Toward Self-Learning End-to-End Dialog Systems

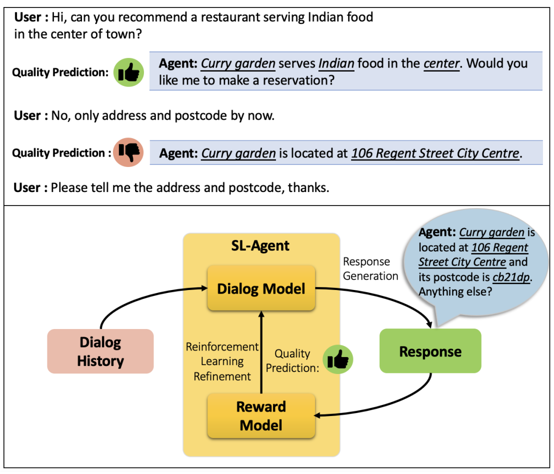
Xiaoying Zhang[1], **Baolin Peng**[2], **Jianfeng Gao**[2], **Helen Meng**[1]
[1]The Chinese University of Hong Kong, Hong Kong
[2]Microsoft Research, Redmond
{zhangxy, hmmeng}@se.cuhk.edu.hk
{bapeng, jfgao}@microsoft.com

Abstract

End-to-end task-oriented dialog systems often suffer from out-of-distribution (OOD) inputs after being deployed in dynamic, changing, and open environments. In this work, we propose SL-AGENT[1], a self-learning framework that combines supervised learning, reinforcement learning, and machine teaching for building end-to-end dialog systems in a more realistic changing environment setting. SL-AGENT consists of a dialog model and a pre-trained reward model to judge the quality of a system response. SL-AGENT enables dialog agents to automatically adapt to environments with user behavior changes by learning from human-bot interactions via reinforcement learning, with the incorporated pre-trained reward model. We validate SL-AGENT in four different dialog domains. Experimental results show the effectiveness of SL-AGENT for automatically adapting to changing environments using both automatic and human evaluations. Furthermore, experiments on a challenging domain extension setting demonstrate that SL-AGENT can effectively adapt to new tasks using limited human corrections provided via machine teaching. We will release code, data, and pre-trained models for further research.

1 Introduction

With the recent advances in neural approaches to conversational AI (Gao et al., 2018), the most common approach of building task-oriented dialog systems is to train neural models to imitate expert behaviors in large-scale corpora (Gao et al., 2018; Lei et al., 2018; Zhang et al., 2020). However, large-scale annotated corpora are rarely available for new tasks in real scenarios, due to the high cost of data collecting and labeling. Pre-training and fine-tuning paradigm that leverages Pre-trained Language Models (PLMs) has alleviated the data scarcity problem to a large extent, and becomes increasingly prevalent for building task-oriented dialog systems (Peng et al., 2020b; Wu et al., 2020; Peng et al., 2020a; Ham et al., 2020; Hosseini-Asl et al., 2020).


User : Hi, can you recommend a restaurant serving Indian food in the center of town?

Quality Prediction: 👍 **Agent:** *Curry garden* serves *Indian* food in the *center*. Would you like me to make a reservation?

User : No, only address and postcode by now.

Quality Prediction : 👎 **Agent:** *Curry garden* is located at *106 Regent Street City Centre*.

User : Please tell me the address and postcode, thanks.

SL-Agent
Dialog Model
Dialog History
Response Generation
Agent: Curry garden is located at 106 Regent Street City Centre and its postcode is cb21dp. Anything else?
Response
Reinforcement Learning Refinement
Quality Prediction:
Reward Model

Figure 1: A human-bot dialog example, containing a failed response related to user behavior changes. The bot fails to respond with postcode due to language variations at first, but gives the correct response in the following attempt. These human-bot interactions contain useful information from which the task bot can learn to improve its performance.

However, these data-driven approaches assume an independent and identically distributed (IID) data setting, *i.e.,* a static environment, and usually exhibit a tendency of failures when confronted with out-of-distribution (OOD) examples, *i.e.,* changing environments. In the context of task-oriented dialogs, changing environments are common and arise from (i) *user behavior changes* – a lot more variety of language/policy in the real world than pre-built training corpora; (ii) *task definition changes* – *e.g.,* dialog systems need to handle new functions (with new slots and values) as user and business requirements evolve. As shown in Figure 1, due to changes in user behaviors, the system fails to provide the postcode in the second response,

[1]SELF-LEARNING AGENT.

but gives the correct response in the following attempt (in the top-right corner of the second square). These human-bot interactions accumulated after deployment are usually abundant, task-specific, dynamic, but contain potentially useful information to improve the bot. We argue that task bots should not merely imitate expert behaviors in a corpus but also learn from the human-bot interactions accumulated after deployment to adapt to changing environments, requiring minimal to zero human annotations.

In this paper, we propose SL-AGENT, a novel self-learning framework for building task bots in a more realistic setting, which allows the bot to adapt to changing environments by learning from the human-bot interactions after deployment. SL-AGENT consists of a neural dialog model and a reward model, as illustrated in Figure 1, where the neural dialog model tracks belief states and generates responses, and the reward model judges the quality of an agent response from three perspectives: (*i*) fluency, (*ii*) success (*iii*) dialog flow consistency. The reward model is pre-trained using large-scale dialog corpora, based on a Transformer-based language model, to gain the capability of quick adaptation for judging the quality of a response on any new task. As depicted in Figure 2, SL-AGENT operates in the following steps: (*i*) First, the dialog model and pre-trained reward model are fine-tuned to new tasks with limited task-specific dialogs. (*ii*) Then, the bot interacts with users and collects human-bot dialogs. (*iii*) Next, the bot is refined with the collected human-bot dialogs using reinforcement learning, where the response quality being judged by the fine-tuned reward model. (*iv*) Finally, machine teaching is utilized to correct representative dialogs to update the bot or provide instructions on how to handle new functions or tasks.

We conduct comprehensive empirical studies on four different dialog domains with both automatic and human evaluation. Results demonstrate that SL-AGENT is an effective self-learning framework, enabling task bots to automatically adapt to environments with user behavior changes by learning from human-bot interactions accumulated after deployment[2]. In a challenging domain extension setting, SL-AGENT effectively adapts to a new environment using limited human corrections provided via machine teaching. The promising results exhibit the remarkable self-learning ability of SL-AGENT framework for handling user behavior changes, and its lifelong learning capability for handling task definition changes, without substantial amounts of extra data, after deployment.

In summary, the main contributions of our work are two-fold:

- We propose a novel self-learning framework SL-AGENT, that combines supervised learning, reinforcement learning, and machine teaching, which allows task bots to effectively adapt to changing environments that may differ dramatically from the setting in which the bots are built.
- We demonstrate that SL-AGENT enables a deployed bot to automatically adapt to environments with user behavior changes by learning from human-bot interactions, based on four well-studied dialog tasks using both automatic and human evaluations.

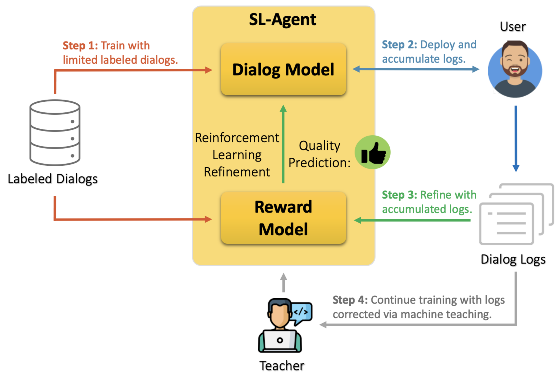

SL-Agent
Step 1: Train with limited labeled dialogs.
Dialog Model
Step 2: Deploy and accumulate logs.
User
Labeled Dialogs
Reinforcement Learning Refinement
Quality Prediction:
Step 3: Refine with accumulated logs.
Reward Model
Dialog Logs
Step 4: Continue training with logs corrected via machine teaching.
Teacher

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

2 Related Work

RL for Dialog Policy Learning RL has been largely implemented in dialog systems for policy optimization. Young et al. (2013); Peng et al. (2018, 2017); Liu and Lane (2018, 2017) 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 fully end-to-end optimization. Therefore, the aforementioned approaches can be directly compared within this paper.

[2]We refer to this automatic learning process as self-learning in the paper.

Pre-trained Models for Dialog Systems Pre-trained language models have been adopted to develop task-oriented dialog systems and achieve state-of-the-art performance on various tasks (Wu et al., 2020; Peng et al., 2020a; Coope et al., 2020; Henderson et al., 2019; Ham et al., 2020; Hosseini-Asl et al., 2020). However, all of these works assume a static environment. Such assumption is generally violated in practice after dialog systems being deployed. Our method enables dialog models to automatically adapt to changing environments by learning from human-bot interactions.

Handling OOD examples for Dialog Systems Several attempts have been made to handle OOD inputs from changing environments (Shah et al., 2018; Su et al., 2016; Gašić et al., 2011; Gašić and Young, 2013; Liu et al., 2018; Rajendran et al., 2019; Dai et al., 2020). Shah et al. (2018); Gašić et al. (2011); Gašić and Young (2013); Liu et al. (2018) proposed to learn from human-bot interactions with reinforcement learning but relies on the queried human feedback score after each session. Su et al. (2016) introduced a reward model, which is trained with a large pre-collected dialog corpus, to judge the quality of system responses. All these efforts either demand a large pre-collected corpus to train reward models or utilize simple LSTM (Hochreiter and Schmidhuber, 1997) architecture, thus limiting the capability of efficiently adapting to new tasks or environments. SL-AGENT leverages pre-trained models to build reward function for eliminating the need for a large pre-collected corpus and thus is more effective and economical.

3 SL-AGENT

3.1 Dialog Model

SL-AGENT is a general framework that is compatible with any generative end-to-end dialog models. In this paper, we employ SOLOIST (Peng et al., 2020a) [3], 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 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. More details can be found in Peng et al. (2020a).

Synthetic Dialog Construction. Directly deploying the agent with a dialog model trained using limited examples to serve users yields unsatisfactory performance. The primary challenge is that the dialog model cannot handle unseen slot values due to insufficient coverage by training examples. To alleviate this issue, 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[5].

3.2 Reward Model

The reward model judges the quality of agent response in terms of (*i*) fluency, (*ii*) success, (*iii*) dialog flow consistency. We formulate the quality prediction problem as a binary classification task. As shown in Figure 3, the reward model R is a Transformer parameterized by $\boldsymbol{\theta_R}$ with inputs $\boldsymbol{x}$ defined as Equation 1.

$$\boldsymbol{y} = p_{\boldsymbol{\theta_R}}(\boldsymbol{s}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{r}). \tag{2}$$

The model is optimized towards promoting correct belief states and responses given the dialog history, and discouraging incorrect ones.

Note that dialog turn is the concatenation of input-output pairs of four dialog modules. To enhance the supervision, we propose to integrate two auxiliary tasks, *i.e.,* belief state tracking and response generation, and train the reward model using multi-task learning. Each task is described in detail as follows.

[3] Models and data are available at https://aka.ms/soloist.

[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.

[5] Note that synthetic dialog construction technique is a supplemental part of SL-AGENT, which can be enhanced by any state-of-the-art data-augmentation methods.

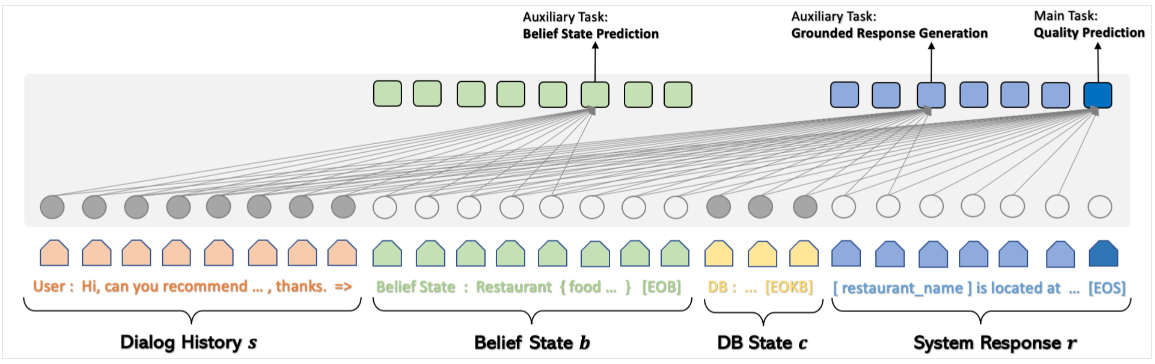


Figure 3: Architecture and training objectives of the reward model.

Main Task: Quality Prediction. For each dialog turn, we add a special token [EOS] at the end of the token sequence (as in Figure 3), to capture the overall dialog information. Then, we leverage the representation on [EOS] to predict the quality of belief state and response, *i.e.,* whether it is positive example $(y = 1)$ or negative example $(y = 0)$ (Peng et al., 2020a) using cross-entropy:

$$\mathcal{L}_{\mathrm{Q}} = y \log \left(p_{\boldsymbol{\theta_R}}(\boldsymbol{x})\right) + (1-y) \log \left(1 - p_{\boldsymbol{\theta_R}}\left(\boldsymbol{x}'\right)\right). \tag{3}$$

In order to endow the reward model with the capability of predicting the quality of belief state and agent response in terms of fluency, success and dialog flow consistency, we construct negative examples $\boldsymbol{x}'$ as follows: (*i*) negative belief with replaced belief, DB state $(\boldsymbol{s}, \boldsymbol{b}', \boldsymbol{c}', \boldsymbol{r})$; (*ii*) negative belief and response with replaced belief, DB state and response $(\boldsymbol{s}, \boldsymbol{b}', \boldsymbol{c}', \boldsymbol{r}')$; (*iii*) negative response with corrupted response $(\boldsymbol{s}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{r}')$ (i.e, replaced response, half-cut response, and response with repeated tokens).

Auxiliary Task: Belief Prediction. The objective of predicting the belief state, based on dialog history s, is defined as:

$$\mathcal{L}_{\mathrm{B}} = \log p(\boldsymbol{b} \mid \boldsymbol{s}) = \sum_{t=1}^{T_b} \log p_{\boldsymbol{\theta_R}}\left(b_t \mid b_{<t}, \boldsymbol{s}\right) \tag{4}$$

where the length of the belief state sequence is T_b, and $b_{<t}$ refers to all tokens before t.

Auxiliary Task: Grounded Response Generation. Similarly, the training objective of generating delexicalized response, grounded in the dialog history s, belief state b and DB state c, is (Peng et al., 2020a):

$$\begin{aligned} \mathcal{L}_{\mathrm{R}} &= \log p(\boldsymbol{r} \mid \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{s}) \\ &= \sum_{t=1}^{T_r} \log p_{\boldsymbol{\theta_R}}\left(r_t \mid r_{<t}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{s}\right) \end{aligned} \tag{5}$$

Full Training Objective. The multi-task objective for learning the model parameters $\boldsymbol{\theta_R}$ is represented as:

$$\mathcal{L}_{\boldsymbol{\theta_R}}(\mathcal{D}) = \sum_{n=1}^{|\mathcal{D}|} \left(\mathcal{L}_{\mathrm{Q}}\left(\boldsymbol{x}_n\right) + \mathcal{L}_{\mathrm{B}}\left(\boldsymbol{x}_n\right) + \mathcal{L}_{\mathrm{R}}\left(\boldsymbol{x}_n\right)\right) \tag{6}$$

where $|\mathcal{D}|$ is the number of training examples.

Similar to the dialog model, a pragmatic reward model is expected to be able to judge the quality of agent responses for any new task with limited training examples, and captures the potential to predict the quality of responses to unseen user behaviors. To achieve this goal, 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. The pre-training corpora used in this paper is the same as Peng et al. (2020a).

3.3 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 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 reward score is either 1 for the predicted positive example, or -1 otherwise. The training objective is represented as:

$$\mathcal{L}_{\theta_D} = -\sum_{t=1}^{T_{\hat{b}}} \log p_{\boldsymbol{\theta_D}}\left(\hat{b}_t \mid \hat{b}_{<t}, \boldsymbol{s}\right) \times R(\boldsymbol{s}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}, \boldsymbol{r}) \\ -\sum_{t=1}^{T_r} \log p_{\boldsymbol{\theta_D}}\left(r_t \mid r_{<t}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}, \boldsymbol{s}\right) \times R(\boldsymbol{s}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}, \boldsymbol{r}), \tag{7}$$

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.

3.4 Machine Teaching

Deployed task bots need learn to deal with new tasks, *i.e.,* task definition changes, as user requirements or environment evolve. Training task bots to adapt to a new environment requires high-quality annotated task-specific dialog corpora, which is expensive to collect and annotate. Machine teaching is already shown as an effective task bot training paradigm, which enables dialog authors to visualize dialogs, “teach" deployed task bots with new knowledge through active conversing (Simard et al., 2017; Williams and Liden, 2017; Shukla et al., 2020). Therefore, we leverage machine teaching to generate a certain number of annotated task-specific dialogs in a new domain with minimal human corrections.

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 session logs. (*ii*) Human experts select a few representative failed dialogs to construct training examples in new domains 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 the dialog model and reward model) is trained on these training examples continually (*i.e.,* both the dialog model and reward model), which enables the deployed task bot to handle task definitions changes, and user behaviour changes through RL refining in new iterations.

4 Experiments

In this section, we first describe how we design evaluations on changing environments. Then we introduce the experiments we conduct on four dialog domains using both automatic and human evaluation.

4.1 Experimental Setup

We validate the efficiency and flexibility of proposed SL-SOLOIST on four MultiWOZ single-domain dialog datasets (Budzianowski et al., 2018), reorganized by Peng et al. (2020a). Data statistics are shown in Table 2. Based on above datasets, we construct two settings to represent the changing environments – **Setting I** for user behavior changes; and **Setting II** for task definition changes.

Implementation Details. To construct training examples (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; Peng et al., 2020a). We implement proposed reward model based on Huggingface Pytorch Transformer (Wolf et al., 2020) using GPT-2 (Radford et al., 2019), which has 117M parameters. 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

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$_S$	53.00	29.00	8.49	73.74	54.55	6.94	59.00	29.50	4.29	62.50	41.50	7.33
SL-SOLOIST	**64.00**	**41.00**	**9.08**	**74.24**	**58.08**	**10.42**	**61.50**	**34.00**	**7.61**	**75.00**	**44.50**	**10.60**
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 1: End-to-end evaluation results on four tasks. SOLOIST$_5$ is trained with 5 examples. SOLOIST$_S$ refers to training with synthetic dialogs constructed from the 5 training examples. SL-SOLOIST indicates refining with 45 simulated human-bot dialogs using SL-AGENT. SOLOIST$_{50}$ denotes training with whole 50 labeled examples, which can be seen as an upper bound, quoted from Peng et al. (2020a). (Difference in mean is significant with $p<0.01$ based on Combined.)

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

Table 2: Data statistics of four single-domain dialog datasets (Peng et al., 2020a).

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 to complete new tasks using multi-task training objectives in the pre-training stage. 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.

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 generated response against human response. (*iv*) Combined(%) assesses the overall quality, which is defined as: Combined = (Inform + Success) × 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 Turks 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(%) 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.

4.2 Results of Setting I - User Behaviour Changes

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 user behavior changes[6]. We randomly sample 5 examples 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 for the task bot. Hence, it is applicable to simulate user behavior changes by modifying the remaining 45 dialogs as unlabeled imperfect human-bot interactions (through adding noise (*i.e.,* corrupting response[7]). This simulation setting allows us to perform a detailed analysis of SL-AGENT without

[6]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.

[7]Note that the associated belief states annotations are not used by SL-SOLOIST. (Construction process is described in Appendix C.)

Model	Attraction			Train			Hotel			Restaurant		
	Inform	Success	BLEU	Inform	Success	BLEU	Inform	Success	BLEU	Inform	Success	BLEU
SOLOIST$_S$	53.00	29.00	8.49	73.74	54.55	6.94	59.00	29.50	4.29	62.50	41.50	7.33
SL-SOLOIST	**64.00**	**36.00**	**8.84**	**78.28**	**60.10**	**9.06**	**62.00**	**31.50**	**7.39**	**70.00**	**45.00**	**10.93**
SL-SOLOIST*	66.00	40.00	9.01	75.76	67.17	10.38	62.50	32.50	7.83	70.50	47.50	11.36

Table 3: Automatic evaluation results on four tasks in Human-in-the-loop Setting. The first row refers to previously reported SOLOIST$_S$. SL-SOLOIST indicates refining with 20 real human-bot dialogs using SL-AGENT. SL-SOLOIST* refers to refining with 20 real human-bot dialogs using turn-level human feedback score (an upper bound). (Difference in mean is significant with p<0.01 based on Combined.)

much cost and easily reproduce the experimental results.

Simulation Evaluation Results. The end-to-end evaluation results on four domains are presented in Table 1. SOLOIST$_5$ is trained with 5 randomly sampled training examples. SOLOIST$_S$ is trained using synthetic dialogs constructed from the 5 randomly sampled training dialogs, which are the same dialogs used for training SOLOIST$_5$. The results show that the proposed synthetic dialog construction method improves the performance over all metrics by a significantly large margin, in particular `Inform` and `Success`, which demonstrates the effectiveness of synthetic dialog construction method on improving model generalization capability.

SL-SOLOIST is further refined based on SOLOIST$_S$ using proposed SL-AGENT, where the reward model is trained using the same training examples as SOLOIST$_5$. We observe that compared to SOLOIST$_S$, SL-SOLOIST dramatically improves the performance further and achieves comparable performance with SOLOIST$_{50}$ on all tasks. The results verify the vast potential of the proposed SL-AGENT, allowing the bot to automatically adapt to environments with user behaviours by learning from unlabeled noisy human-bot dialogs.

Human-in-the-Loop Evaluation Setup. Simulation setting allows effortless experimental studies to validate the effectiveness of SL-AGENT. However, the results are likely biased. Therefore, in the human-in-the-loop setting, we deploy SOLOIST$_S$ online and recruit human users to converse with it to achieve the assigned user goal. We collect 20 real human-bot dialogs to refine SOLOIST$_S$, resulting in the agent SL-SOLOIST.

Human-in-the-Loop Evaluation Results. The evaluation results on `Restaurant` are shown in Table 3. We observe that SL-SOLOIST refined with 20 real human-bot dialogs outperforms SOLOIST$_S$ over majority of evaluation metrics. We conclude that the results of human-in-the-loop evaluation and simulation evaluation are consistent, confirming that SL-SOLOIST enables self-learning after deployment by learning from interactions.

4.3 Results of Setting II – Task Definition Changes

Setup. We follow the domain extension experiment setting in Lipton et al. (2018) to assess the ability of SL-SOLOIST to quickly adapt to the changes in task definition. 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]*, and corresponding values for each DB entry (in Appendix E). 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 (in Appendix D). Finally, 10 examples are obtained through machine teaching for training and 50 for testing. We fine-tune the dialog model SOLOIST$_S$ and the reward model trained with 5 dialogs in train set, 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 dialogs, which are then used to refine itself.

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 extended functions. SOLOIST$_S$+TEACH outperforms the baseline by 17 points in terms of `Combined` score, which exhibits the effectiveness of machine teaching for adapting the agent to a new task. SL-SOLOIST+TEACH is the best performing system, lifting the `Combined`

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**

Table 4: Evaluation results on task definition changes. SOLOIST$_S$ denotes SOLOIST trained with synthetic dialogs constructed from 5 training dialogs in `Restaurant` domain. SOLOIST$_S$+TEACH is fine-tuned with 10 dialogs in `Restaurant-Ext` provided via machine teaching. SL-SOLOIST+TEACH is refined using 20 real human-bot dialogs collected after deployment. (Difference in mean is significant with p<0.01 based on Combined.)

Model	Restaurant				
	SR w/o g	SR w/ g	Under.	Appr.	Turns
SOLOIST$_S$	31.82	29.54	3.86	4.13	10.00
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.

score by approximately 7 points. The results demonstrate that SL-SOLOIST is able to adapt to new tasks and continually improve itself by learning from interactions, revealing the effectiveness of SL-AGENT that combines supervised learning, reinforcement learning and machine teaching for adaptations to changing environments.

4.4 Interactive Human Evaluation

Setup. Corpus-based evaluation is conducted using automatic evaluation metrics, which are rough proxies for agent response quality. Furthermore, automatic evaluation results may not adequately reflect the capability of dialog systems for helping users complete tasks in the real world, as real user inputs are more dynamic, complex, even with noise. Therefore, we conduct human evaluations to evaluate the performance of SOLOIST$_S$ and SL-SOLOIST interacting with human users, following the evaluation protocol in DSTC9 track 1 challenge (Gunasekara et al., 2020), with 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 achieves better performance than SOLOIST$_S$ over all the metrics, which are consistent with the automatic evaluation results. The significant improvement over SOLOIST$_S$ on two success rate metrics, especially success rate with grounding, verifies the effectiveness of the SL-AGENT for refining the dialog agent after deployment, as it more adequately reflects the system's capability for completing tasks in real scenarios.

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 in `Restaurant` domain. The first five rows indicate evaluation results of fine-tuned GPT-2, BERT, BERT-Large, RoBERTa, RoBERTa-Large, respectively. The last row refers to previously reported SL-SOLOIST. (Difference in mean is significant with p<0.01 based on Combined.)

4.5 Ablation Studies on Reward Model

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. The results of different auxiliary tasks in the multi-task objective are shown in Appendix B.

5 Conclusion

In this paper, we propose SL-AGENT, a novel self-learning framework in a more realistic changing environment setting. It combines supervised learning, reinforcement learning and machine teaching to enable the task bot to adapt to a changing environment by learning from user interactions after deployment. We demonstrate the effectiveness of SL-AGENT for automatically adapting to environments with user behavior changes in four dialog domains. In addition, SL-AGENT enables quick adaptation to an environment with task definition changes without the need for lots of extra data via machine teaching. 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.

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*.

Sam Coope, Tyler Farghly, Daniela Gerz, Ivan Vulić, and Matthew Henderson. 2020. Span-convert: Few-shot span extraction for dialog with pretrained conversational representations. *arXiv preprint arXiv:2005.08866*.

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*.

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.

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 Gašić and Steve Young. 2013. Gaussian processes for pomdp-based dialogue manager optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):28–40.

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.

Matthew Henderson, Inigo Casanueva, Nikola Mrkvsic, Pei-Hao Su, Tsung-Hsien Wen, and Ivan Vulic. 2019. Convert: Efficient and accurate conversational representations from transformers. *arXiv preprint arXiv:1911.03688*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

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.

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 and Ian Lane. 2018. Adversarial learning of task-oriented neural dialog models. *arXiv preprint arXiv:1805.11762*.

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*.

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*.

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, 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*.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020b. Few-shot natural language generation for task-oriented dialog. *arXiv preprint arXiv:2002.12328*.

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*.

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.

Chien-Sheng Wu, Steven Hoi, Richard Socher, and Caiming Xiong. 2020. Tod-bert: pre-trained natural language understanding for task-oriented dialogue. *arXiv preprint arXiv:2004.06871*.

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 $p_{\boldsymbol{\theta}^*_{\boldsymbol{D}}}$.

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 7;
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 Ablation Studies on Different Auxiliary Tasks in the Multi-Task Objective for Reward Model

We conduct ablation studies on `Restaurant` domain to analyze the effect of different auxiliary tasks in multi-task objective through adding only one auxiliary generation task. The results are shown in Table 7. The first row refers to fine-tuned GPT-2 with single quality prediction task. The second, third, fourth row indicates only adding response generation, only adding belief generation, and full objective, respectively. The results show that only adding auxiliary response generation task or belief generation task improves GPT-2 to some extent, but leg behind the results of full objective (*i.e.,* SL-SOLOIST) over all metrics. The result of SL-SOLOIST verifies the importance of multi-task objective for the reward model. GPT-2$_{\texttt{+Response}}$ legs behind GPT-2$_{\texttt{+Belief}}$ in terms of Combined, which may due to the high word overlap between delexicalized responses.

Reward model	`Restaurant`			
	`Inform`	`Success`	`BLEU`	`Combined`
GPT-2	67.00	41.50	9.30	63.55
GPT-2$_{\texttt{+Response}}$	67.00	41.50	10.40	64.65
GPT-2$_{\texttt{+Belief}}$	71.50	43.00	9.33	66.58
SL-SOLOIST	**75.00**	**44.50**	**10.60**	**70.35**

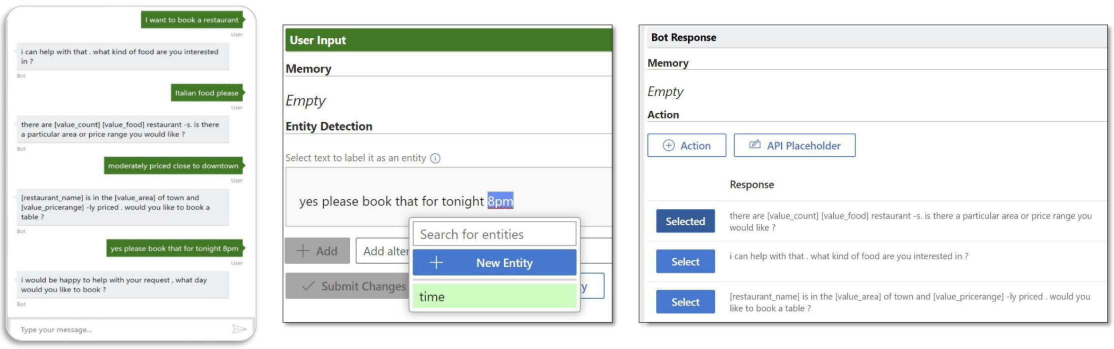
Table 7: Ablation study results on different auxiliary tasks in the multi-task objective for reward model in `Restaurant` domain. The first three rows indicate evaluation results of fine-tuned GPT-2 with single quality prediction task, adding response prediction only, adding belief prediction only, respectively. The last row refers to full objective (previously reported SL-SOLOIST). (Difference in mean is significant with p<0.01 based on Combined.)

C Simulated Human-Bot Corpora Construction

The unlabeled simulated human-bot corpora is constructed as follows: for each dialog turn, (i) we remove belief state annotations; (ii) we add negative examples by corrupting the response (*i.e.,* replaced response, half-cut response, and response with repeated tokens). We will release the simulated human-bot corpora for reproducible research.

D Machine Teaching Process

E Restaurant-Ext DB entry


I want to book a restaurant

User

i can help with that . what kind of food are you interested in ?

Bot

Italian food please

User

there are [value_count] [value_food] restaurant -s. is there a particular area or price range you would like ?

Bot

moderately priced close to downtown

User

[restaurant_name] is in the [value_area] of town and [value_pricerange] -ly priced . would you like to book a table ?

Bot

yes please book that for tonight 8pm

User

i would be happy to help with your request , what day would you like to book ?

Bot

Type your message...

User Input

Memory

Empty

Entity Detection

Select text to label it as an entity

yes please book that for tonight 8pm

Bot Response

Memory

Empty

Action

	Response
Selected	there are [value_count] [value_food] restaurant -s. is there a particular area or price range you would like ?
Select	i can help with that . what kind of food are you interested in ?
Select	[restaurant_name] is in the [value_area] of town and [value_pricerange] -ly priced . would you like to book a table ?

(a) Human-bot dialog (b) Belief State Edit (c) Response Edit

Figure 4: Illustration of machine teaching process via CL (Shukla et al., 2020). Dialog authors edit the human-bot conversion log in (a) via correcting its belief states in (b), and selecting/inserting/correcting a appropriate response, creating a response with action templates, adding slots and corresponding values for new tasks in (c) (Peng et al., 2020a)

```
{
  "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",
  "signature": "african babooti",
  "type": "restaurant",
  "delivery fee": "5 pounds",
  "dish": "Greek Chicken Pasta",
  "start_time": "09:50",
  "end_time": "22:30"
},
```

Figure 5: `Restaurant-Ext` DB entry.