# Post-processing Networks: Method for Optimizing Pipeline Task-oriented Dialogue Systems using Reinforcement Learning

**Atsumoto Ohashi** **Ryuichiro Higashinaka**
Graduate School of Informatics, Nagoya University
ohashi.atsumoto.c0@s.mail.nagoya-u.ac.jp
higashinaka@i.nagoya-u.ac.jp

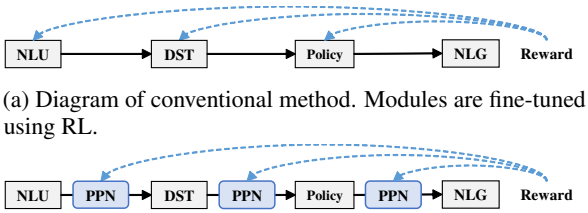
## Abstract

Many studies have proposed methods for optimizing the dialogue performance of an entire pipeline task-oriented dialogue system by jointly training modules in the system using reinforcement learning. However, these methods are limited in that they can only be applied to modules implemented using trainable neural-based methods. To solve this problem, we propose a method for optimizing a pipeline system composed of modules implemented with arbitrary methods for dialogue performance. With our method, neural-based components called post-processing networks (PPNs) are installed inside such a system to post-process the output of each module. All PPNs are updated to improve the overall dialogue performance of the system by using reinforcement learning, not necessitating each module to be differentiable. Through dialogue simulation and human evaluation on the MultiWOZ dataset, we show that our method can improve the dialogue performance of pipeline systems consisting of various modules[1].

## 1 Introduction

Task-oriented dialogue systems can be classified into two categories: pipeline systems, in which multiple modules take on a sequential structure, and neural-based end-to-end systems (Chen et al., 2017; Gao et al., 2018; Zhang et al., 2020b).

A typical pipeline system consists of four modules (Zhang et al., 2020b): natural language understanding (NLU), dialogue state tracking (DST), Policy, and natural language generation (NLG). Each module can be implemented individually using various methods (e.g., rule-based and neural-based) (Ultes et al., 2017; Zhu et al., 2020). In a pipeline system, the inputs and outputs of each module are explicit, making it easy for humans to interpret.




(a) Diagram of conventional method. Modules are fine-tuned using RL.

(b) Diagram of proposed method. Each PPN that post-processes output of each module is optimized using RL.

Figure 1: Comparison of conventional and proposed methods

However, since each module is processed sequentially, errors in the preceding module can easily propagate to the following ones, and the performance of the entire system cannot be optimized (Tseng et al., 2021). This results in low dialogue performance of the entire system (Takanobu et al., 2020).

In contrast, neural-based methods can optimize entire neural-based end-to-end systems, which allows for less error propagation than pipeline systems and high dialogue performance (Dinan et al., 2019; Gunasekara et al., 2020). The drawback of these methods is the large amount of annotation data required to train systems (Zhao and Eskenazi, 2016). Compared with pipeline systems, neural-based end-to-end systems are also less interpretable and more difficult to adjust or add functions.

To marry the benefits of both pipeline and end-to-end systems, methods (Liu et al., 2018; Mehri et al., 2019; Lee et al., 2021; Lin et al., 2021) have been proposed for optimizing an entire pipeline system in an end-to-end fashion by using reinforcement learning (RL) (Figure 1(a)). These methods are powerful because they jointly train and fine-tune neural-based implementations of the modules, such as NLU, Policy, and NLG, by using RL. However, these methods may not always be applicable because there may be situations in which modules can only be implemented with rules or the modules' internals cannot be accessed, such as with a Web

[1]Our code is publicly available at https://github.com/nu-dialogue/post-processing-networks



*Proceedings of the SIGdial 2022 Conference*, pages 1–13
Heriot-Watt University, Edinburgh, UK. 07-09, September, 2022 

API.

With this background, we propose a method for optimizing an entire pipeline system composed of modules implemented in arbitrary methods. We specifically focus on modules that output fixed sets of classes (i.e., NLU, DST, and Policy) and install neural-based components (post-processing networks; PPNs) in the system to post-process the outputs of these modules, as shown in Figure 1(b). Each PPN modifies the output of each module by adding or removing information as necessary to facilitate connections to subsequent modules, resulting in a better flow of the entire pipeline. To enable the appropriate post-processing for the entire system, each PPN uses the states of all modules in the system when executing post-processing. The post-processing of each PPN is optimized using RL so that the system can improve its dialogue performance, e.g., task success. A major advantage of our method is that each module does not need to be trainable since PPNs are trained instead.

To evaluate the effectiveness of our method, we applied PPNs to pipeline systems consisting of modules implemented with various methods (e.g., rule-based and neural-based) on the basis of the MultiWOZ dataset (Budzianowski et al., 2018) and conducted experiments by using dialogue simulation and human participants. The contributions of this study are as follows.

- We propose a method of improving the dialogue performance of a pipeline task-oriented dialogue system by post-processing outputs of modules. Focusing on NLU, DST, and Policy, our method can be applied to various pipeline systems because PPNs do not depend on the implementation method of each module or a combination of modules.

- Dialogue simulation experiments have shown that our method can improve the dialogue performance of pipeline systems consisting of various combinations of modules. Additional analysis and human evaluation experiments also verified the effectiveness of the proposed method.

## 2 Related Work

Our study is related to optimizing an entire dialogue system with a modular architecture. Wen et al. (2017) proposed a method for implementing all the functions of NLU, DST, Policy, and NLG modules by using neural networks, enabling the entire system to be trained. Lei et al. (2018) incorporated both a decoder for generating belief states (i.e., DST module) and a response-generation decoder (i.e., NLG module) into a sequence-to-sequence model (Sutskever et al., 2014). Zhang et al. (2020a) also proposed a method for jointly optimizing a system that includes three decoders that respectively execute the functions of DST, Policy, and NLG. Liang et al. (2020) extended the method of Lei et al. (2018) by jointly optimizing four decoders that generate user dialogue acts (DAs), belief states, system DAs, and system responses. However, these systems are trained in a supervised manner and require large amounts of data (Liu et al., 2017).

Our study is related to improving the dialogue performance of a pipeline system by using RL. Zhao and Eskenazi (2016) and Li et al. (2017) implemented DST and Policy in a neural model and used the Deep Q Network (Mnih et al., 2013) algorithm to optimize the system to achieve robustness against errors that occur in interactions. Liu et al. (2018) proposed a Policy-learning optimization method for real users by combining supervised learning, imitation learning, and RL. Mehri et al. (2019) proposed a method for training a response-generation model by using RL while using the hidden states of the learned NLU, Policy, and NLG. Methods have been proposed (Lee et al., 2021; Lin et al., 2021) for building a pipeline system with individually trained modules and fine-tuning specific modules by using RL, which significantly improved the performance of the overall system. These methods are powerful because they can fine-tune a system directly through RL. However, they can only be applied to systems consisting of specific differentiable modules implemented using neural-based methods, not to systems consisting of non-differentiable modules. Our method is independent of the module-implementation method, trainability of each module in pipeline systems, and combination of modules.

## 3 Proposed Method

We developed our method to improve the dialogue performance of an entire pipeline system by optimizing the output of each module through post-processing. Post-processing means modifying the output by adding or removing information from the actual output of the module. With our method,

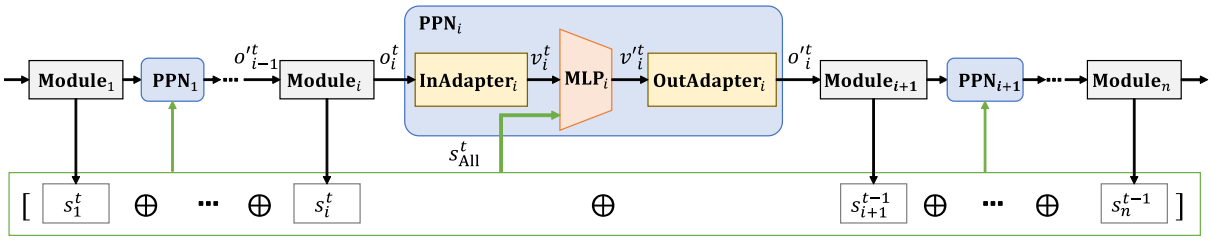



Figure 2: Architecture of our proposed method. Output of each module is post-processed by subsequent PPN. Each PPN has InAdapter to convert output label $o$ of module into multi-binary vector $v$, MLP to post-process multi-binary vector into $v'$ on basis of $v$ and state $s_{\text{All}}$ of all modules, and OutAdapter to restore $v'$ to output label $o'$.

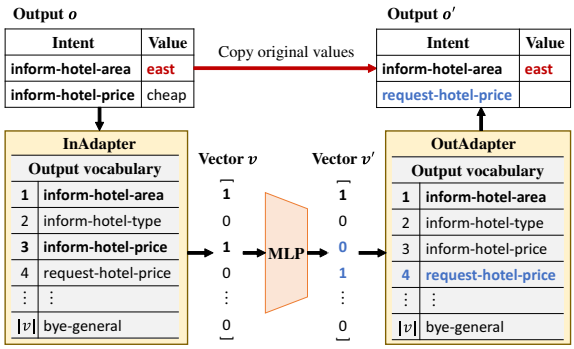



Figure 3: Procedure in which InAdapter converts output label $o$ into vector $v$ and OutAdapter restores vector $v'$ into output label $o'$ by using output vocabulary (in this case, output labels are DAs of NLU). Value information, which cannot be encoded in $v$, is copied directly from $o$ when creating $o'$.

each PPN needs to execute post-processing appropriate for all modules so that the entire system can improve overall dialogue performance. With this in mind, each PPN post-processes the target module's output while using the latest states of all modules in the system. Basically, each module's state is the latest output of each module. However, if a module can provide information that represents its state in more detail than the module's output, the PPN also uses that information (see Section 3.1). Figure 2 shows the architecture of PPNs applied to a pipeline system consisting of Module$_1$, ..., Module$_n$.

### 3.1 Post-processing Algorithm

The following equations describe the steps in which PPN$_i$ post-processes the output $o^t_i$ of Module$_i$ at turn $t$, as in Figure 2.

$$o^t_i, s^t_i = \text{Module}_i(o'^t_{i-1}) \quad (1)$$

$$v^t_i = \text{InAdapter}_i(o^t_i) \quad (2)$$

$$s^t_{\text{All}} = [s^t_1; ...; s^t_i; s^{t-1}_{i+1}; ...; s^{t-1}_n] \quad (3)$$

$$v'^t_i = \text{MLP}_i([v^t_i; s^t_{\text{All}}]) \quad (4)$$

$$o'^t_i = \text{OutAdapter}_i(v'^t_i) \quad (5)$$

As in a general pipeline system, Module$_i$ first receives the output $o'^t_{i-1}$ of the preceding Module$_{i-1}$ and outputs $o^t_i$ as the result of its processing (Eq. (1)) (e.g., for the NLU module, it receives the user's utterance as input and outputs the user's DAs). At the same time, Module$_i$ outputs its additional information $s^t_i$ obtained in the processing, which is related to the state of Module$_i$ (Eq. (1)). Basically, $s^t_i$ is the same as $o^t_i$. However, if Module$_i$ can provide more detailed information about its state obtained in the processing (e.g., for the NLU module, it typically outputs confidence scores of predicted user's DAs), Module$_i$ outputs that information as $s^t_i$.

Next, $o^t_i$ is input to PPN$_i$. In PPN$_i$, InAdapter$_i$ creates a multi-binary vector $v^t_i$, which is a vector representation of $o^t_i$ (Eq. (2)). The left half of Figure 3 shows a concrete example of an InAdapter converting a module output into a multi-binary vector. The InAdapter$_i$ is created by handcrafted rules using the output vocabulary set of Module$_i$. At the same time as creating $v^t_i$, $s^t_{\text{All}} = [s^t_1; ...; s^t_i; s^{t-1}_{i+1}; ...; s^{t-1}_n]$, which is a concatenation of the latest states of Module$_1$, ..., Module$_n$, are also created (Eq. (3)). Note that $s^{t-1}$ is used for states of Module$_{i+1}$, ..., Module$_n$ because modules after Module$_i$ have not produced their states in turn $t$.

The $v^t_i$ and $s^t_{\text{All}}$ created thus far are combined and input to multi-layer perceptron (MLP) MLP$_i$, which outputs a multi-binary vector $v'^t_i$ (Eq. (4)). The dimensions of $v'^t_i$ are the same as the vocabulary size of Module$_i$. At this point, the changes in the original vectors $v^t$ and $v'^t$ become the result of post-processing. That is, the dimension, the value in $v^t_i$ of which is 1 and value in $v'^t_i$ of which is 0, is the information deleted by MLP$_i$, and the reverse is the information added by MLP$_i$. Finally, OutAdapter$_i$ converts $v'^t_i$ into $o'^t_i$, the output label representation of Module$_i$. Some of the value information is directly copied from $o^t_i$ when creating

$o_i'^t$ since these values are not given by $v_i'^t$. If there is no need to fill in the value, it is left empty. The right half of Figure 3 shows a concrete example of an OutAdapter converting a multi-binary vector into a label representation of a module's vocabulary. As with InAdapter$_i$, OutAdapter$_i$ is created by hand-crafted rules using the output vocabulary set of Module$_i$.

At runtime, in the initial turn, the states of some modules that have never processed yet are initialized with zero vector (i.e., $s^0 = \mathbf{0}$). In the subsequent turn $t$, as mentioned above, PPN$_i$ uses the preceding modules' states $[s_1^t, ..., s_i^t]$ and the succeeding modules' states $[s_{i+1}^{t-1}, ..., s_n^{t-1}]$.

With our method, the MLPs of all PPNs are optimized jointly by using RL via interaction with users (see Section 3.3). To apply PPNs to a system, we only need the vocabulary set of each module to implement an InAdapter and OutAdapter for conversion. Therefore, our method can be applied to both differentiable and non-differentiable implementations of the modules. Since we want first to verify the idea of PPNs, we only used MLPs and focused on NLU, DST, and Policy in this study. Once the verification is complete, we aim to apply PPNs to more complex modules, such as NLG.

### 3.2 Pre-training with Imitation Learning

It is not easy to optimize an MLP from scratch by using RL. Many studies have shown that model performance can be improved by imitation learning, which is a scheme for learning to imitate the behavior of experts before RL is conducted (Argall et al., 2009; Rajeswaran et al., 2017). We considered the actual output $o_i$ of Module$_i$ to be the behavior of the expert for PPN$_i$ and conducted supervised learning so that PPN$_i$ copies $o^t$ before RL. This should allow each PPN to focus only on "how to modify the module's output $o$" during RL.

With our method, a pipeline system consisting of Module$_1$,..., Module$_n$ first executes dialogue sessions for sampling training data. In each dialogue, we sample the $[s_{\mathrm{All}}, v]$ of each module for all turns. At this stage, no PPNs execute post-processing, and no MLPs are used. When training MLPs by imitation learning, supervised learning is carried out using the sampled data. We train all MLPs to execute a multi-labeling task in which the input is $[v; s_{\mathrm{All}}]$ and the output label is $v$. Binary cross-entropy is used to update the MLP to minimize the difference between $v$ and $v' = \mathrm{MLP}([v; s_{\mathrm{All}}])$.

### 3.3 Optimization with Reinforcement Learning

The goal with PPNs is to improve dialogue performance (e.g., task success) by each PPN post-processing the output of each module. Therefore, the MLP of each PPN needs to be optimized using RL for maximizing the rewards related to dialogue performance. We use proximal policy optimization (PPO) (Schulman et al., 2017) as the RL algorithm, which is a stable and straightforward policy-gradient-based RL algorithm.

The following steps show the learning algorithm of a PPN for each iteration:

**Step. 1** The pipeline system with PPNs interacts with a user. Each PPN post-processes and samples the $s_{\mathrm{All}}^t$, $v^t$, $v'^t$, and reward $r^t$ of each MLP in turn $t$. The sampled $(s_{\mathrm{All}}^t, v^t, v'^t, r^t)$ are added to the post-processing history (called *trajectory*) of each PPN. As an $r^t$, we give the same value to all PPNs. These trials are repeated until the trajectory reaches a predetermined size (called *horizon*).

**Step. 2** The PPN to be updated in this iteration is selected on the basis of the *PPN-selection strategy*, which is a rule for selecting PPNs to be updated in each iteration. We have three strategies described in the next paragraph.

**Step. 3** The MLPs of the PPNs selected in Step. 2 are updated using the PPO algorithm. Each MLP is updated for multiple epochs using the trajectory sampled in Step. 1 as training data.

Since it is not apparent which modules' PPN should be updated and in what order, we prepared the following three PPN-selection strategies: ALL (select all PPNs in every iteration), RANDOM (randomly select one or more PPNs in each iteration), and ROTATION (select one PPN at each iteration in order). In the following experiments, we examined which strategy is the best.

## 4 Experiments

To confirm the effectiveness of our method, we applied PPNs to several different pipeline systems and evaluated dialogue performance using dialogue simulation. We also carried out a human evaluation.

### 4.1 Dataset

We evaluated PPNs using modules and a user simulator implemented using the MultiWOZ dataset (Budzianowski et al., 2018), which is a task-oriented dialogue dataset between a clerk and tourist at an information center. MultiWOZ contains 10,438 dialogues; one to three domains (seven domains in total in the dataset) appear simultaneously in each dialogue.

### 4.2 Platform and User Simulator

ConvLab-2[2] (Zhu et al., 2020) is a platform for multi-domain dialogue systems, which provides pre-implemented models of each module in the pipeline system and tools for end-to-end evaluation of the dialogue system.

We used the user simulator implemented in ConvLab-2. The simulator interacts with the dialogue system in natural language on the basis of the user goal given for each dialogue session. The simulator consists of a BERT (Devlin et al., 2019)-based NLU (Chen et al., 2019), an agenda-based Policy (Schatzmann et al., 2007), and a template-based NLG. The agenda-based Policy models a user's behavior in MultiWOZ by using a stack-like agenda created using hand-crafted rules. A user goal for each dialogue is randomly generated: the domains are randomly selected from one to three domains (out of all seven domains) on the basis of the domains' frequency in MultiWOZ; the slots are also randomly selected on the basis of the slots' frequency in MultiWOZ.

### 4.3 Evaluation Metrics

In evaluating each dialogue, we used the number of turns[3] (Turn) to measure the efficiency of completing each dialogue; the smaller the Turn is, the better the system performance. We also measured whether the system responds to the requested slot by the user without excess or deficiency (Inform F1) and whether the entity presented by the system met the condition of the user goal (Match Rate). We also used Task Success as a result of Match Rate and Inform Recall being equal to 1 within 20 turns. The above four metrics are the major ones for dialogue evaluation and have been used in many studies using ConvLab-2 (Li et al., 2020; Takanobu et al., 2020; Hou et al., 2021).

### 4.4 Implementation

#### 4.4.1 System Configurations

To select the modules that make up a pipeline system, we referred to Takanobu et al. (2020), who developed and evaluated various combinations of modules using ConvLab-2. For the models of each module (NLU, DST, Policy, and NLG), we included both classical rule-based and recent neural-based models. Note that, since this study focused on whether PPNs can be used to optimize pipeline systems consisting of non-trainable modules, we did not update modules even if the modules may be trainable. Each of the models[4] we prepared are as follows.

**NLU** We used BERT NLU (Chen et al., 2019) for the NLU module. This model estimates DAs by tagging which domain-intent-slot each token in a user utterance represents by using a pre-trained BERT (Devlin et al., 2019). The InAdapter/OutAdapter are created using the DA set defined in BERT NLU (see Figure 3 for an illustration of an InAdapter-processing example by using a DA set). We used the estimated probabilities of each DA as BERT NLU's state $s$.

**DST** We used two models for the DST module: Rule DST (Zhu et al., 2020) and TRADE (Wu et al., 2019). Rule DST updates the dialogue state consisting of belief state, database search results, current user DAs, and previous system DAs at each turn by directly using the DAs estimated by the NLU. On the contrary, TRADE is a neural-based model that directly extracts slot-value pairs and generates belief states using the dialogue history as input. For DST modules, a belief state is subject to post-processing. Therefore, we created an InAdapter/OutAdapter on the basis of the slot types defined in the belief state on ConvLab-2. As states of Rule DST and TRADE, an entire dialogue state is converted into a multi-binary vector by using a vectorizer implemented in ConvLab-2.

**Policy** We used four models for the Policy module: Rule Policy (Zhu et al., 2020), MLE Policy, PPO Policy (Schulman et al., 2017), and LaRL Policy (Zhao et al., 2019). Rule Policy is a model based on hand-crafted rules. MLE Policy is a model trained on state-action pairs in MultiWOZ using supervised learning. PPO Policy is

[2] https://github.com/thu-coai/ConvLab-2

[3] One user utterance and its system response form one turn.

[4] For models, we used the best ones provided by ConvLab-2 as of October 20, 2021

| Module | Models | $\|s\|$ | $\|v\|$ |
|---|---|---|---|
| NLU | BERT | 175 | 175 |
| DST | Rule, TRADE | 340 | 24 |
| Policy | Rule, MLE, PPO | 209 | 209 |
| | LaRL | 0 | 0 |
| NLG | Template, SC-LSTM | 0 | 0 |

Table 1: Dimensions $|s|$ of state $s$ output from each module and $|v|$ of vector $v$ processed by PPN of each module. Number of output vocabularies defined for each module and $|v|$ are equal.

a fine-tuned model based on MLE Policy using the PPO RL algorithm. Unlike the other Policy models, LaRL Policy is an LSTM-based model trained to directly generate system utterances instead of system DAs by using RL. We created an InAdapter/OutAdapter using the DA set defined in each model. For states of MLE Policy and PPO Policy, we used the estimated probability of each DA. For Rule Policy's state, we used a binary vector representation of DAs. Since the output of LaRL is a natural language, it was not subject to post-processing in this study.

**NLG** We used two models for the NLG module: Template NLG and SC-LSTM (Wen et al., 2015). Template NLG creates system responses by inserting values into templates of utterances manually created in advance for each DA. SC-LSTM is an LSTM-based model that generates utterances on the basis of DAs. For the same reason as for LaRL Policy, we did not implement PPNs for Template NLG and SC-LSTM in these experiments.

Table 1 shows the dimensions of each module's state $s$ described above and the number of dimensions of the multi-binary vector $o$ of each PPN (i.e., the vocabulary of each module). Note that for the DST modules, the dimensions of $s$ and $v$ are different. This is because $s$ is a vector representation of a dialogue state, which includes a belief state, database search results, user's DAs, etc., and $v$ is a vector representation of a belief state only.

#### 4.4.2 Training

Throughout all experiments, the data used for imitation learning of each pipeline system was sampled by simulating 10,000 turns, corresponding to approximately 1,000 dialogue sessions. In RL for each system, we trained 200 iterations, where one iteration consists of approximately 100 dialogue sessions. Following Takanobu et al. (2019), we gave a reward of $-1$ for each turn, and when the task was a success, we gave the maximum number of turns $\times$ 2 at the end of the dialogue session, i.e., 40 in our case. See Section A.1 of the appendix for more training details.

To test each system, we ran 1,000 dialogues using a system that achieved the best Task Success during the RL training. Throughout all experiments, we trained with five different random seeds and reported the average of their scores as the final performance.

| PPN-selection strategy | Success | Inform | Match | Turn |
|---|---|---|---|---|
| ALL | 64.2 | **71.9** | 76.6 | 9.20 |
| RANDOM | **66.1** | 71.5 | **78.7** | **8.61** |
| ROTATION | 60.4 | 70.5 | 73.2 | 9.10 |

Table 2: Performance after PPN training with each PPN-selection strategy

### 4.5 Experimental Procedure

We conducted four experiments. The first experiment was conducted to determine which of the PPN-selection strategies (see Section 3.3) is appropriate. We used a combination of BERT NLU, Rule DST, MLE Policy, and Template NLG as the system configuration. The reasons for using this combination are that (1) the Task Success of a system composed of this module combination is around 50%. Therefore, it would be easy to understand the impact of the PPNs, and (2) MLE Policy is used as the initial weight in many RL methods (Takanobu et al., 2019; Li et al., 2020), making it a reasonable starting point for RL. The second experiment was conducted to verify whether the PPNs work for any combination of modules; we combined some of the modules described in Section 4.4.1 to build pipeline systems and applied PPNs. The third experiment was conducted to investigate the contribution of the PPN of each module and $s_{\mathrm{All}}$ to the overall performance of the system. The final experiment was a human evaluation; we examined whether the proposed method is effective not only for a simulator but also for humans.

### 4.6 Comparison of Post-processing-network-selection Strategies

Figure 4 shows the learning process in the three PPN-selection strategies. Task Success and Inform F1 at 50 iterations show that ALL reached the highest score about 100 iterations earlier than RANDOM and ROTATION. This is a reasonable result

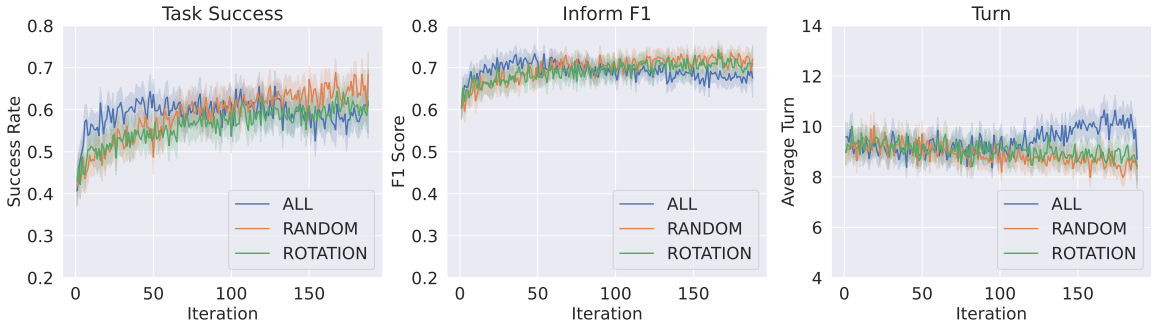



Figure 4: Scores of each evaluation metric in learning process with three PPN-selection strategies

| System | Model Combination | | | | w/ PPN | Task Success | Inform F1 | Match Rate | Turn |
|---|---|---|---|---|---|---|---|---|---|
| | NLU | DST | Policy | NLG | | | | | |
| SYS-RUL | BERT | Rule | Rule | Template | | 84.1 | 87.4 | 90.2 | 5.92 |
| | | | | | ✓ | 84.0 | 86.3 | **92.4** | 6.33 |
| SYS-MLE | BERT | Rule | MLE | Template | | 43.3 | 62.4 | 27.8 | 9.03 |
| | | | | | ✓ | **66.1** | **71.5** | **78.7** | **8.61** |
| SYS-PPO | BERT | Rule | PPO | Template | | 54.9 | 65.5 | 55.2 | 8.41 |
| | | | | | ✓ | **68.8** | **72.1** | **77.8** | **8.37** |
| SYS-SCL | BERT | Rule | Rule | SC-LSTM | | 38.3 | 57.5 | 56.7 | 13.53 |
| | | | | | ✓ | **44.2** | **71.7** | **71.8** | **11.04** |
| SYS-TRA | TRADE | | Rule | Template | | 19.0 | 45.6 | 36.4 | 12.08 |
| | | | | | ✓ | 18.8 | **49.2** | 31.6 | 12.14 |
| SYS-LAR | BERT | Rule | LaRL | | | 21.6 | 44.9 | 27.6 | 13.24 |
| | | | | | ✓ | **23.9** | **50.9** | **34.1** | **12.77** |

Table 3: Combination of models for each pipeline system and scores before and after applying PPNs to each system. 'w/ PPN' indicates whether PPNs are applied to the system. Scores that have been improved using PPNs are in bold.

since the number of updates for each MLP in ALL was up to four times that for the other strategies. However, it was unstable after 50 iterations, and the scores of Task Success, Inform F1, and Turn all worsened as the learning process progressed. This is probably because the gradients of each MLP were calculated simultaneously in the PPO update algorithm, which caused each MLP to update in a different gradient direction, making it difficult for each MLP to coordinate with one another.

Although the learning speed of ROTATION and RANDOM was slow, all metrics consistently improved. Turn and Inform F1 also showed stable improvements compared with ALL. For RANDOM and ROTATION, each MLP computed its gradient after the other MLPs computed and updated their gradients one by one, which probably prevented significant discrepancies among MLPs and stabilized learning.

Table 2 shows the final performance of each strategy. RANDOM outperformed ALL in all the final scores, and ROTATION was inferior to ALL in Task Success, Inform F1, and Match Rate. Since the learning was stable and the final performance was generally better than the other strategies, we decided to use RANDOM in the following experiments.

### 4.7 Comparison of Model Combinations

We built six pipeline systems with different model combinations. Table 3 summarizes the comparison of the scores when PPNs were applied to each system. For a fair comparison, systems without PPNs were also evaluated on the average scores[5] of 1,000 dialogues conducted with five different random seeds.

Table 3 shows that Task Success improved for most of the systems. In addition, all systems improved in Inform F1 or Match Rate. These results indicate that post-processing with PPNs can improve the dialogue performance of a pipeline system without touching the module internals. However, neither Task Success nor Turn improved for SYS-RUL and SYS-TRA. The common feature of these two systems is that they use Rule Policy and Template NLG. These modules are carefully designed by hand and originally have high accu-

[5]Although we used the latest models implemented in ConvLab-2, we could not reproduce the scores reported in https://github.com/thu-coai/ConvLab-2#end-to-end-performance-on-multiwoz

| System | w/ $s_{\text{All}}$ | Success | Inform | Match | Turn |
|---|---|---|---|---|---|
| SYS-MLE | | 43.3 | 62.4 | 27.8 | 9.03 |
| +$\text{PPN}_{\text{NLU}}$ | | 59.6 | **73.1** | 65.8 | 9.59 |
| +$\text{PPN}_{\text{DST}}$ | | 46.7 | 65.1 | 36.7 | 9.41 |
| +$\text{PPN}_{\text{Policy}}$ | | 59.9 | 67.3 | 67.9 | 9.20 |
| +$\text{PPN}_{\text{All}}$ | | 59.7 | 68.0 | 69.9 | 9.84 |
| +$\text{PPN}_{\text{NLU}}$ | ✓ | 62.2 | 72.1 | 64.0 | 9.36 |
| +$\text{PPN}_{\text{DST}}$ | ✓ | 47.9 | 66.1 | 40.2 | 9.21 |
| +$\text{PPN}_{\text{Policy}}$ | ✓ | 65.8 | 67.6 | 76.9 | **8.56** |
| +$\text{PPN}_{\text{All}}$ | ✓ | **66.1** | 71.5 | **78.7** | 8.61 |
| +Fine-tuned Policy | | 71.9 | 74.3 | 80.4 | 7.88 |

Table 4: Impact analysis of PPNs. Subscripts (i.e., NLU, DST, Policy, and All) indicate that PPN was applied to that one specific module or all modules. 'w/ $s_{\text{All}}$' indicates whether $s_{\text{All}}$ was used. Row of Fine-tuned Policy shows scores when SYS-MLE's Policy was fine-tuned using RL.

racy, leading to little room for improvement in this configuration.

In general, there were large differences in performance among the systems regardless of whether PPN was used. As mentioned above, this is due to the performance differences among the modules comprising the systems. For example, SYS-RUL is considered to have significantly higher performance than the other systems due to the use of elaborately designed rules and templates.

### 4.8 Impact of Post-processing Networks

We investigated the impact of each module's PPN and $s_{\text{All}}$. We used SYS-MLE as a base configuration for this experiment since its performance was most improved with our method (see Table 3); we considered it appropriate to measure the impact of PPNs. In Table 4, the results of applying PPNs to only one of the NLU, DST, and Policy are shown, as well as the results of applying PPNs without using $s_{\text{All}}$. The system performance consistently improved when only a single module's PPN was applied. In particular, +$\text{PPN}_{\text{Policy}}$ achieved the best performance (Task Success improved by more than 20%), indicating that the PPN of Policy contributed the most to dialogue performance. When $s_{\text{All}}$ was not used, most of the scores decreased. This indicates that each PPN can execute post-processing more appropriately by using the states of all modules in the system.

To confirm the degree of performance improvement achieved with the PPNs, the method of fine-tuning the modules by using RL was used as the upper bound of post-processing. Only the Policy module was fine-tuned, as is common with conventional methods (Liu et al., 2018; Lin et al., 2021).

| System | Success | Turn | Und. | App. | Sat. |
|---|---|---|---|---|---|
| SYS-MLE | 39.0 | 11.0 | 2.93 | 3.12 | 2.46 |
| +$\text{PPN}_{\text{NLU}}$ | 53.7 | 11.1 | 3.10 | 3.37 | 2.93 |
| +$\text{PPN}_{\text{DST}}$ | 60.0 | 10.4 | **3.30** | **3.43** | **3.28**$^*$ |
| +$\text{PPN}_{\text{Policy}}$ | **62.5**$^*$ | **8.20**$^*$ | 2.93 | 3.03 | 3.00 |
| +$\text{PPN}_{\text{All}}$ | 57.5 | 9.00 | 2.83 | 3.00 | 2.95 |

Table 5: Results of human evaluation for each system configuration. Asterisks indicate statistically significant differences ($p < 0.05$) over SYS-MLE.

The bottom row of Table 4 shows the results when the Policy of SYS-MLE was fine-tuned by PPO (Schulman et al., 2017) (see Section A.2 of the appendix for training details). The difference between +$\text{PPN}_{\text{All}}$ and +Fine-tuned Policy is small with 5.8%. This is a promising result considering that our proposed method does not touch on the internal architecture of Policy.

### 4.9 Human Evaluation

Five systems (SYS-MLE and four systems with our proposed method, i.e., +$\text{PPN}_{\text{NLU}}$, +$\text{PPN}_{\text{DST}}$, +$\text{PPN}_{\text{Policy}}$, and +$\text{PPN}_{\text{All}}$) in Table 4 were used for the human evaluation. Not that $s_{\text{All}}$ was used in all four systems. About forty Amazon Mechanical Turk (AMT) crowd workers were recruited to interact with each of the five systems and judged on Task Success. As in the simulation experiments (see Section 4.2), user goals were randomly generated for each dialogue. After the interaction, the workers also evaluated the system's ability to understand the language (Und.), accuracy of the system's responses (App.), and overall satisfaction with the interaction (Sat.) on a 5-point Likert scale. See Section B of the appendix for the procedures taken by the workers.

Table 5 shows the results. All four systems with our proposed method performed better than SYS-MLE, which is similar to the result in Table 4. Wilcoxon rank-sum tests were conducted using the top score in each evaluation metric and the score of SYS-MLE, and statistically significant differences were confirmed for Task Success and Turn in +$\text{PPN}_{\text{Policy}}$ and interaction satisfaction in +$\text{PPN}_{\text{DST}}$. In contrast, there were no significant differences in scores for language understanding and responses' appropriateness. This is probably because RL was conducted with rewards that only relied on Task Success and Turn. The performance of +$\text{PPN}_{\text{NLU}}$ did not improve as much as in Table 4. A possible reason is the overfitting of +$\text{PPN}_{\text{NLU}}$ with the user simulator. The same over-

fitting might have occurred in the NLU's PPN in $+\mathrm{PPN}_{\mathrm{All}}$, which resulted in a smaller improvement in scores of $+\mathrm{PPN}_{\mathrm{All}}$.

We also investigated how PPNs executed post-processing by analyzing the actual dialogue logs collected in this experiment. A specific case study is described in Section C of the appendix. Generally, in the dialogue of $+\mathrm{PPN}_{\mathrm{Policy}}$, we observed that $\mathrm{PPN}_{\mathrm{Policy}}$ added necessary DAs when the original Policy failed to output them.

## 5 Conclusions and Future Work

We proposed a method for optimizing pipeline dialogue systems with post-processing networks (PPNs). Through dialogue simulation and human evaluation experiments on the MultiWOZ dataset, we showed that the proposed method is effective for a pipeline system consisting of modules with various models.

For future work, we plan to design more sophisticated rewards in RL such as module-specific rewards. We also plan to extend PPNs to handle natural language generation by implementing them using Transformer-based models. We are also considering to apply PPNs to modules dealing with speech recognition and multi-modal processing.

## Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 19H05692. We used the computational resource of the supercomputer "Flow" at Information Technology Center, Nagoya University. We thank Yuya Chiba and Yuiko Tsunomori for their helpful comments and feedback. Thanks also go to Ao Guo for his advice on the human evaluation experiment.

## References

Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, pages 469–483.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026.

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *ACM SIGKDD Explorations Newsletter*, pages 25–35.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. BERT for Joint Intent Classification and Slot Filling. *arXiv preprint arXiv:1902.10909*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, Shrimai Prabhumoye, Alan W. Black, Alexander Rudnicky, Jason Williams, Joelle Pineau, Mikhail Burtsev, and Jason Weston. 2019. The Second Conversational Intelligence Challenge (ConvAI2). *arXiv preprint arXiv:1902.00098*.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. 2020. Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TPRO. *arXiv preprint arXiv:2005.12729*.

Jianfeng Gao, Michel Galley, and Lihong Li. 2018. Neural Approaches to Conversational AI. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1371–1374.

Chulaka Gunasekara, Seokhwan Kim, Luis Fernando D'Haro, Abhinav Rastogi, Yun-Nung Chen, Mihail Eric, Behnam Hedayatnia, Karthik Gopalakrishnan, Yang Liu, Chao-Wei Huang, Dilek Hakkani-Tür, Jinchao Li, Qi Zhu, Lingxiao Luo, Lars Liden, Kaili Huang, Shahin Shayandeh, Runze Liang, Baolin Peng, Zheng Zhang, Swadheen Shukla, Minlie Huang, Jianfeng Gao, Shikib Mehri, Yulan Feng, Carla Gordon, Seyed Hossein Alavi, David Traum, Maxine Eskenazi, Ahmad Beirami, Cho Eunjoon, Paul A. Crook, Ankita De, Alborz Geramifard, Satwik Kottur, Seungwhan Moon, Shivani Poddar, and Rajen Subba. 2020. Overview of the Ninth Dialog System Technology Challenge: DSTC9. *arXiv preprint arXiv:2011.06486*.

Zhengxu Hou, Bang Liu, Ruihui Zhao, Zijing Ou, Yafei Liu, Xi Chen, and Yefeng Zheng. 2021. Imperfect also Deserves Reward: Multi-Level and Sequential Reward Modeling for Better Dialog Management. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2993–3001.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Hwaran Lee, Seokhwan Jo, Hyungjun Kim, Sangkeun Jung, and Tae-Yoon Kim. 2021. SUMBT+LaRL: Effective Multi-Domain End-to-End Neural Task-Oriented Dialog System. *IEEE Access*, pages 116133–116146.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying Task-oriented Dialogue Systems with Single Sequence-to-Sequence Architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1437–1447.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017. End-to-End Task-Completion Neural Dialogue Systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, pages 733–743.

Ziming Li, Sungjin Lee, Baolin Peng, Jinchao Li, Julia Kiseleva, Maarten de Rijke, Shahin Shayandeh, and Jianfeng Gao. 2020. Guided Dialogue Policy Learning without Adversarial Learning in the Loop. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2308–2317.

Weixin Liang, Youzhi Tian, Chengcai Chen, and Zhou Yu. 2020. MOSS: End-to-End Dialog System Framework with Modular Supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8327–8335.

Zichuan Lin, Jing Huang, Bowen Zhou, Xiaodong He, and Tengyu Ma. 2021. Joint System-Wise Optimization for Pipeline Goal-Oriented Dialog System. *arXiv preprint arXiv:2106.04835*.

Bing Liu, Gokhan Tur, Dilek Hakkani-Tur, Pararth Shah, and Larry Heck. 2017. End-to-End Optimization of Task-Oriented Dialogue Model with Deep Reinforcement Learning. *arXiv preprint arXiv:1711.10712*.

Bing Liu, Gokhan Tür, Dilek Hakkani-Tür, Pararth Shah, and Larry Heck. 2018. Dialogue Learning with Human Teaching and Feedback in End-to-End Trainable Task-Oriented Dialogue Systems. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2060–2069.

Shikib Mehri, Tejas Srinivasan, and Maxine Eskenazi. 2019. Structured Fusion Networks for Dialog. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 165–177.

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*.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. 2017. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. *arXiv preprint arXiv:1709.10087*.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-Based User Simulation for Bootstrapping a POMDP Dialogue System. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, pages 149–152.

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*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of Advances in neural information processing systems*, pages 3104–3112.

Ryuichi Takanobu, Hanlin Zhu, and Minlie Huang. 2019. Guided Dialog Policy Learning: Reward Estimation for Multi-Domain Task-Oriented Dialog. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 100–110.

Ryuichi Takanobu, Qi Zhu, Jinchao Li, Baolin Peng, Jianfeng Gao, and Minlie Huang. 2020. Is Your Goal-Oriented Dialog Model Performing Really Well? Empirical Analysis of System-wise Evaluation. In *Proceedings of the 21st Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 297–310.

Bo-Hsiang Tseng, Yinpei Dai, Florian Kreyssig, and Bill Byrne. 2021. Transferable Dialogue Systems and User Simulators. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 152–166.

Stefan Ultes, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Dongho Kim, Iñigo Casanueva, Paweł Budzianowski, Nikola Mrkšić, Tsung-Hsien Wen, Milica Gašić, and Steve Young. 2017. PyDial: A Multi-domain Statistical Dialogue System Toolkit. In *Proceedings of ACL 2017, System Demonstrations*, pages 73–78.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A Network-based End-to-End Trainable Task-oriented Dialogue System. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 438–449.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable Multi-Domain State Generator for Task-Oriented Dialogue Systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 808–819.

Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020a. Task-Oriented Dialog Systems That Consider Multiple Appropriate Responses under the Same Context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9604–9611.

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

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*, pages 1–10.

Tiancheng Zhao, Kaige Xie, and Maxine Eskenazi. 2019. Rethinking Action Spaces for Reinforcement Learning in End-to-end Dialog Agents with Latent Variable Models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1208–1218.

Qi Zhu, Zheng Zhang, Yan Fang, Xiang Li, Ryuichi Takanobu, Jinchao Li, Baolin Peng, Jianfeng Gao, Xiaoyan Zhu, and Minlie Huang. 2020. ConvLab-2: An Open-Source Toolkit for Building, Evaluating, and Diagnosing Dialogue Systems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 142–149.

# A Training Details

## A.1 Training Post-processing Networks

**Model** All MLPs of the PPNs for all modules are implemented in three layers: one input layer, one hidden layer, and one output layer, and the dimensionality of the hidden layer is 128 for all layers. The number of dimensions of the input and output layers are $|o| + |s_{\text{All}}|$ and $|o_i|$, respectively. The activation functions are all ReLUs.

**Imitation Learning** The sampled data of 10,000 turns were split as $\text{training} : \text{validation} = 8 : 2$. All MLPs were trained on a batch size of 32 for 20 epochs using the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 1e-3. The weights at the epoch with the highest accuracy for validation were used for the following RL.

**Reinforcement Learning** The hyperparameters shown in Table 6 were determined with reference to the implementation of PPO in ConvLab-2. We used Generalized Advantage Estimation (Schulman et al., 2015). Referring to Engstrom et al. (2020), the learning rate was annealed linearly in accordance with the current iteration. The computational resource used was a single NVIDIA Tesla V100 SXM2 GPU with 32GB RAM. In training, the trajectory was sampled in parallel by eight processes, and it took 5 to 17 hours, depending on the system, to complete the training of 200 iterations.

## A.2 Fine-tuning of Policy

The MLE Policy of SYS-MLE in Section 4.8 was fine-tuned with PPO using the same user simulator used for training PPNs. The hyperparameters used for training were the same as those used in ConvLab-2, as shown in Table 6. To evaluate the fine-tuned Policy, training and testing (consisting of 1,000 dialogue sessions) were conducted with five random seeds.

| Hyperparameters | | PPN | Fine-tuned Policy |
|---|---|---|---|
| Number of iterations | | 200 | 200 |
| Batch size | | 1024 | 1024 |
| Epoch | | 5 | 5 |
| Mini batch size | | 32 | 32 |
| Discount factor $\gamma$ | | 0.99 | 0.99 |
| GAE factor $\lambda$ | | 0.95 | 0.95 |
| Optimizer | policy net | Adam | RMSprop |
| | value net | | Adam |
| Learning rate | policy net | 1e-4 | 1e-4 |
| | value net | | 5e-5 |

Table 6: Hyperparameter settings in PPO

# B Details of Human Evaluation

Referring to Takanobu et al. (2020), we designed the following experimental procedure. First, each worker is presented with an instruction for a randomly generated user goal. Next, the user interacts with one of the five systems in Table 5 for up to 20 turns. Workers determine whether the interaction succeeded or failed within 20 turns; after 20 turns, the interaction is automatically marked as failed. To ensure the quality of the workers, several qualifications were set; the eligible workers should (1) reside in an English-speaking country, (2) have a task accomplishment number on AMT greater than 10, (3) have a task-approval rate greater than 90%, and (4) correctly answer all the common sense questions (total of five questions) we designed. The time limit for the task was 10 minutes, and the reward was $1.7. To account for workers who may cancel the task amid the dialogue session, more than 40 workers were recruited for each system. At the end, 41 workers for SYS-MLE and $+\text{PPN}_{\text{NLU}}$ and 40 workers for $+\text{PPN}_{\text{DST}}$, $+\text{PPN}_{\text{Policy}}$, and $+\text{PPN}_{\text{All}}$ participated in the experiment.

# C Case Study

Tables 7 and 8 show the user goal and example of a dialogue session between a worker and $+\text{PPN}_{\text{Policy}}$, respectively, in the human evaluation experiment. Table 8 also includes the outputs of the system's Policy and the post-processing result of PPNs. In S1, the original MLE Policy did not generate any DAs. At this time, $\text{PPN}_{\text{Policy}}$ generated additional DAs informing the user that "free parking is available at the hotel found in the database" and asked if there was anything else the user needs. In S2, the user requested a reservation, but MLE Policy could not make the reservation. In contrast, $\text{PPN}_{\text{Policy}}$ successfully made the reservation and provided a booking reference number.

| Domain | Task | Slot |
|---|---|---|
| Hotel | Info | Area = West<br>Parking = Yes |
| | Book | Day = Sunday<br>Time = 11:00<br>People = 1 |
| Restaurant | Fail Info | Area = West<br>Food = Mediterranean |
| | Info | Area = West<br>Food = Thai |
| | Reqt | Postcode<br>Address |

Table 7: User goal used in Table 8

| | |
|---|---|
| U1 | I need a hotel in the west with free parking for 7 people on saturday for 3 nights |
| S1 | **Policy outputs:** |
| | **PPN$_{\text{Policy}}$ outputs:** Inform-Hotel-Parking-yes, reqmore-general |
| | **System response:** It offers free parking . Is there anything else I can help you with today ? |
| U2 | can i book a room for 7 people for 3 nights on Saturday |
| S2 | **Policy outputs:** reqmore-general-none |
| | **PPN$_{\text{Policy}}$ outputs:** Book-Booking-Ref-00000017, reqmore-general |
| | **System response:** Here is the booking information : Booking was successful . Reference number is : 00000017. You are welcome . Is there anything else I can help you with today ? |
| U3 | I need a mediterranean restaurant in the west |
| S3 | **Policy outputs:** NoOffer-Restaurant-Area-west, NoOffer-Restaurant-Food-Mediterranean, NoOffer-Restaurant, Request-Restaurant-Food |
| | **PPN$_{\text{Policy}}$ outputs:** Inform-Taxi-Car-ford, NoOffer-Restaurant-Food-Mediterranean, NoOffer-Restaurant, Request-Restaurant-Food |
| | **System response:** A ford is booked for you . There are no mediterranean restaurants unfortunately . No , I ' m sorry . The search did n't pull up any matches . What type of food would you like ? |
| U4 | I need a thai restaurant in the west |
| S4 | **Policy outputs:** Inform-Restaurant-Name-sala thong, Inform-Restaurant-Area-west, Inform-Restaurant-Food-thai, Inform-Restaurant-Price-expensive, Inform-Booking |
| | **PPN$_{\text{Policy}}$ outputs:** Inform-Restaurant-Name-sala thong, Inform-Restaurant-Area-west, Inform-Restaurant-Food-thai, Inform-Restaurant-Price-expensive, Inform-Hotel-Internet-yes, Inform-Booking |
| | **System response:** How does the sala thong sound ? It is in the west area . That is a thai restaurant . They are expensive. The place provides free wifi . Would you like for me to try and make a reservation ? |
| U5 | Yes |
| S5 | **Policy outputs:** Inform-Booking, Request-Booking-Day, Request-Booking-People, Request-Booking-Time, reqmore-general |
| | **PPN$_{\text{Policy}}$ outputs:** Inform-Hospital-Phone-01223216297, Inform-Booking, Request-Booking-Day, Request-Booking-People, Request-Booking-Time, reqmore-general |
| | **System response:** The phone is 01223216297 . Would you like me to book it ? What day would you like that reservation ? How many people will be with you ? What time would you like me to make your reservation ? You are welcome . Is there anything else I can help you with today ? |
| U6 | yes. what is the address? |
| S6 | **Policy outputs:** Request-Booking-Day, Request-Booking-Time, reqmore-general |
| | **PPN$_{\text{Policy}}$ outputs:** Inform-Hotel-Addr-144 thornton road, Inform-Hotel-Post-cb30nd, Book-Booking-Ref-00000017, Request-Booking-Day, Request-Booking-People, reqmore-general |
| | **System response:** They are located at 144 thornton road. The postal code for that hotel is cb30nd . Here is the booking information : Booking was successful . Reference number is : 00000017. What day would you like your booking for ? How many people will be with you ? Is there anything else I can help you with today ? |
| U7 | No |
| S7 | **Policy outputs:** reqmore-general |
| | **PPN$_{\text{Policy}}$ outputs:** reqmore-general |
| | **System response:** Did you need any further assistance today ? |
| U8 | **Success** |

Table 8: Example of dialogue history between worker and +PPN$_{\text{Policy}}$ in human evaluation experiment. DAs appropriately added by PPN$_{\text{Policy}}$ are in blue, and those inappropriately added are in red.