second, in advantage clipping, we estimate and clip the advantages of useless responses and normal ones separately. The clipped advantage distinguishes useless actions from others and reduces the probabilities of useless actions efficiently. In experiments on Cambridge Restaurant Dialogue System, LCPO uses only 260 training dialogues to achieve 80% success rate, while PPO baseline requires 2160 dialogues. Besides, LCPO receives $3.7/5$ scores in human evaluation where the agent interactively collects 100 real-user dialogues in the training phase.

1 Introduction

Based on dialogue policies, task-oriented dialogue systems decide when and how to give or request information from users. Learning dialogue policies is often formulated as a reinforcement learning (RL) problem since we usually receive feedback from users for the whole dialogue but not the correct answer for a single response (Young et al., 2013; Levin et al., 1997). With high-capacity of function approximation, deep reinforcement learning has been widely applied to dialogue policy optimisation (Su et al., 2016; Li et al., 2016; Casanueva et al., 2017). Typically, when applying deep reinforcement learning for dialogue policy management, more than thousands of dialogues are required to reach convergence (Casanueva et al., 2017). However, requiring thousands of human dialogues during training is quite impractical for most academic or real-life scenarios. Users might lose patience and exhibit different behaviour during training. Therefore, in most prior work, the agents are trained via simulated users instead of real ones (Liu et al., 2018; Gao et al., 2018).

Model-based reinforcement learning (MBRL) is commonly applied to make dialogue policy optimisation sample-efficient. MBRL approaches for dialogue management build a user model to predict users' behaviour (Wu et al., 2020b,a; Peng et al., 2018; Su et al., 2018; Wu et al., 2019; Zhang et al., 2019). Using the user model, DDQ (Peng et al., 2018) generates pseudo-data. The accuracy of the user model strongly affects the quality of generated pseudo-data. If the behaviour of pseudo-data is far from real users' behaviour, dialogue policies learnt from these data might not be optimal (Su et al., 2018). Manipulating when to use how much data in experience buffers becomes critical in these approaches.

Trainable-action-mask (TAM) (Wu et al., 2020b) blocks useless actions by learning action-masks from data to explore the action space more efficiently. Instead of predicting the users' behaviour directly, TAM predicts only the termination and similarity of future dialogue states to ease the training difficulties. However, the wrong predictions of the user model block the wrong actions, which makes the policy performance unstable. Moreover, the wrong output of policy does not learn from the predictions of the user model since it is blocked. Wrong values in policy networks make the performance unstable.

In this work, we propose loop-clipping policy optimisation (LCPO), which clips off useless actions in trajectories, computes advantages of actions in/out of the loop separately and optimises policy based on proximal policy optimisation (PPO) (Schulman et al., 2017). First, LCPO is a

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3420–3428
June 6–11, 2021.

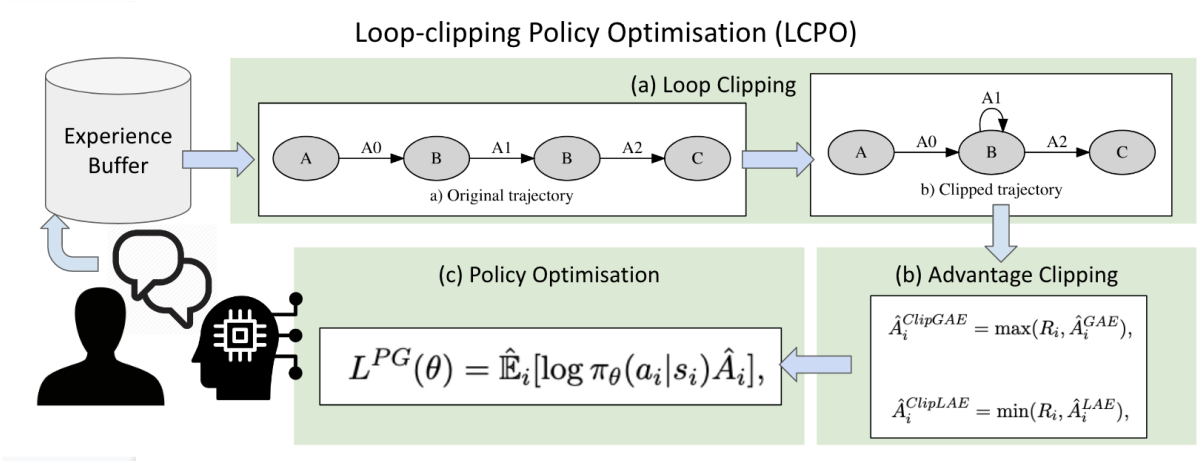

Figure 1: Illustration of LCPO. (a) Loop clipping. Clip off loops in trajectories to make them succinct. (b) Advantage clipping. Set a threshold such that the advantages of loops are always less than other useful actions. (c) Policy optimisation. We adopt proximal policy optimisation (PPO) in this paper.

model-free and parameter-free algorithm. There is no additional effort of tuning hyperparameters of the user model. Also, it takes almost no extra running time during testing. Second, instead of brutally blocking actions like TAM does, LCPO directly reduces the probabilities of useless actions which makes optimisation smoother and easier. In our experiment on the Cambridge Restaurant Dialogue System, LCPO uses only 260 dialogues in the training phase to reach an 80% success rate, while the PPO baseline requires 2160. In the human-in-the-loop experiment, LCPO that trained with only 100 dialogue receives 3.7/5 scores and high remarks of conciseness and fluency. Overall, our main contributions are two-fold:

- We propose LCPO, a parameter-free, sample efficient algorithm to optimise dialogue policies. This algorithm is easy to implement and has barely any overhead.
- We demonstrate that training dialogue systems with real users is feasible within 100 dialogues on Cambridge Restaurant Dialogue System.

2 Preliminaries

This section goes through the notations in this paper. We start with formulating dialogue management as an RL problem in section 2.1. In section 2.2, we explain how to optimise the policy through proximal policy optimisation (PPO). In section 2.4, we explain what is episodic memory.

2.1 Reinforcement learning for dialogue systems

When applying reinforcement learning for dialogue management (Levin et al., 1997; Young et al., 2013; Williams, 2008), a state s, or a belief state, is the belief distribution over users' requests. An action a is the summarised action taken by a system. A reward r and a termination t are given by simulated users or real users. An episode E is a dialogue. The goal of reinforcement learning is to learn a policy $\pi(a_i|s_i)$ that maximises the cumulative reward $R = \sum_{i=0}^{L} \gamma^i r_i$, where L is the length of the dialogue.

2.2 Proximal Policy Optimisation (PPO)

Policy gradient is a fundamental optimisation algorithm with the loss function:

$$L^{PG}(\theta) = \hat{\mathbb{E}}_i[\log \pi_\theta(a_i|s_i)\hat{A}_i], \quad (1)$$

where $\hat{A}_i$ is the estimated advantage at timestep i.

In order to ensure new policy is not changing far from the old one, trust region policy optimisation (TRPO) is set to surrogate the KL-divergence between the old and the current policies. In a similar but much simpler way, proximal policy optimisation (PPO) (Schulman et al., 2017) clips probability ratios r_i to mitigate the excessive updates in TRPO.

$$r_i = \frac{\pi_\theta(a_i|s_i)}{\pi_{\theta_{old}}(a_i|s_i)}. \quad (2)$$

$$L^{PPO}(\theta) = \hat{\mathbb{E}}[\min(r_i(\theta)\hat{A}_i, r_i^{clip}\hat{A}_i)], \quad (3)$$

where

$$r_i^{clip} = clip(r_i(\theta), 1-\epsilon, 1+\epsilon) \quad (4)$$

Advantage $\hat{A}_i$ and state-value $\hat{V}_i$ are estimated by generalised advantage estimation (GAE) as follow:

$$\hat{A}_i = \delta_i + \gamma\lambda\hat{A}_{i+1}, \quad (5)$$

$$\hat{V}_i = \hat{A}_i + V_i, \quad (6)$$

where γ decays future state-value, which represents our confidence in state-value estimation. λ decays the future TD-error, which represents a trade-off between bias and variance of advantage estimation. V_i is the predicted state-value of s_i, and δ_i is the TD-error:

$$\delta_i = r_i + \gamma V_{i+1} - V_i. \quad (7)$$

2.3 Trainable-action-mask (TAM)

Trainable-action-mask (TAM) (Wu et al., 2020b) is a model-based baseline that blocks useless actions directly. TAM learns a user model during dialogue interaction. The user model predicts the termination, reward, and the similarity between the current and the next dialogue state, and the action mask is constructed based on these features.

Though TAM is simple and effective, it is not stable enough. The first reason is a common pitfall of model-based approaches: the user model is hard to train and usually leads to inaccurate predictions that harm the dialogue policy. Second, the policy and state-value approximator (i.e. the policy network and value network in PPO) do not learn from the predictions of the user model. The wrong values estimated by these networks can not be updated efficiently since these actions are blocked.

2.4 Episodic memory

In most policy gradient algorithms, the history of interactions is recorded in a memory buffer M, which contains several episodes E. An episode E consists of N transitions $\{T_0, T_1, ..T_N\}$. A transition $T_i \equiv \{s_i, a_i, s_{i+1}, r_i, t_i\}$, where s_i is the current state. a_i is the action taken on s_i, which leads to the next state s_{i+1} with a reward r_i. If the episode terminates after taking action a_i, t_i is $True$ or otherwise $False$.

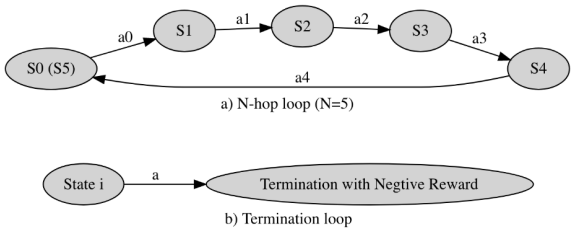

Figure 2: Illustration of two kinds of loops. In this paper, loop means transitions consist of useless or unwanted actions. (a) N-hop loop L^N, where $N = 5$ in this example. (b) Termination loop L^T.

3 Loop-clipping Policy Optimisation (LCPO)

In this paper, we propose loop-clipping policy optimisation (LCPO) to improve sample efficiency. As illustrated in Figure 1, LCPO consists of three components: loop clipping, advantage clipping, and policy optimisation. We adopt proximal policy optimisation (PPO) (Schulman et al., 2017) for the policy optimisation part in this work.

Firstly, we give definitions to loops in section 3.1, and illustrate how to get clean trajectories via loop clipping in section 3.2. In section 3.3, we demonstrate how to estimate and clip advantages and state-values of loops for policy optimisation. Note that in the following subsections, we utilise two domain knowledge in dialogue systems.

- **Prior 1**: Information gain is non-negative since by asking more questions, we know better about user needs.
- **Prior 2**: The last action of a failed dialogue and actions that loop over the same state are unwanted.

3.1 Definition of loop

In this paper, loop means transitions that consist of useless or unwanted actions. As illustrated in Figure 2, we define two kinds of the loop: N-hop loop and termination loop corresponding to our prior 2.

Definition 1. *A N-hop loop L_i^N is a sequence of N transitions $\{T_i, T_{i+1}, ..T_{i+N-1}\}$ where $s_i = s_{i+N}$.*

Since in a loop that the starting state s_i becomes the same as final state s_N, $\{a_i, a_{i+1}, .., a_{i+N-1}\}$ is a useless action sequence on state s_i. In dialogue systems, N-hop loop might result from repetitively asking the same questions or giving the same information. Compared with the definition of useless

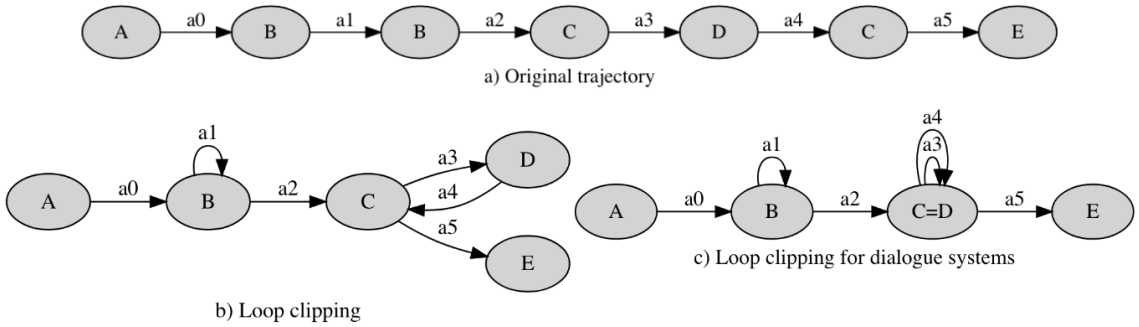

Figure 3: Loop clipping. The shaded circles are states and the arrows are actions taken between states. (a) The original trajectory. (b) The clean trajectory after loop clipping. In this example, a 1-hop loop and a 2-hop loop are detected. (c) In dialogue systems, the states in a loop are the same since there is no information gain.

actions in TAM (Wu et al., 2020b) which only considers the similarity of the next state (i.e 1-hop loop), N-hop loop is a more general definition and is able to detect more useless actions.

Definition 2. *A termination loop L_i^T is a transition T_i state s_i where $t_i = True$ and $r_i \leq 0$.*

In dialogue systems, a_i is a useless action on state s_i since the dialogue is terminated and failed. For example, termination loops might result from saying goodbye before completing tasks or making users out of patience. Note that the definition of loops utilises domain knowledge and might not be suitable for other applications.

3.2 Loop clipping

As illustrated in Figure 3, the original trajectory might contain several identical states. We search for the identical states pair-wisely and detect loops by definitions in section 3.1. [1] The detected loops are clipped off from the original trajectory. After clipping, the trajectory becomes succinct so that reward signals can be assigned to useful actions effectively (Figure 3b).

In dialogue systems, the information for each state in a loop is the same since there is no information gain after taking useless actions. Therefore, a N-hop loop can be viewed as multiple one-hop loops as illustrated in Figure 3c.

3.3 Loop advantage estimation (LAE)

After loop clipping, the original trajectory is split into a clean trajectory and several loops. Then We estimate the advantages of the clean trajectory and loops separately. For the clean trajectory, standard generalised advantage estimation (GAE) (Schulman et al., 2015) is applied as shown in Eq. 5, 6. If we only update the policy based on clean trajectories, the clipped useless actions will not be treated as training data. Therefore, these useless actions will not be penalised, resulting in unwanted lengthy dialogue. We first illustrate how to estimate state-values and advantage for loops, noted as loop advantage estimation (LAE) to distinguish from GAE. Second, we propose an advantage clipping trick, which makes the policy optimisation much more sample-efficient.

State-value estimation According to prior 1, in dialogue systems, information gain $V_{i+1} - V_i \geq 0$.

In a loop L_i^N with length N, since the

$$\hat{V}_i \leq \hat{V}_{i+1} \leq .. \leq \hat{V}_{i+N} \tag{8}$$

and the same states share the same value i.e.

$$\hat{V}_i = \hat{V}_{i+N}, \tag{9}$$

all the state-values in L_i^N are the same:

$$\hat{V}_i = \hat{V}_{i+1} = .. = \hat{V}_{i+N}. \tag{10}$$

Advantage estimation The loop advantage for action a_i is:

$$\hat{A}_i^{LAE} = \delta_i + \gamma\lambda\hat{A}^{GAE}, \tag{11}$$

where $\delta_i = r_i + \gamma V_{i+1} - V_i$. Note that $\hat{A}^{GAE}$ is the next advantage $\hat{A}_{i+N}$ after the loop L_i^n. No matter how long the loop is, the loop advantage is computed from the transition after loop.

When state-values converge, $V_{i+1} \simeq V_i$ in loop

[1] When the loop structure is complex. The solution of N-hop loop clipping is not unique.

L_i by eq. 10. We can see that

$$\begin{aligned} V_{i+1} \simeq V_i &\implies \delta_i \simeq r_i + (\gamma - 1)V_i \doteq R_i \\ &\implies \hat{A}_i^{LAE} \simeq R_i + \gamma\lambda\hat{A}^{GAE} \\ &\implies \hat{A}_i^{LAE} \simeq \hat{A}_{i+1}^{LAE}, \end{aligned} \tag{12}$$

where $R_i = r_i + (\gamma - 1)V_i$. It is straightforward that when values converge, the advantage of loop is the advantage of best actions A_i^{GAE} with a one-turn penalty for all useless actions on state s_i (since the agent wastes one more turn on the same state). When $\hat{A}^{GAE}$ converges to zero, $\hat{A}^{LAE}$ converges to R_i.

Advantage clipping However, we found that the advantage estimation is still not very accurate in the early stage of training process. The advantages of looping actions sometimes are higher than others and these actions are not penalised.

To properly penalise the looping actions, we clip the advantages in both LAE and GAE. The clipping threshold $R_i = r + (\gamma - 1)V_i$ since $\hat{A}^{LAE}$ converges to this value.

$$\hat{A}_i^{ClipGAE} = \max(R_i, \hat{A}_i^{GAE}), \tag{13}$$

$$\hat{A}_i^{ClipLAE} = \min(R_i, \hat{A}_i^{LAE}), \tag{14}$$

where $R_i = r_i + (\gamma - 1)V_i$, so that

$$\hat{A}_i^{ClipGAE} \geq R_i \geq \hat{A}_i^{ClipLAE}. \tag{15}$$

This trick distinguishes bad responses from good ones explicitly and makes policy converge faster.

The instruction of LCPO implementation is in Algorithm 1.

Algorithm 1: LCPO Algorithm

```
Collect N transitions into Memory M
for Episode E in M do
    // Loop clipping
    for T_i in E do
        if i<ptr then
            continue
        for T_i in E do
            if s_i == s_j then
                ptr = j
        if i<ptr then
            T_i ∈ 𝕃
    // Advantage estimation in
       reversed order
    for Transition T_i in reversed(E) do
        if T_i ∈ 𝕃 then
            Estimate Â, V̂ via clipped LAE
             (Eq. 14, 10)
        else
            Estimate Â, V̂ via clipped GAE
             (Eq. 13, 6)
Optimise the policy π via PPO (Eq. 3), with
 K epochs and mini-batch size B
```

4 Experiments

Experiments are conducted on the Cambridge restaurant dialogue system using the PyDial toolkit (Ultes et al., 2017). We evaluate the agents on both a simulated user and real users. From section 4.1 to 4.5, we illustrate the experiments with a simulated user. For human-in-the-loop experiment, see section 4.6.

4.1 Settings

User simulator We use a goal-driven simulated user on the semantic level (Schatzmann et al., 2007; Schatzmann and Young, 2009). The maximum dialogue length is set to 25 turns and $\gamma = 0.99$. The reward is defined as 20 for a successful dialogue minus the number of turns in the dialogue. 15% semantic error rate (SER) is included in the user simulator to accommodate for automatic speech recognition (ASR) error.

Policy optimisation Proximal policy optimisation (PPO) is applied. The state and action dimension of policy and value networks are 268 and 16. Dimensions of two hidden layers are 130 and 50. The agent collects a $N = 100$ transitions to update the policy π with $K = 10$ epochs and mini-batch size $B = 16$. After an update, the memory is flushed and becomes empty again. Optimiser is ADAM (Kingma and Ba, 2014) with a learning rate 0.001. Entropy coefficient is 0.01 and standardised advantages is applied. During testing, actions are sampled from the output distributions of the policy network.

Loops Detection In theory, the starting state and the ending state are identical in a loop. Yet, due to numerical uncertainty, we use cosine similarity with threshold $\eta = 0.99$ to justify whether two states are the same. Under this strict setting, two states are considered different if they have any dif-

Figure 4: Learning curves of different algorithms. *Left:* Success rate. *Right:* Average number of turns. The results are evaluated by 10 runs. The lines are averages and the shades represent standard deviations.

	Low resource experiment@200D				Final performance@2000D			
Algorithm	Suc.	Turns	#Loops	#Dialog	Suc.	Turns	#Loops	Time
PPO	$37.8 \pm 17.2\%$	**7.4** ± 2.3	**3.5**	2160	$95.16 \pm 1.1\%$	7.8 ± 0.4	2.3	**58 min**
TAM	$36.2 \pm 6.8\%$	8.6 ± 0.9	4.1	1140	$93.58 \pm 1.5\%$	7.4 ± 0.5	2.5	125 min
LCPO	**76.0** $\pm 2.9\%$	11.5 ± 0.8	4.7	**260**	**95.7** $\pm 1.1\%$	**6.3** ± 0.3	**0.9**	68 min

Table 1: Baseline comparison. The highest performance in each column is highlighted. The success rate, number of turns, and number of loops are reported for 200 and 2000 training dialogues. The number of training dialogue required to reach 80% success rate and the training time usage are also listed in the table. #Dialogue means the average required number of dialogues to reach 80% success rate. Time means the average training time for 2000 training dialogues.

ferent slot-values. In practice the optimal threshold depends on the noise level of state observation.

Evaluation In the experiment with the simulated user, we evaluate each agent with 500 dialogues after every 100 training dialogues. The mean and standard deviation of performance is computed over 10 runs with different neural networks initialisation. The mean $\pm$ standard deviation is depicted as the shaded area.

The x-axes of figures are in log-scale to emphasise both the early stage and the final performance of the training process.

4.2 Baseline Comparisons

In figure 4, we compare the performance of PPO (Schulman et al., 2017), TAM (Wu et al., 2020b), and LCPO. The left part of the figure shows the learning curves of the success rate. We can see that LCPO is considerably stable and sample-efficient. Worth to note that LCPO has the best final performance. TAM learns slower, and PPO requires a large number of training dialogues.

The right part of the figure shows the average turns taken by the agent. The lower, the better. We can see that LCPO takes more turns in the beginning but becomes more concise than the baselines later.

In table 1, we can see the detail of performance at 200 and 2000 training dialogues respectively. In low resource scenario, where the dialogue policy is trained by 200 dialogues, LCPO outperforms other baselines with small variance. Yet the average number of loops in each dialogue is higher. That is because LCPO takes more turn than other agents. Other agents often give poor responses so that the users leave the dialogue out of patience with fewer turns.

Regarding final performance at 2000 dialogues, all of the agents perform similarly. We can note that LCPO takes the least number of turns since its algorithm prevents from doing useless actions. LCPO requires only 260 dialogues to reach 80% success rate while PPO takes 2160. In addition, LCPO is light-packed and does not consume a lot of additional training time like TAM.

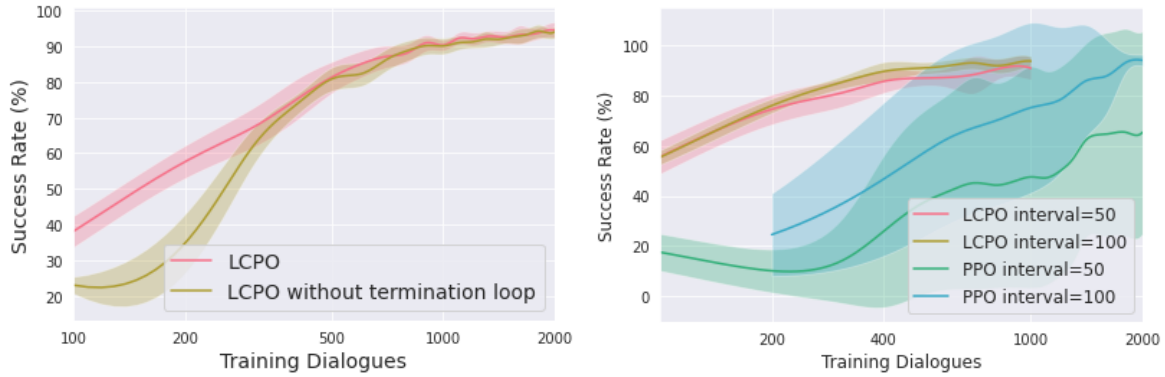

Figure 5: *Left:* Ablation study of termination loop. *Right:* Robustness to hyper-parameters.

4.3 Ablation study: termination loop

In the left part of figure 5, the red and brown lines are LCPO with and without clipping termination loop L^T respectively. We can see that without clipping L^T, the learning curves become less stable and inhibit the cold-start problem at the beginning of training.

In a failed dialogue, some actions are good and should not be penalised for the failure of conversation. Therefore, we should clip off the last transition in failed dialogue, so that the rest transitions in the clean trajectories (not in loops) are not penalised for the failure. For example, if we clip off the last the action "bye" in a failed dialogue, only 'bye' is strongly penalised while other normal interactions are not.

4.4 Ablation study: advantage clipping

We propose 4 agents for comparisons: 1) clip both GAE and LAE, 2) clip LAE, 3) clip GAE, 4) no advantage clipping. The success rates after training with 100, 200, 2000 dialogues are reported in Table 2.

In the low-resource scenario (less than 200 dialogues), clipping both GAE and LAE outperforms other methods considerably. And LCPO with no advantage clipping is the worst. Without clipping, inaccurate advantage estimation in the early stage of the training process cannot reduce the probabilities of useless actions efficiently.

Regarding the final performance after training agents with 2000 dialogues, all of the methods perform similarly. Yet, if we only clip the GAE, the final performance is slightly worse than others. That is because not all the actions in clean trajectories are useful. The 'clean' trajectories still contain several useless actions though not detected.

Methods	Success Rate @100	@200	@2000
Clip GAE+LAE	**55.4**	**76.0**	93.7
Clip LAE	50.7	73.5	**93.9**
Clip GAE	51.1	72.6	91.0
No adv clip	45.7	61.8	93.5

Table 2: Comparison of different advantage clipping methods. Success rates are reported after training agents with 100, 200, 2000 dialogues. The highest success rate in each column is highlighted.

Assigning larger advantages to all actions in clean trajectories makes performance unstable.

4.5 Robustness to hyperparameters

In the right part of figure 5, policy update interval is set to 50 and 100 for PPO and LCPO. The red and brown lines are LCPO and the green and blue lines are PPO with different update intervals. We can see that the performance of PPO is strongly affected by the update interval. In contrast, LCPO still shows high stability and sample efficiency. Its robustness to hyperparameters makes tuning LCPO effortless.

4.6 Human-in-the-loop Evaluation

General Settings The dialogue system uses a rule-based belief tracker, and an NLG model (Wen et al., 2015). In each dialogue, one of the agents is randomly picked to talk with a user. The users have to interact with the agent according to a given instruction on the user goal sampled from the corpus. The users can decide to leave the dialogue session if they are out of patience.

Training Settings We experiment on two training algorithms: PPO and LCPO. The hyperparameters of PPO and LCPO are the same as the simulated user experiment. A human user interacts with each agent for 100 dialogues. At the end of each dialogue, the user gives 20 scores to the agent for a successful dialogue and gives 0 scores for a failed one. A penalty of -1 is also applied in each turn.

A successful dialogue means the restaurant given by the agent must fulfil all the constraints and the requested information like phone number or address must be provided. In other words, the agents only receive feedback on the aspect of task completion.

Evaluation Settings Each human user interacts with each agent for 5 dialogue and gives his/her feedback on four aspects:

- **Task completion**: The agent finds a restaurant that meets the constrains. The requested information is also given.
- **Conciseness**: The agent is to the point and does not ask/provide the same information repetitively.
- **Fluency**: The agent does not interrupt the dialogue flow and answer the questions logically.
- **Overall score**: The overall score for chatting with this agent.

Each agent is evaluated on 100 dialogues, the mean and variance of each score are reported in Table 3. The scores are range from 0 to 5. We also evaluate the agents by a simulated user via 500 dialogues for each agent.

Results In table 3, we can see that LCPO significantly outperform PPO in all aspects. The task completion is close to the success rate evaluated by the simulated user. Conciseness is the feature of this work, and the improvement is also the most considerable. Regarding fluency, the difference between PPO and LCPO is smaller. Sometimes a fluent conversation takes more turns. Sometimes a non-logical response can complete the task as well (e.g. inform a restaurant name in the beginning). However, LCPO is still better than PPO in terms of fluency since a non-logical response usually accompanies with no information gain.

	PPO	LCPO
Task Completion	2.0 ± 1.7	**3.2** ± 1.5
Conciseness	1.8 ± 0.8	**3.9** ± 1.1
Fluency	2.6 ± 0.5	**3.6** ± 0.9
Overall score	2.1 ± 0.4	**3.7** ± 0.9
Success rate (SimUser)	41.7%	**66.8**%

Table 3: Human-in-the-loop experiment. Human users evaluate each agent in four aspects. Each agent is trained by interacting with a human for 100 dialogues. The highest success rate in each row is highlighted. The last row is the success rates over 500 dialogues evaluated by a simulated user.

5 Conclusion

Our contributions are:

- We propose LCPO to improve sample efficiency for dialogue policy optimisation. LCPO has two critical components: loop clipping and advantage clipping. Both of them are strongly effective in low resource scenario and easy to implement. LCPO also demonstrates strong robustness to hyperparameters.
- We train and evaluate dialogue agents with real users on the Cambridge Restaurant domain. We also demonstrate that human-in-the-loop training is feasible within 100 dialogues. The evaluation has four aspects to clarify what has been learnt by each agent. LCPO outperforms PPO in all aspects.

In this paper, LCPO integrates with PPO. In the future, we will generalise loop clipping method to other off-policy reinforcement learning approaches with episodic memory since off-policy approaches are considered more sample-efficient.

References

Iñigo Casanueva, Paweł Budzianowski, Pei-Hao Su, Nikola Mrkšić, Tsung-Hsien Wen, Stefan Ultes, Lina Rojas-Barahona, Steve Young, and Milica Gašić. 2017. A benchmarking environment for reinforcement learning based task oriented dialogue management. *arXiv preprint arXiv:1711.11023*.

Jianfeng Gao, Michel Galley, and Lihong Li. 2018. Neural approaches to conversational ai. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1371–1374.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 1997. Learning dialogue strategies within the markov decision process framework. In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pages 72–79. IEEE.

Jiwei Li, Alexander H Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2016. Dialogue learning with human-in-the-loop. *arXiv preprint arXiv:1611.09823*.

Bing Liu, Gokhan Tur, Dilek Hakkani-Tur, Pararth Shah, and Larry Heck. 2018. Dialogue learning with human teaching and feedback in end-to-end trainable task-oriented dialogue systems. *arXiv preprint arXiv:1804.06512*.

Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Kam-Fai Wong. 2018. Integrating planning for task-completion dialogue policy learning. *arXiv preprint arXiv:1801.06176*.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 149–152. Association for Computational Linguistics.

Jost Schatzmann and Steve Young. 2009. The hidden agenda user simulation model. *IEEE transactions on audio, speech, and language processing*, 17(4):733–747.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

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

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. *arXiv preprint arXiv:1605.07669*.

Shang-Yu Su, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Yun-Nung Chen. 2018. Discriminative deep dyna-q: Robust planning for dialogue policy learning. *arXiv preprint arXiv:1808.09442*.

Stefan Ultes, Lina M Rojas Barahona, Pei-Hao Su, David Vandyke, Dongho Kim, Inigo Casanueva, Paweł Budzianowski, Nikola Mrkšić, Tsung-Hsien Wen, Milica Gasic, et al. 2017. Pydial: A multi-domain statistical dialogue system toolkit. *Proceedings of ACL 2017, System Demonstrations*, pages 73–78.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Jason D Williams. 2008. The best of both worlds: Unifying conventional dialog systems and pomdps. In *Ninth Annual Conference of the International Speech Communication Association*.

Yen-chen Wu, Bo-Hsiang Tseng, and Milica Gasic. 2020a. Actor-double-critic: Incorporating model-based critic for task-oriented dialogue systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 854–863.

Yen-Chen Wu, Bo-Hsiang Tseng, and Carl Edward Rasmussen. 2020b. Improving sample-efficiency in reinforcement learning for dialogue systems by using trainable-action-mask. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8024–8028. IEEE.

Yuexin Wu, Xiujun Li, Jingjing Liu, Jianfeng Gao, and Yiming Yang. 2019. Switch-based active deep dyna-q: Efficient adaptive planning for task-completion dialogue policy learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7289–7296.

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.

Zhirui Zhang, Xiujun Li, Jianfeng Gao, and Enhong Chen. 2019. Budgeted policy learning for task-oriented dialogue systems. *arXiv preprint arXiv:1906.00499*.