# Toward Self-Learning End-to-End Task-Oriented Dialog Systems

**Xiaoying Zhang**[1], **Baolin Peng**[2], **Jianfeng Gao**[2], **Helen Meng**[1]
[1]The Chinese University of Hong Kong, Hong Kong
[2]Microsoft Research, Redmond
{zhangxy, hmmeng}@se.cuhk.edu.hk
{bapeng, jfgao}@microsoft.com

## Abstract

End-to-end task bots are typically learned over a static and usually limited-size corpus. However, when deployed in dynamic, changing, and open environments to interact with users, task bots tend to fail when confronted with data that deviate from the training corpus, *i.e.,* out-of-distribution samples. In this paper, we study the problem of automatically adapting task bots to changing environments by learning from human-bot interactions with minimum or zero human annotations. We propose SL-AGENT[1], a novel self-learning framework for building end-to-end task bots. SL-AGENT consists of a dialog model and a pre-trained reward model to predict the quality of an agent response. It enables task bots to automatically adapt to changing environments by learning from the unlabeled human-bot dialog logs accumulated after deployment via reinforcement learning with the incorporated reward model. Experimental results on four well-studied dialog tasks show the effectiveness of SL-AGENT to automatically adapt to changing environments, using both automatic and human evaluations. We will release code and data for further research.

## 1 Introduction

The most common approach of building end-to-end task-oriented dialog systems is to train neural models to imitate human behaviors in fixed task-specific annotated corpora (Gao et al., 2018; Zhang et al., 2020). Existing state-of-the-art approaches usually adopt Pre-trained Language Models (PLMs) (Peng et al., 2020a; Ham et al., 2020; Hosseini-Asl et al., 2020) to build end-to-end dialog systems. However, these data-driven approaches assume an independent and identically distributed (IID) data setting[2], *i.e.,* a static environment[3], and usually exhibit a tendency of failure, when confronted with out-of-distribution (OOD) examples in real-world scenarios, *i.e.,* changing environments.

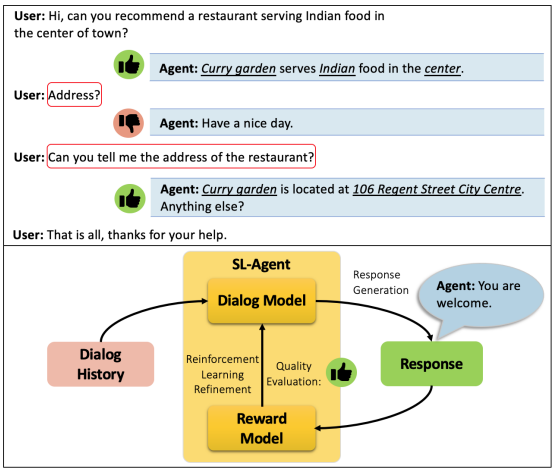



Figure 1: Illustration of the proposed SL-AGENT with a human-bot dialog example. $(i)$ The human-bot dialog example, containing an inappropriate response related to unseen user behaviors (upper part). $(ii)$ Demonstration of the refining process in SL-AGENT with the exhibited dialog example (lower part).

In the context of task-oriented dialog systems, changing environments are quite common and arise from the following two aspects: $(i)$ *unseen user behaviors* – real users may query with unseen language patterns and unknown user goals (*i.e.,* unseen slot values and dialog flows) of the designated tasks outside the pre-built training corpora (Liu et al., 2018; Peng et al., 2020b). For example, real users may query entities in the database but not covered by the training examples. $(ii)$ *task definition extensions* – dialog systems need to handle new functions or new tasks as user and business requirements evolve, *i.e.,* add new slot types (Lipton et al., 2018; Gasic et al., 2014). For example, a restaurant bot designed for the table-booking service may also encounter queries about delivery service after deployment. These human-bot interactions accu-

[1]SELF-LEARNING AGENT.
[2]Assume the same user behaviors at deployment as in the training stage.
[3]Environment is the Agent's world in which it lives and interacts.



*Proceedings of the SIGdial 2022 Conference*, pages 516–530
Heriot-Watt University, Edinburgh, UK. 07-09, September, 2022 

mulated after deployment are cheap, dynamic and contain useful information (Hancock et al., 2019), *i.e.,* unseen user behaviors are related to the training examples and the probabilistic dialog model may generate appropriate responses. As shown in the upper part of Figure 1, when user queries casually about address, the system fails to provide address in the second response, but gives it in the third response, when user queries in a detailed way (similar to the training examples). Therefore, rather than merely imitating human behaviors in a fixed corpus, task bots are desired to spontaneously learn from the interactions with real users, progressively improve and adapt after being deployed in dynamic and constantly changing environments.

There are several attempts to leverage human-bot interactions to improve task bots in changing environments. For example, Liu et al. (2018); Shah et al. (2018); Dai et al. (2020) propose to query humans for adequate feedback scores or annotations. However, it relies on human annotations or user feedback, which can be costly and sometimes users are unwilling to give any feedback. In addition, these works center on dialog policy optimization or retrieval-based task bots. Automatically adapting task bots to changing environments is imperative for end-to-end dialog model yet under-explored. Furthermore, these works usually omit task definition extensions.

In this paper, we propose SL-AGENT, a novel self-learning framework for building end-to-end task bots in a more realistic changing environment setting with minimum or zero human annotations. It consists of a neural dialog model and a pre-trained reward model, where the dialog model generates responses and the reward model judges the quality of agent responses. Specifically, we devise a data augmentation strategy to construct positive and negative examples based on the given dialog training corpus to endow the reward model with the capability to judge the quality of responses for unlabeled human-bot dialog logs. The bot (including dialog model and reward model) is first trained with the same available training data, then deployed to converse with real users and collect human-bot dialog logs. After that, as shown in the lower part of Figure 1, the bot is refined with the unlabeled human-bot dialog logs via reinforcement learning, where the response quality is judged by the reward model. In this way, the bot can automatically adapt to unseen user behaviors, without extra human annotations. Regarding the problem of extensions in task definitions, machine teaching is utilized to correct representative failed dialogs with minimum human annotations to provide necessary instructions on how to handle new functions. After that, the bot quickly adapts to new functions through the self-learning procedure.

Our contributions are summarized as below:

- We propose a new research problem *i.e.,* how to enable task bots to automatically adapt themselves to changing environments by learning from interactions with minimum or zero human annotations.
- We propose a novel self-learning framework SL-AGENT that equips with a pre-trained reward model trained by the devised data-augmentation strategy to build generative end-to-end task bots in a realistic changing environment setting, with minimum or zero human annotations.
- We conduct comprehensive experiments on four datasets to demonstrate the effectiveness of SL-AGENT for enabling automatic adaptation to changing environments by learning from the unlabeled human-bot dialog logs using both automatic and human evaluations.

## 2 Related Work

**RL for Dialog Policy Learning** Reinforcement learning has been widely applied to dialog systems for policy optimization. Young et al. (2013); Peng et al. (2018, 2017); Liu and Lane (2017); Gasic et al. (2014); Tseng et al. (2021) formulate dialog policy learning as a sequential problem and use REINFORCE (Williams, 1992) and/or Q-learning (Watkins and Dayan, 1992) to optimize the dialog policy. SL-AGENT utilizes a similar REINFORCE algorithm but focuses on generative end-to-end optimization.

**Adapting to Changing Environments for Dialog Systems** Several attempts have been made to deal with changing environments after deployment. Rajendran et al. (2019); Dai et al. (2020) propose to learn from the human-bot interactions but requires lots of human corrections. Shah et al. (2018); Liu et al. (2018); Gašić et al. (2011); Gasic et al. (2014) propose to learn from human-bot interactions via reinforcement learning based on the queried human feedback scores after each dialog. To reduce the efforts of querying humans, Su et al. (2016)

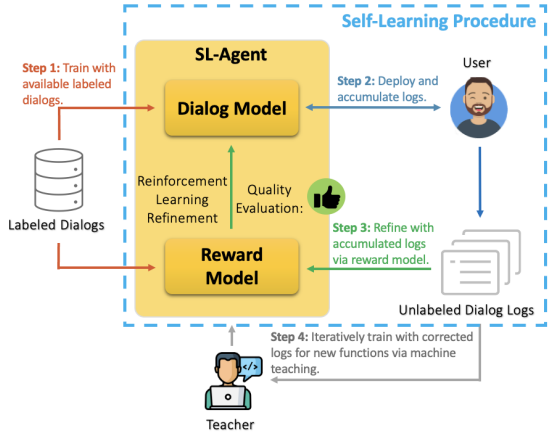



Figure 2: Training pipeline of the proposed SL-AGENT.

introduces a session-level Bi-LSTM reward model trained with extra pre-collected classification corpus to predict the task success of each dialog. Nevertheless, session-level reward model may underestimate the quality of responses in single dialog turns. Different from the works mentioned above, SL-AGENT leverages a turn-level pre-trained reward model built on the given dialog corpus using the devised data augmentation approach and focuses on generative end-to-end dialog systems. Another line of research is using data-augmentation methods to generate diverse user behaviors during the training stage (Gao et al., 2020; Li et al., 2020b). Additionally, Madotto et al. (2020); Liu et al. (2021) continually collect extra labeled data to train task bots but aim to overcome the catastrophic forgetting problem, which is a different research topic (*i.e.,* continual learning) from our paper.

# 3 SL-AGENT

## 3.1 Overview

As depicted in Figure 2, SL-AGENT contains two components: (i) a dialog model for generating responses (Section 3.2); (ii) a pre-trained reward model for judging the quality of agent responses and outputting reward scores to guide the refinement of the dialog model (Section 3.3). Specifically, SL-AGENT operates in the following steps: (i) First, the bot (both dialog model and pre-trained reward model) is fine-tuned with the same available annotated task-specific dialogs. (ii) Then, the bot is deployed online to converse with users and accumulate unlabeled human-bot dialog logs. (iii) Next, the dialog model is refined with these human-bot dialog logs via reinforcement learning, using the reward scores from the reward model (Section 3.4). (*iv*) For task definition extensions, machine teaching is utilized to correct representative failed dialogs to provide instructions on how to handle new functions (Section 3.5). After that, the bot further improves through the self-learning procedure.

## 3.2 Dialog Model

SL-AGENT is a general framework that is compatible with any generative end-to-end dialog models (Peng et al., 2020a; Ham et al., 2020; Hosseini-Asl et al., 2020). In this paper, we employ SOLOIST (Peng et al., 2020a), a pre-trained end-to-end dialog model, resulting in an agent termed SL-SOLOIST[4].

We briefly review SOLOIST for completeness. SOLOIST formulates the end-to-end dialog generation as a sequence generation problem, by sequentially concatenating the inputs and outputs of 4 dialog modules (*i.e.,* NLU, DST, POL, NLG) in a typical dialog system. Each dialog turn is represented as:

$$\boldsymbol{x} = (\boldsymbol{s}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{r}), \tag{1}$$

where $\boldsymbol{s}$ is the entire dialog history, $\boldsymbol{b}$ is the annotated belief state, $\boldsymbol{c}$ refers to DB state fetched from database, and $\boldsymbol{r}$ is the delexicalized agent response. SOLOIST employs a Transformer-based model with parameters $\boldsymbol{\theta_D}$ to characterize the sequence generation probability $p_{\boldsymbol{\theta_D}}(\boldsymbol{x})$. Initialized with GPT-2 (Radford et al., 2019), the model is pre-trained on large-scale annotated dialog corpora, and then fine-tuned with limited task-specific dialogs.

**Synthetic Dialog Construction.** To identify user behaviors with unseen slot values, we propose to synthesize dialog examples by exhausting database (DB) values and substitute corresponding slot values of in the training set. Specifically, for each dialog turn $\boldsymbol{x}$, we replace slot values in the utterances and user goal with corresponding new values of the randomly sampled DB entry.

## 3.3 Reward Model

The human-bot dialog logs accumulated after deployment may contain previously unseen user behaviors with unseen language patterns and unknown user goals. To enable the dialog model to identify these new types of user inputs to which the previously trained system cannot respond appropriately, we propose a reward model that judges the

[4]In this paper, SL-AGENT refers to the proposed framework and SL-SOLOIST is an instance of it, which utilizes SOLOIST as its dialog model.

quality of an agent response through a reward score (a positive reward for an appropriate response, a negative reward for an inappropriate response).

We formulate the quality evaluation problem as a binary classification task. Dialog responses are jointly determined by the dialog history, generated belief state, and fetched DB state. Therefore, given the training data $\mathcal{D}$ (annotated with belief states and DB states), we build a turn-level reward model $R$, which is parameterized by a Transformer $\boldsymbol{\theta_R}$ with the input dialog turn sequence $\boldsymbol{x}$, defined as Equation 1 to characterize the classification probability: $p_{\boldsymbol{\theta_R}}(\boldsymbol{x}) = p_{\boldsymbol{\theta_R}}(\boldsymbol{s}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{r})$.

The reward model $R$ is trained using contrastive objective to discriminate between an appropriate response (*i.e.,* positive example $\boldsymbol{x}$) and an inappropriate response (*i.e.,* negative example $\hat{\boldsymbol{x}}$), given the dialog history. Specifically, for each dialog turn, we construct several positive examples $\{\boldsymbol{x}_m\}_{m=1}^{M}$ and negative examples $\{\hat{\boldsymbol{x}}_{\boldsymbol{n}}\}_{n=1}^{N}$ based on the sequence $\boldsymbol{x}$, to add the relevance of real-world scenarios and endow the reward model with the capability of evaluating the response quality. Then we apply a binary classifier on top of the output sequence representation from the Transformer to discriminate between a positive example $\boldsymbol{x}$ ($y = 1$) and a negative example $\hat{\boldsymbol{x}}$ ($y = 0$). The training objective for a single example in the training set $\mathcal{D}$ is defined as:

$$\begin{aligned}\mathcal{L}_{\boldsymbol{\theta_R}} =& y \sum_{m=1}^{\mathcal{M}} \log\left(p_{\boldsymbol{\theta_R}}(\boldsymbol{x}_m)\right) \\ &+(1-y) \sum_{n=1}^{\mathcal{N}} \log\left(1 - p_{\boldsymbol{\theta_R}}\left(\hat{\boldsymbol{x}}_{\boldsymbol{n}}\right)\right),\end{aligned} \tag{2}$$

**Positive Examples.** For each dialog turn, we consider two kinds of user utterances: $(i)$ the original user utterance in the training set $\mathcal{D}$, to identify the appropriate response to the user behavior; $(ii)$ the paraphrased user utterances generated based on the original user utterance using back translation (Edunov et al., 2018), to enhance the ability of reward model for identifying user behaviors with diverse language patterns.

**Negative Examples.** Based on the analysis on 200 human-bot dialog logs collected from the evaluation platform of DSTC8 Track 1 challenge (Li et al., 2020a)[5], we summarize 5 types of dialog turns that have inappropriate responses (in Appendix J). Then, for each dialog turn in the training data $\mathcal{D}$, we construct negative examples $\hat{\boldsymbol{x}}$ (in brackets) according to these 5 types:

- **Repetition** The dialog model failed to understand the user's repeated query and generated the same response twice. (Repeating the response from the previous turn.)
- **Inconsistency** The dialog model generated an incoherent response. (Randomly sampling a response from the dataset $\mathcal{D}$ to replace the original response .)
- **Partial Information** The dialog model partially understood user request and answered incompletely. (For those user utterances with multiple slots request, randomly dropping a slot answer in the original response.)
- **Non-fluency** The dialog model generated a non-fluent response. (Randomly repeating some word tokens in the original response.)
- **Misunderstanding** The dialog model generated the incoherent belief state and response. (Randomly sampling a belief state and response from the dataset $\mathcal{D}$ to replace the original belief state and response.)

To boost the model performance with limited annotated task-specific corpora, we propose to follow the pre-training and fine-tuning paradigm to build the reward model, *i.e.,* pre-train the reward model using large-scale annotated heterogeneous dialog corpora, then fine-tune the pre-trained reward model with annotated task-specific data using the same training objective. The pre-training corpora is Schema dataset (Rastogi et al., 2019).

### 3.4 Refine with Reinforcement Learning

The interactions between the agent and users can be modeled as a sequential decision problem. As such, the dialog model can be refined via the REINFORCE algorithm (Williams, 1992). The policy is the trained dialog model $p_{\boldsymbol{\theta_D}}(\boldsymbol{x})$, the initial state is the dialog history $\boldsymbol{s}$, and the action space corresponds to the vocabulary set $\mathcal{V}$. The reward perceived by the dialog model is $R\left(\boldsymbol{s}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{r}\right)$ from the reward model. The parameters $\boldsymbol{\theta_D}$ are updated by maximizing the cumulative reward score. The refining procedure is described in detail as follows:

For each RL episode, we randomly sample a dialog turn with dialog history and delexicalized

[5]These human-bot dialog logs contain the evaluation scores and comments from Amazon Mechanical Turks.

response. We run the dialog model to generate belief state $\hat{\boldsymbol{b}}$, based on the input dialog history sequence $\boldsymbol{s}$. At each time step $t$, we sample a token $\hat{b}_t$ according to the model distribution, where the logits' distribution of the model is first filtered using Nucleus (top-p) filtering (Holtzman et al., 2019), then redistributed via softmax function. Then we retrieve DB state $\hat{\boldsymbol{c}}$ from the database using $\hat{\boldsymbol{b}}$, and sample the delexicalized response sequence $\boldsymbol{r}$ following same sampling procedure, based on the token sequence $(\boldsymbol{s}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}})$. Note that the delexicalized response is given as part of the input. Then we feed the concatenation of dialog history $\boldsymbol{s}$, generated belief state $\hat{\boldsymbol{b}}$, retrieved DB state $\hat{\boldsymbol{c}}$ and the response $\boldsymbol{r}$, i.e. $(\boldsymbol{s}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}, \boldsymbol{r})$ into the reward model $p_{\boldsymbol{\theta_R}}(\boldsymbol{x})$ to obtain the reward score $R(\boldsymbol{s}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}, \boldsymbol{r})$. The positive reward is 1, negative reward is -1. The training objective for a single example is represented as:

$$\mathcal{L}_{\theta_D} = -\sum_{t=1}^{T_{\hat{b}}} \log p_{\boldsymbol{\theta_D}}\left(\hat{b}_t \mid \hat{b}_{<t}, \boldsymbol{s}\right) \times R(\boldsymbol{s}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}, \boldsymbol{r}) - \sum_{t=1}^{T_r} \log p_{\boldsymbol{\theta_D}}\left(r_t \mid r_{<t}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}, \boldsymbol{s}\right) \times R(\boldsymbol{s}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}, \boldsymbol{r}), \tag{3}$$

where the length of generated belief state and input delexicalized response are $\boldsymbol{T_{\hat{b}}}$, $\boldsymbol{T_r}$, respectively. Algorithm 1 (in Appendix A) summarizes the self-learning-based RL refining framework for refining the dialog model.

### 3.5 Minimum annotations via Machine Teaching

To handle the queries about new functions in additional dialog turns, we need to introduce new slot-value pairs, action templates, *etc.*(An example is in Appendix G.) Machine teaching is an efficient approach to training task bots (Simard et al., 2017; Williams and Liden, 2017). In this paper, we implement machine teaching via Conversational Learner (CL) (Shukla et al., 2020). The teaching process is conducted in three steps: $(i)$ The trained task bot is deployed online to fulfill the given goals by interacting with real users, leaving a handful of human-bot dialog logs. $(ii)$ Human experts select a few representative failed dialogs to construct training examples with new functions by adding new action templates, introducing new slot-value pairs, correcting inappropriate responses and annotations (*i.e.,* belief states). $(iii)$ The deployed task bot (*i.e.,* both dialog model and reward model) is trained on these training examples to handle new functions.

| Domain | Attraction | Train | Hotel | Restaurant |
|---|---|---|---|---|
| #Train | 50 | 50 | 50 | 50 |
| #Valid | 50 | 50 | 50 | 50 |
| #Test | 100 | 200 | 200 | 200 |

Table 1: Data statistics of four single-domain dialog datasets (Peng et al., 2020a; Budzianowski et al., 2018).

## 4 Experiments

### 4.1 Experimental Setup

We validate the efficiency and flexibility of proposed SL-AGENT on four different end-to-end dialog tasks using MultiWOZ single-domain dialog datasets (Budzianowski et al., 2018), reorganized by Peng et al. (2020a). Data statistics are shown in Table 1. Based on above datasets, we construct two settings to represent the changing environments – **Setting I** for unseen user behaviors and **Setting II** for task definition extensions.

**Implementation Details.** To implement the proposed reward model, we conduct experiments with several Transformer-based models and GPT-2 (Radford et al., 2019) (enhanced with auxiliary generation task) shows better performance than others. Therefore, we implement proposed reward model using GPT-2-117M and the multi-task training objective. Full details are in Appendix B.

**Automatic Evaluation Metrics.** We report the results using the same automatic evaluation metrics following Budzianowski et al. (2018): $(i)$ `Inform`(%) evaluates whether the agent returns an appropriate entity. $(ii)$ `Success`(%) judges whether the agent correctly answers all requested attributes. $(iii)$ `BLEU`(%) measures the word overlap of the generated response against human response. $(iv)$ `Combined`(%) assesses the overall quality, which is defined as: `Combined` = (`Inform` + `Success`) $\times$ 0.5 + `BLEU`.

**Human Evaluation Metrics.** Following the same evaluation protocol in the DSTC9 Track 1 challenge (Gunasekara et al., 2020), we conduct human evaluations to judge the agent quality. For each dialog session, Amazon Mechanic Turkers are presented with a goal and instructions, then they are required to converse with agent to achieve the goal via natural language. At the end of each dialog session, Turks are required to assess the overall dialog quality using the following five metrics: $(i)$ `Success w/o g`(%) judges whether the agent completes the task. $(ii)$ `Success w/ g`(%)

| Model | Attraction | | | Train | | | Hotel | | | Restaurant | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Inform | Success | BLEU | Inform | Success | BLEU | Inform | Success | BLEU | Inform | Success | BLEU |
| SOLOIST$_5$ | 27.00 | 14.00 | 4.07 | 72.73 | 32.32 | 5.43 | 25.00 | 3.50 | 2.93 | 26.50 | 2.00 | 4.71 |
| SOLOIST$_S$ | 60.00 | 33.00 | 8.14 | 73.74 | 54.55 | 6.94 | 56.00 | 29.50 | 7.05 | 62.50 | 41.50 | 7.33 |
| SOLOIST+PARG | 60.00 | 32.00 | 8.83 | 75.25 | 56.06 | 8.45 | 58.00 | 29.00 | 7.71 | 64.00 | 42.00 | 9.17 |
| SOLOIST-OA | 61.00 | 36.00 | 8.66 | 74.75 | 55.05 | 7.58 | 56.50 | 29.00 | 7.14 | 64.50 | 42.50 | 8.56 |
| SL-SOLOIST | **64.00** | **40.00** | **8.99** | **75.76** | **61.62** | **10.97** | **60.50** | **39.50** | **8.34** | **75.00** | **44.50** | **10.60** |
| SOLOIST-TH | 66.00 | 41.00 | 9.01 | 77.27 | 62.87 | 10.70 | 60.00 | 42.50 | 9.82 | 70.50 | 46.00 | 11.76 |
| SOLOIST$_{50}$ | 86.00 | 65.00 | 12.90 | 80.81 | 64.65 | 9.96 | 74.50 | 43.50 | 8.12 | 81.00 | 55.50 | 12.80 |

Table 2: End-to-end evaluation results on four tasks. The forth to sixth rows indicate the results of refining with 45 simulated (unlabeled) human-bot dialog logs, based on SOLOIST$_S$. SOLOIST$_{50}$ is quoted from Peng et al. (2020a). (SL-SOLOIST significantly outperforms all baselines in mean with p<0.01 based on Combined.)

| Model | Attraction | | | Train | | | Hotel | | | Restaurant | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Inform | Success | BLEU | Inform | Success | BLEU | Inform | Success | BLEU | Inform | Success | BLEU |
| SOLOIST$_S$ | 60.00 | 33.00 | 8.14 | 73.74 | 54.55 | 6.94 | 56.00 | 29.50 | 7.05 | 62.50 | 41.50 | 7.33 |
| SOLOIST-OA | 63.00 | 34.00 | 8.66 | 77.78 | 55.05 | 8.13 | 58.50 | 30.00 | 7.08 | 63.00 | 42.00 | 10.03 |
| SL-SOLOIST | **70.00** | **36.00** | **8.68** | **78.28** | **60.10** | **9.06** | **62.00** | **33.50** | **7.39** | **70.00** | **45.00** | **10.93** |
| SOLOIST-TH | 68.00 | 40.00 | 9.01 | 76.77 | 62.63 | 9.55 | 62.50 | 35.50 | 7.83 | 70.50 | 47.50 | 11.36 |

Table 3: Automatic evaluation results on four tasks in Real-Scenario Setting. The first row refers to previously reported SOLOIST$_S$. The last three rows refer to refining with 30 real (unlabeled) human-bot dialog logs based on SOLOIST$_S$. (SL-SOLOIST significantly outperforms all baselines in mean with p<0.01 based on Combined.)

judges whether the agent completes the task and provides matched slot values against the database record. (*iii*) `Understanding`(1-5) measures the understanding correctness of user utterances. (*iv*) `Appropriateness`(1-5) indicates the appropriateness, naturalness, and fluency of an agent response. (*v*) `Turns` reports the average number of dialog turns for successful dialog sessions.

**Compared Methods.** To demonstrate the effectiveness of the proposed reward model in SL-AGENT, we use SOLOIST as the dialog model to compare the performance of different methods.

- SOLOIST$_5$ is trained with 5 labeled dialogs, randomly sampled from the train set.
- SOLOIST$_S$ is trained using synthetic dialogs constructed from the 5 labeled dialogs used for training SOLOIST$_5$.
- SOLOIST+PARG is trained on SOLOIST$_S$ with paraphrased dialogs (Gao et al., 2020; Edunov et al., 2018) constructed from the 5 labeled dialogs, *i.e.,* data-augmentation baseline for adapting to unseen user behaviors.
- SOLOIST-OA is refined with unlabeled human-bot dialog logs based on SOLOIST$_S$ using the session-level reward of task success from online activate reward model (trained using the same 5 labeled dialogs as SOLOIST$_5$) and partially queried session-level human feedback score (Su et al., 2016).
- SL-SOLOIST (Ours) is refined with unlabeled human-bot dialog logs based on SOLOIST$_S$ using the pre-trained reward model in SL-AGENT, which is fine-tuned using the same 5 labeled dialogs as SOLOIST$_5$. Machine teaching is not utilized by now[6].
- SOLOIST-TH is refined with unlabeled human-bot dialog logs based on SOLOIST$_S$ using queried turn-level human feedback score, which is an upper bound.
- SOLOIST$_{50}$ is trained with whole 50 labeled dialogs, which can be regarded as the result of sufficient human corrections, *i.e.,* the highest bound. (Details are shown in Appendix C.)

## 4.2 Results of Setting I - Unseen User Behaviors

**Simulation Evaluation Setup.** Deploying a trained agent to interact with real human users and collect dialog logs is labor-intensive and costly for experimental purposes. Hence, we construct a setting to simulate unseen user behaviors. We randomly sample 5 dialogs from the training set as labeled data to train a task bot (*i.e.,* both dialog model and reward model). Note that the remaining 45 dialogs contain unseen user behaviors with unseen

[6]To better demonstrate the self-learning capability of SL-AGENT, machine teaching is only used in the setting of task definition extensions. However, machine teaching can be optionally used to update the bot for better performance in the setting of unseen user behaviors.

language patterns and unknown user goals. Hence, it is applicable to simulate unseen user behaviors by modifying the remaining 45 dialogs as unlabeled imperfect human-bot dialog logs (through adding noise, *i.e.,* corrupting responses[7]). These 45 unlabeled human-bot dialog logs are further used for refining SOLOIST$_S$, resulting in SOLOIST-OA, SL-SOLOIST, SOLOIST-TH. This simulation setting allows us to perform a detailed analysis of the reward model in SL-AGENT without much cost and easily reproduce the experimental results.

**Simulation Evaluation Results.** The end-to-end evaluation results on four different tasks are presented in Table 2. SOLOIST$_S$ significantly outperforms SOLOIST$_5$ over all evaluation metrics on all tasks, which shows the effectiveness of the proposed synthetic dialog construction for identifying user behaviors with unseen slot values. SL-SOLOIST outperforms SOLOIST+PARG over all the metrics, which demonstrates the higher efficiency of directly learning from human-bot dialog logs. We observe that SL-SOLOIST outperforms SOLOIST-OA by a large margin, and achieves comparable performance with SOLOIST-TH (refining with turn-level human feedback score, *i.e.,* the upper bound). This shows the strong capability of the turn-level pre-trained reward model in SL-AGENT for predicting the quality of responses. We conjecture that our proposed reward model trained with the proposed data-augmentation strategy is more robust to unseen user behaviors and thus ports richer useful information to dialog models. The results verify the vast potential of the proposed SL-AGENT, allowing the bot to automatically adapt to unseen user behaviors without extra human annotations. Results of further policy improvement are shown in Appendix E.

**Real-Scenario Evaluation Setup.** Simulation setting allows effortless experimental studies to validate the effectiveness of the reward model in SL-AGENT. However, the results are likely biased. Therefore, in the real-scenario setting, we deploy SOLOIST$_S$ online and recruit human users to converse with it. We collect 30 real (unlabeled) human-bot dialog logs to refine SOLOIST$_S$, resulting in the agent SOLOIST-OA, SL-SOLOIST, SOLOIST-TH.

**Real-Scenario Evaluation Results.** The evaluation results on four tasks are shown in Table 3.

[7]Note that the associated labels of belief states are not used. Construction details are in Appendix D.

| Model | Restaurant-Ext | | | |
|---|---|---|---|---|
| | Inform | Success | BLEU | Combined |
| SOLOIST$_S$ | 54.00 | 0.00 | 6.42 | 33.42 |
| SOLOIST$_S$+TEACH | 64.00 | 18.00 | 9.34 | 50.34 |
| SL-SOLOIST+TEACH | **68.00** | **24.00** | **11.76** | **57.76** |
| SOLOIST-TH+TEACH | 68.50 | 26.00 | 11.88 | 59.13 |

Table 4: Automatic evaluation results on task definition extensions. (Difference in mean is significant with p<0.01 based on Combined.)

We observe that SL-SOLOIST refined using the reward model in SL-AGENT outperforms other methods over all evaluation metrics on all tasks. Furthermore, SL-SOLOIST achieves comparable performance with SOLOIST-TH, even achieves better performance on certain metrics. We conclude that the results of real-scenario evaluation and simulation evaluation are consistent, confirming that SL-SOLOIST enables effective self-learning after deployment by learning from interactions.

### 4.3 Results of Setting II – Task Definition Extensions

**Setup.** We follow the domain extension experiment setting in Lipton et al. (2018) to assess the ability of SL-SOLOIST to quickly handle task definition extensions. We extend existing `Restaurant`, denoted as `Restaurant-Ext`, with additional functions by introducing 4 new slots, *i.e., [restaurant_dish]*, *[value_price]*, *[start_time]*, *[end_time]* in added dialog turns (in Appendix G), and corresponding values for each DB entry (in Appendix H). The first slot is about the restaurant's signature dish, and the last three are related to delivery service. We leverage Conversational Learner (CL) (Shukla et al., 2020), a practical machine teaching tool, to visualize and select dialogs for constructing training examples on the `Restaurant-Ext` domain by providing corrections and introducing new slots. Finally, 10 examples are obtained through machine teaching for training, 50 for validating and 50 for testing. We fine-tune the dialog model SOLOIST$_S$ and the previously trained reward model[8], using 10 corrected dialogs, resulting the agent denoted as SOLOIST$_S$+TEACH. Then, SOLOIST$_S$+TEACH is deployed to converse with real human to collect 20 real (unlabeled) human-bot dialog logs, which are then used to refine itself, resulting in SL-SOLOIST+TEACH. To better show the effective-

[8]The reward model used for obtaining SL-SOLOIST in the Table 2. It is trained with 5 labeled dialogs in the train set.

| Model | Restaurant | | | | |
|---|---|---|---|---|---|
| | SR w/o g | SR w/ g | Under. | Appr. | Turns |
| SOLOIST$_S$ | 31.82 | 29.54 | 3.86 | 4.13 | 10.00 |
| SOLOIST-OA | 33.42 | 30.86 | 3.89 | 4.12 | 9.97 |
| SL-SOLOIST | **43.10** | **36.21** | **3.97** | **4.13** | **9.89** |

Table 5: Human evaluation results. SR w/o g: Success rate without grounding, SR w/ g: Success rate with grounding, Under.: Understanding score, Appr.: Appropriateness score.

ness of the reward model in SL-AGENT, we also report the result of SOLOIST-TH+TEACH, which is refined using the turn-level human feedback score.

**Results.** The evaluation results are presented in Table 4. We observe that SOLOIST$_S$ has zero success rate, which is predictable as it does not have any knowledge of the new functions. SOLOIST$_S$+TEACH outperforms the baseline by 17 points in terms of Combined score, which exhibits the effectiveness of machine teaching for handling new functions. SL-SOLOIST+TEACH lifts the Combined score by approximately 7 points, achieving comparable performance with SOLOIST-TH+TEACH. The results show that SL-SOLOIST+TEACH can adapt to new tasks and continually improve itself by automatically learning from the interactions, revealing, with minimum annotations from machine teaching, SL-AGENT enables flexible adaptations to new functions.

### 4.4 Interactive Human Evaluation

**Setup.** We conduct human evaluations to evaluate the performance of SOLOIST$_S$, SOLOIST-OA, SL-SOLOIST interacting with human users, following the evaluation protocol in DSTC9 track 1 challenge (Gunasekara et al., 2020), with 100 Turkers involved and 100 dialogs gathered for analysis, respectively.

**Results.** The human evaluation results on Restaurant domain are presented in Table 5. The results show that SL-SOLOIST outperforms SOLOIST$_S$, SOLOIST-OA over all the metrics, which are consistent with the automatic evaluation results. The significant improvement on two success rate metrics, especially success rate with grounding, verifies the effectiveness of the reward model in SL-AGENT for refining the dialog agent after deployment without additional human annotations, as it more adequately reflects the system's capability of completing tasks in real scenarios. Two interactive examples are in Appendix F.

| Reward model | Restaurant | | | |
|---|---|---|---|---|
| | Inform | Success | BLEU | Combined |
| GPT-2 | 67.00 | 41.50 | 9.30 | 63.55 |
| BERT | 68.00 | 42.50 | 9.55 | 64.80 |
| BERT-Large | 66.00 | 44.00 | 11.09 | 66.09 |
| RoBERTa | 72.00 | 45.00 | 9.23 | 67.73 |
| RoBERTa-Large | 69.50 | 46.50 | 10.20 | 68.20 |
| SL-SOLOIST | **75.00** | **44.50** | **10.60** | **70.35** |

Table 6: Ablation study results on using different PLMs for reward models. (Difference in mean is significant with p<0.01 based on Combined.)

### 4.5 Ablation Study

**Impact of different PLMs for reward models.** We conduct ablation studies on Restaurant domain to analyze the influence of choosing different PLMs and multi-task training objective on the reward model. We choose several popular PLMs including BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019). Note that all the models share the same pre-training and fine-tuning procedure, except that BERT and RoBERTa are trained with quality prediction task while SL-SOLOIST is optimized using multi-task learning. We show in Table 6 that RoBERTa performs better than BERT. GPT-2 (on which SL-SOLOIST is built) trained with single quality prediction task, yields significantly worse performance than other methods. We speculate that bidirectional Transformer encoder enables BERT and RoBERTa to capture richer context information. SL-SOLOIST achieves consistent performance improvements over all the metrics, showing the effectiveness of multi-task learning for the reward model.

## 5 Conclusion

In this paper, we propose a new research problem *i.e.,* how to enable task bots to automatically adapt themselves to changing environments by learning from interactions with minimum or zero human annotations. In addition, we propose SL-AGENT, a novel self-learning framework. We verify its effectiveness on automatically adapting to changing environments on four dialog tasks by learning from the unlabeled human-bot dialog logs via reinforcement learning with an incorporated pre-trained reward model. As for future work, there are more ways that a task bot could learn to improve itself, *e.g.,* during machine teaching, human experts could provide not only correct labels but also feedback in natural language. We leave the theme of effective machine teaching to future work.

## 6 Ethical Considerations

During the collection, annotation and evaluation procedure of the human-bot dialog logs, all involved Amazon Mechanic Turkers and human annotators have been informed of the research purpose in advance, and any of their privacy will not be disclosed or violated during the research period. All other used datasets are open-sourced datasets. In summary, we abide by all research ethics.

## 7 Acknowledgements

This research is affiliated with the CUHK MoE-Microsoft Key Laboratory for Human-centric Interface Technologies. The project is partially sponsored by a grant from the HKSAR Research Grants Council General Research Fund (project number 14207619). In addition, we would like to thank Yifei Yuan, Kun Zhang, Kun Li and Jingyan Zhou in particular for their insightful comments and persevering support.

## References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz–a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.

Yinpei Dai, Hangyu Li, Chengguang Tang, Yongbin Li, Jian Sun, and Xiaodan Zhu. 2020. Learning low-resource end-to-end goal-oriented dialog for fast and reliable system deployment. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 609–618.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*.

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.

Silin Gao, Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020. Paraphrase augmented task-oriented dialog generation. *arXiv preprint arXiv:2004.07462*.

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 *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 312–317. IEEE.

Milica Gasic, Dongho Kim, Pirros Tsiakoulis, Catherine Breslin, Matthew Henderson, Martin Szummer, Blaise Thomson, and Steve J. Young. 2014. Incremental on-line adaptation of pomdp-based dialogue managers to extended domains. In *INTERSPEECH*.

Chulaka Gunasekara, Seokhwan Kim, Luis Fernando D'Haro, Abhinav Rastogi, Yun-Nung Chen, Mihail Eric, Behnam Hedayatnia, Karthik Gopalakrishnan, Yang Liu, Chao-Wei Huang, et al. 2020. Overview of the ninth dialog system technology challenge: Dstc9. *arXiv preprint arXiv:2011.06486*.

Donghoon Ham, Jeong-Gwan Lee, Youngsoo Jang, and Kee-Eung Kim. 2020. End-to-end neural pipeline for goal-oriented dialogue systems using gpt-2. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 583–592.

Braden Hancock, Antoine Bordes, Pierre-Emmanuel Mazare, and Jason Weston. 2019. Learning from dialogue after deployment: Feed yourself, chatbot! *arXiv preprint arXiv:1901.05415*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *arXiv preprint arXiv:2005.00796*.

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

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 (Volume 1: Long Papers)*, pages 1437–1447.

Jinchao Li, Baolin Peng, Sungjin Lee, Jianfeng Gao, Ryuichi Takanobu, Qi Zhu, Minlie Huang, Hannes Schulz, Adam Atkinson, and Mahmoud Adada. 2020a. Results of the multi-domain task-completion dialog challenge. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence, Eighth Dialog System Technology Challenge Workshop*, volume 7.

Shiyang Li, Semih Yavuz, Kazuma Hashimoto, Jia Li, Tong Niu, Nazneen Rajani, Xifeng Yan, Yingbo Zhou, and Caiming Xiong. 2020b. Coco: Controllable counterfactuals for evaluating dialogue state trackers. *arXiv preprint arXiv:2010.12850*.

Zachary Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. 2018. Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Bing Liu and Ian Lane. 2017. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 482–489. IEEE.

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*.

Qingbin Liu, Pengfei Cao, Cao Liu, Jiansong Chen, Xunliang Cai, Fan Yang, Shizhu He, Kang Liu, and Jun Zhao. 2021. Domain-lifelong learning for dialogue state tracking via knowledge preservation networks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2301–2311.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul Crook, Bing Liu, Zhou Yu, Eunjoon Cho, and Zhiguang Wang. 2020. Continual learning in task-oriented dialogue systems. *arXiv preprint arXiv:2012.15504*.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2020a. Soloist: Few-shot task-oriented dialog with a single pretrained auto-regressive model. *arXiv preprint arXiv:2005.05298*.

Baolin Peng, Chunyuan Li, Zhu Zhang, Chenguang Zhu, Jinchao Li, and Jianfeng Gao. 2020b. Raddle: An evaluation benchmark and analysis platform for robust task-oriented dialog systems. *arXiv preprint arXiv:2012.14666*.

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

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. *arXiv preprint arXiv:1704.03084*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Janarthanan Rajendran, Jatin Ganhotra, and Lazaros C Polymenakos. 2019. Learning end-to-end goal-oriented dialog with maximal user task success and minimal human agent use. *Transactions of the Association for Computational Linguistics*, 7:375–386.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2019. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *arXiv preprint arXiv:1909.05855*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Pararth Shah, Dilek Hakkani-Tur, Bing Liu, and Gokhan Tur. 2018. Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 41–51.

Swadheen Shukla, Lars Liden, Shahin Shayandeh, Eslam Kamal, Jinchao Li, Matt Mazzola, Thomas Park, Baolin Peng, and Jianfeng Gao. 2020. Conversation learner–a machine teaching tool for building dialog managers for task-oriented dialog systems. *arXiv preprint arXiv:2004.04305*.

Patrice Y Simard, Saleema Amershi, David M Chickering, Alicia Edelman Pelton, Soroush Ghorashi, Christopher Meek, Gonzalo Ramos, Jina Suh, Johan Verwey, Mo Wang, et al. 2017. Machine teaching: A new paradigm for building machine learning systems. *arXiv preprint arXiv:1707.06742*.

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*.

Bo-Hsiang Tseng, Yinpei Dai, Florian Kreyssig, and Bill Byrne. 2021. Transferable dialogue systems and user simulators. *arXiv preprint arXiv:2107.11904*.

Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning*, 8(3-4):279–292.

Jason D Williams and Lars Liden. 2017. Demonstration of interactive teaching for end-to-end dialog control with hybrid code networks. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 82–85.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam

Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

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.

Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020. Task-oriented dialog systems that consider multiple appropriate responses under the same context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9604–9611.

## A RL Refining Algorithm

**Algorithm 1** Self-learning-based RL refining framework.

**Input:**

Training examples $\mathcal{D}$ in the form of dialog turns;

Trained agent with dialog model $p_{\boldsymbol{\theta}_D}(\boldsymbol{x})$ and reward model $p_{\boldsymbol{\theta}_R}(\boldsymbol{x})$.

**Output:**

Refined agent with updated dialog model $p_{\boldsymbol{\theta}^*_D}$.

1: **while** not converged **do**
2: Randomly sample a dialog turn, i.e. token sequences of dialog history $\boldsymbol{s}$;
3: Run dialog model $p_{\boldsymbol{\theta}_D}$ on dialog history $\boldsymbol{x} = (\boldsymbol{s})$ to generate belief state $\hat{\boldsymbol{b}}$;
4: Retrieve DB state $\hat{\boldsymbol{c}}$ from a database using generated belief state $\hat{\boldsymbol{b}}$;
5: Sample corresponding response $\boldsymbol{r}$ based on dialog history $\boldsymbol{s}$, belief state $\hat{\boldsymbol{b}}$ and DB state $\hat{\boldsymbol{c}}$;
6: Use the reward model to predict the quality of the belief state and response with reward score, $R(\boldsymbol{s}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}, \boldsymbol{r})$;
7: Calculate the loss according to Equation 3;
8: Update the parameters of the dialog model, $\boldsymbol{\theta_D} \leftarrow \boldsymbol{\theta_D} + \alpha \nabla_{\boldsymbol{\theta}_D} \mathcal{L}_{\boldsymbol{\theta}_D}$.
9: **end while**

## B Implementation Details

User : I would like to find an expensive restaurant that severs Chinese food. System : Sure, which area do you prefer ? User : How about in the north part of town. [BOS] Belief State : Restaurant { pricerange = expensive, food = Chinese, area = north } [EOB] DB : Restaurant 2 match [EOD] System: The [restaurant_name] is a great [value_food] restaurant. Would you like to book a table there ? [EOS]

Figure 3: Illustration of the training example, *i.e.,* the processed dialog turn in the training data.

To construct training examples as shown in Figure 3, we tokenize the dialog turn sequence using byte pair encodings (Sennrich et al., 2015) and delexicalize responses by replacing slot values with corresponding special slot tokens (Lei et al., 2018). We conduct experiments with several Transformer-based models and GPT-2 (Radford et al., 2019) (enhanced with auxiliary generation tasks) shows

better performance than others. Therefore, we implement proposed reward model based on Huggingface Pytorch Transformer (Wolf et al., 2020) using GPT-2-117M. We pre-train reward model for 10 epochs using Schema dataset (Rastogi et al., 2019), which contains 22,825 dialogs in 17 domains. The reward model is pre-trained on two 24G Nvidia P40 with a mini-batch of 8 and learning rate of 5e-5, using Adam optimizer (Kingma and Ba, 2014), where the training examples are truncated or padded to the max length of 500.

We fine-tune the pre-trained reward model and dialog model (*i.e.,* pre-trained SOLOIST) for 20 epochs with limited number of labeled task-specific dialogs for new tasks. During refinement, top-p is selected as 0.5 for all models. We perform gradient clipping with the max norm as 1 for learning model parameters, with the batch size as 1 and learning rate as 5e-6. The dialog model is refined on a single 24G Nvidia P40 until converging on the validation set. During testing, Nucleus filtering is also used for decoding with top-p as 0.5.

## C Experimental Details

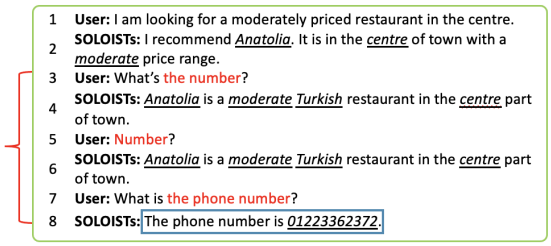
To demonstrate the effectiveness of SL-AGENT, we use SOLOIST as the dialog model to compare the performance of different methods, since existing state-of-the-art task-oriented dialog models share similar input-output pairs and training objectives as SOLOIST. (We report the results in mean of 5 runs with 5 different seeds.) $(i)$ To obtain SOLOIST$_S$, we implement the synthetic dialog construction method by exhausting DB values. For each dialog turn of the 5 labeled dialogs, we randomly sample five DB values from the database to replace the original slot values. $(ii)$ To obtain SOLOIST+PARG, we use the Transformer-based machine translation checkpoints (English-German, German-English) (Edunov et al., 2018) to generate 10 paraphrased user utterances for each dialog turn of the 5 labeled dialogs (based on the empirical analysis of translation quality). Then we use these annotated data (with paraphrased user utterances) to train SOLOIST$_S$ for obtaining SOLOIST+PARG. $(iii)$ To obtain SOLOIST-OA, we use the method described in Section 8 to construct successful dialogs and failed dialogs. For successful dialogs, we use the original 5 labeled dialogs, and the dialogs containing paraphrased user utterances. To construct the failed dialogs, we randomly select 2-3 dialog turns in each dialog and corrupt responses according to the negative example construction method in Section 3.3. Then we use these annotated dialogs to train the session-level reward model of (Su et al., 2016). When testing the performance in the simulated setting, we refine the SOLOIST$_S$ with fully correct dialogs and dialogs containing corrupted responses. To achieve better performance, we largely query for session-level human feedback score in both simulated setting and real-scenario setting.

| Model | Restaurant | | | |
|---|---|---|---|---|
| | Inform | Success | BLEU | Combined |
| SOLOIST$_S$ | 62.50 | 41.50 | 7.33 | 59.33 |
| SL-SOLOIST | 75.00 | 44.50 | 10.60 | 70.35 |
| SL-SOLOIST$_{+20}$ | **75.00** | **52.00** | **11.89** | **75.39** |

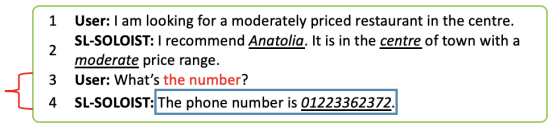
Table 7: End-to-end evaluation results of Policy Improvement in the Restaurant domain. SL-SOLOIST$_{+20}$ refer to continually refining with 20 real (unlabeled) human-bot dialogs based on SL-SOLOIST (reported in Table 2).

## D Simulated Human-Bot Corpora Construction

The unlabeled simulated human-bot corpora is constructed as follows: $(i)$ we remove belief state annotations; $(ii)$ we add negative examples by corrupting responses according to the negative example construction method in Section 3.3. We will release the simulated human-bot corpora for reproducible research. Note that directly replacing the belief states and responses with the generated ones is trivial. However, such approach cannot imitate realistic human-bot interactions. As the user utterances are strictly fixed, "users cannot react to the agent responses accordingly and appropriately". Therefore, we also conduct experiments through conversing with real users in the real-scenario setting and demonstrate the results in Table 3. Furthermore, building a user simulator is inapplicable in our changing environment setting. $(i)$ It is difficult to build reliable user simulators. Building agenda-based user simulators requires sophisticated human expertise for designing rules. $(ii)$ Building model-based user simulators requires sufficient labeled data. Furthermore, model-based user simulators merely imitate expert behaviors in the training corpus, cannot provide user behaviors that are unseen from task bots.




(a) An interactive example between user and SOLOISTs




(b) An interactive example between user and SL-SOLOIST

Figure 4: Two interactive examples. (a) An interactive example between user and SOLOIST$_S$. (b) An interactive example between user and SL-SOLOIST.

## E Policy Improvement

**Policy Improvement Setup.** To demonstrate the effectiveness of SL-AGENT for continually learning from collected human-bot dialog logs, we deploy SL-SOLOIST online and recruit human users to converse with it to achieve the assigned user goal. We collect 20 real human-bot dialog logs to refine SL-SOLOIST, resulting in the agent SL-SOLOIST$_{+20}$. (When refining the SL-SOLOIST, we do not use the knowledge about the user's goal. The response quality is judged by the reward model in SL-SOLOIST.)

**Policy Improvement Results.** The evaluation results on `Restaurant` are shown in Table 7. We observe that SL-SOLOIST$_{+20}$ refined with 20 real (unlabeled) human-bot dialogs outperforms SL-SOLOIST by approximately 5 points in terms of `Combined` score. We conclude that SL-SOLOIST enables continual self-learning after deployment by automatically learning from interactions.

## F Interactive Example

Figure 4 depicts two interactive examples where the same user interacts with SOLOIST$_S$ and SL-SOLOIST to complete the same task. We observe that, in the first four dialog turns, the two agents has the same performance and both correctly recommend a satisfied restaurant. However, as shown in Figure 4 (a), when user queries about the phone number ("what's the number?") in the fifth turn, SOLOIST$_S$ fails to understand user's intent and generates incoherent response, still trying to provide recommendation. The user has to continually query about phone number in the following consecutive turns. As demonstrated in Figure 4 (b), SL-SOLOIST correctly provides the phone number, when user first queries about it. Comparing the two examples, we show that SL-AGENT enables adapting to unseen user behaviors in an automatic way.

## G An Example of Task Definition Extensions

Figure 5 depicts an example of task definition extensions.

| t | Speaker | Utterance ($u_t$) |
|---|---|---|
| 1 | User | Hi, I'm looking for a place with Tuscan food in any price range. |
| 2 | System | I'm sorry, there is not a *Tuscan* restaurant listed. Would you care to try something else? |
| 3 | User | How about any Korean restaurants? |
| 4 | System | *Little Seoul* is a *Korean* restaurant. |
| 5 | User | Phone number please. |
| 6 | System | The phone number is *01223308681*. Is there anything else I can help you with? |
| 7 | User | Does the restaurant offer delivery service? How much does the delivery charge? |
| 8 | System | Yes, and the delivery fee is *4 pounds*. Would you like more information about the service? |
| 9 | User | No. Thank you, goodbye. |
| 10 | System | Thank you. Goodbye. |

Figure 5: An example of task definition extensions. Task bots need learn to provide information about the extended delivery service in additional dialog turns (in Red) as user requirements evolve.

## H An Example of `Restaurant-Ext` DB Entry

An example of `Restaurant-Ext` DB entry is shown in Figure 6.

## I Item Examples of the Input Dialog Turn Sequence

## J Negative Example Construction

```
{
  "address": "Finders Corner Newmarket Road",
  "area": "east",
  "food": "international",
  "id": "30650",
  "introduction": "",
  "location": [
    52.21768,
    0.224907
  ],
  "name": "the missing sock",
  "phone": "01223812660",
  "postcode": "cb259aq",
  "pricerange": "cheap",
  "type": "restaurant",
  "delivery fee": "5 pounds",
  "dish": "Greek Chicken Pasta",
  "start_time": "09:50",
  "end_time": "22:30"
},
```


Figure 6: An example of `Restaurant-Ext` DB entry. Newly added DB information about the extended function is in the red square.

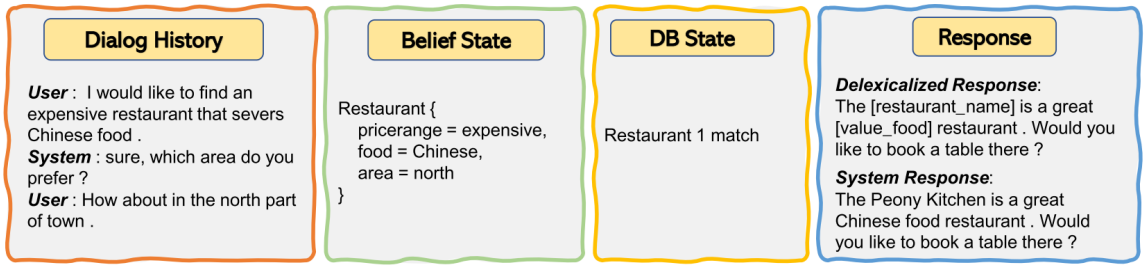



Figure 7: Item examples of the input dialog turn sequence for SOLOIST, cited from (Peng et al., 2020a).

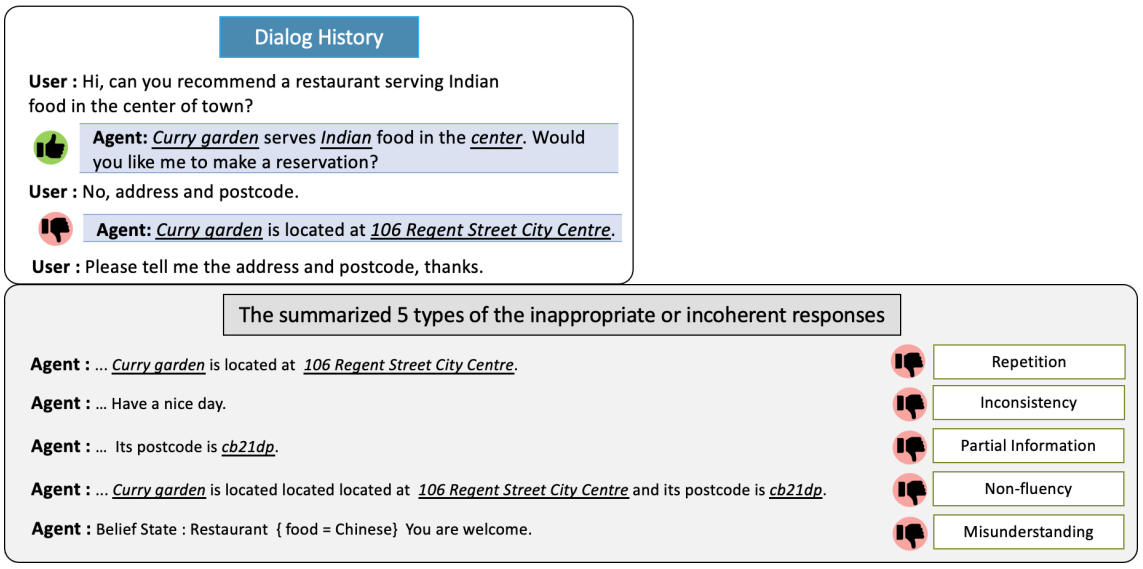



Figure 8: The summarized 5 types of dialog turns that have inappropriate or incoherent responses. (a) Dialog history (top). (b) 5 types of the inappropriate or incoherent responses (bottom).