“Think Before You Speak”: Improving Multi-Action Dialog Policy by Planning Single-Action Dialogs

Shuo Zhang[1] , **Junzhou Zhao**[1*] , **Pinghui Wang**[1*] , **Yu Li**[1] , **Yi Huang**[2] , **Junlan Feng**[2]
[1]MOE KLINNS Lab, Xi’an Jiaotong University, Xi’an 710049, P. R. China
[2]JIUTIAN Team, China Mobile Research, Beijing 100053, P. R. China
{zs412082986, liyu1998}@stu.xjtu.edu.cn, {junzhou.zhao, phwang}@mail.xjtu.edu.cn,
{huangyi, fengjunlan}@chinamobile.com

Abstract

Multi-action dialog policy (MADP), which generates multiple atomic dialog actions per turn, has been widely applied in task-oriented dialog systems to provide expressive and efficient system responses. Existing MADP models usually imitate action combinations from the labeled multi-action dialog samples. Due to data limitations, they generalize poorly toward unseen dialog flows. While interactive learning and reinforcement learning algorithms can be applied to incorporate external data sources of real users and user simulators, they take significant manual effort to build and suffer from instability. To address these issues, we propose Planning Enhanced Dialog Policy (PEDP), a novel multi-task learning framework that learns single-action dialog dynamics to enhance multi-action prediction. Our PEDP method simulates single-action dialog fragments with model-based planning to conceive what to express before deciding the current response. Experimental results on the MultiWOZ dataset demonstrate that our fully supervised learning-based method achieves a solid task success rate of 90.6%, improving 3% compared to the state-of-the-art methods. The source code and the appendix of this paper can be obtained from https://github.com/ShuoZhangXJTU/PEDP.

1 Introduction

Task-oriented Dialog Systems (TDS) assist users in completing specific tasks (e.g., restaurant booking) using natural language, and have been applied to various commercial services [Li *et al.*, 2017; Zhang *et al.*, 2020; Qin *et al.*, 2021; Zhang *et al.*, 2021]. In a TDS, the dialog policy, which decides the next system actions according to the current dialog context, plays a vital role as it dramatically influences the dialog efficiency (e.g., the conciseness and smoothness).

Prior work on dialog policy typically assumes that the agent only produces one action per turn [Mnih *et al.*, 2015; Peng *et al.*, 2018], which leads to lengthy dialogs and increases the risk of task failure. To address these weaknesses, some recent works have investigated *Multi-Action Dialog Policy Learning* (MADPL) that generates multiple actions concurrently as the current system response to boost the expressive power and efficiency of the TDS [Shu *et al.*, 2019; Jhunjhunwala *et al.*, 2020]. MADPL can be formulated as a multi-label classification or sequence generation problem, and thus solved by *Supervised Learning* (SL) based methods that imitate action combinations from labeled dialog samples [Shu *et al.*, 2019].

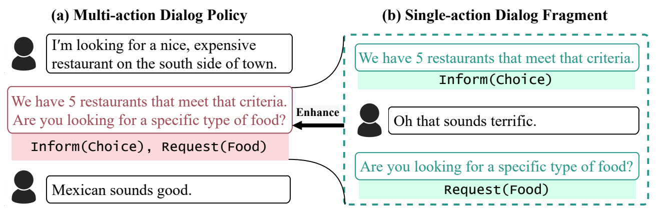
(a) Multi-action Dialog Policy
(b) Single-action Dialog Fragment
I'm looking for a nice, expensive restaurant on the south side of town.
We have 5 restaurants that meet that criteria. Are you looking for a specific type of food?
Inform(Choice), Request(Food)
Mexican sounds good.
Enhance
We have 5 restaurants that meet that criteria.
Inform(Choice)
Oh that sounds terrific.
Are you looking for a specific type of food?
Request(Food)

Figure 1: (a) Example dialog under multi-action dialog policy[1]. We propose to learn single-action dialog dynamics (b) to model conditional act combination patterns and enhance multi-action prediction.

Despite their success, existing SL-based methods often generalize poorly towards real-world human-machine dialog flows [Jhunjhunwala *et al.*, 2020]. Task-oriented dialogs are known to share the “one-to-many” nature that various appropriate responses may exist under the same context [Huang *et al.*, 2020]. Accordingly, in the multi-action setting, various action combinations could be considered appropriate under the same context, which are mostly not covered in limited human-labeled dialog datasets [Jhunjhunwala *et al.*, 2020]. With straightforward imitation, existing SL-based methods will be restricted to a small subspace of the entire action space, leading to limited generalization ability towards unseen dialog scenarios.

Some existing studies try to address the above issue from three different perspectives: 1) *Interactive Learning* [Jhunjhunwala *et al.*, 2020], which performs data augmentation through human-machine interactions; 2) *Reinforcement Learning* (RL) [Takanobu *et al.*, 2019; Zhu *et al.*, 2020], which enables learning from online human-interacting dialogs; and 3) *Adversarial Learning* (AL) [Takanobu *et al.*,

*Corresponding Author

[1]A dialog policy responses by predicting atomic dialog actions represented as “Domain-Intent(Slot)” phrases. We omit the domain (“restaurant”) for clarity.

2019; Li *et al.*, 2020], which provides extra supervision signals with a machine-learned discriminator. However, the performance gain is obtained at the cost of additional supervision. For example, they require tons of human work to label dialogs, build real-world environments such as user simulators, and design learning strategies. Moreover, RL and AL-based algorithms suffer from the unstable training process [Suzuki and Ogata, 2020; Tian *et al.*, 2020], bringing extra deployment costs in real-world applications.

In this work, we present *Planning Enhanced Dialog Policy* (PEDP), an SL-based multi-task learning framework for improved MADPL. Unlike prior methods that directly imitate complex action combination patterns from limited multi-action dialog ***turns***, we propose to further learn such patterns from single-action dialog ***fragments***. We find that one turn in the multi-action dialog usually corresponds to several turns in the single-action setting (i.e., fragments) to achieve the similar (or identical) dialog state transition (see Fig. 1). This phenomenon inspires us to learn to simulate single-action dialog fragments for enhancing multi-action prediction through discovering and incorporating contextually relevant dialog actions. To materialize this idea, we integrate model-based planning into the decision process and propose a *Single-action Dialog Planning* module. It plans several single-action paths (dialog fragments) based on the current dialog state before multi-action prediction. Each path looks ahead several turns through "self-play" between a discrete dialog policy model and a world model that imitates user behaviors. We further propose an *Ensemble Prediction* module that aggregates the multi-action prediction probabilities from multiple paths to reduce the impact of low-quality dialogs. Unlike seeking external data sources, our method better uses existing dialog samples. It performs multi-action prediction in a reasonable and explainable way, thus boosting performance towards unseen dialog flows.

To evaluate the proposed method, we conduct experiments on the MultiWOZ [Budzianowski *et al.*, 2018] and the SGD [Rastogi *et al.*, 2020] datasets. The experimental results demonstrate that the extension of planning brings significant improvement to our model's performance and dialog efficiency, resulting in a promising task success rate of 90.6%, improving 3% compared to the state-of-the-art methods.

2 Related Work

Multi-Action Dialog Policy Learning (MADPL). Recent efforts have been made to learn a multi-action dialog policy for task-oriented dialogs where the agent can produce multiple actions per turn to expand its expressive power. Some attempts [Zhu *et al.*, 2020; Gordon-Hall *et al.*, 2020] (primarily DQN-based methods) incorporate the top-k most frequent dialog action combinations in the dataset into the output space and simplify MADPL as a multi-class classification problem. However, the expressive power and flexibility of the dialog agent are severely limited. Considering the sequential dependency among the acts, Shu *et al.* (2019) proposes to formulate MADPL as a sequence generation problem. However, the order in which people express their intentions is complicated and has not been properly annotated in the existing human-to-human dialog datasets[2]. The discrepancy between the task setting and data annotation can lead to misleading performance. Recent works [Shu *et al.*, 2019; Li *et al.*, 2020] typically cast MADPL as a multi-label classification problem and address it with supervised learning. However, the complex action combinations can exponentially enlarge the output space [Jhunjhunwala *et al.*, 2020]. The limited coverage of the output action space of the existing dialog corpus will greatly hinder SL-based methods' performance. Jhunjhunwala *et al.* (2020) addresses this problem by data augmentation through interactive learning. The problem can also be partially alleviated through RL-based [Zhu *et al.*, 2020] and AL-based methods [Takanobu *et al.*, 2019; Li *et al.*, 2020] by allowing the model to explore more potential dialog scenarios. Unlike existing works that "memorize" action combinations from limited multi-action samples, we mimic the human decision process and discover dialog actions to express before condensing them into a brief response through simulating single-action dialogs.

Planning in Dialog Policy Learning. Planning refers to the process that uses a model of the environment to improve a policy. Peng *et al.* (2018) first introduces background planning for task-completion dialog policy learning. They propose to use Deep Dynamic Q-network, in which a world model is trained to mimic real-world users and provide extra simulated experience for training. The integration of planning can substantially improve data efficiency. Since the world model is not perfect and may generate erroneous experiences (state-action pairs) that hinder policy learning. Su *et al.* (2018) proposes to train a discriminator to filter out low-quality simulated experiences. Wu *et al.* (2019) design a switcher-based mechanism to balance the use of real and simulated experiences automatically. Wang *et al.* (2020) proposes decision-time planning that adopts a rollout algorithm (Monte-Carlo Tree Search) to boost the accuracy of action value estimations. In summary, previous work has focused on RL-based single-action dialog policy learning. Instead, we address SL-based multi-action dialog policy learning and extend planning to generate anticipatory single-action dialog trajectories for further combination.

3 Methodology

In this section, we introduce the *Planning-Enhanced Dialog Policy* (PEDP) framework, an SL-based multi-task learning framework for improved MADPL.

3.1 Task Formulation

We regard MADPL as a multi-label classification task, i.e., given the current dialog state that summarizes the dialog history, we want to predict a *macro-action* that is a set of independent *atomic dialog actions* that serves as the current system response. Following [Li *et al.*, 2020], each atomic dialog action is a concatenation of domain name, action type, and slot name, e.g., "hotel-inform-area".

[2]The sequential order of the actions in the annotated samples is usually inconsistent with the expressive order in text responses.

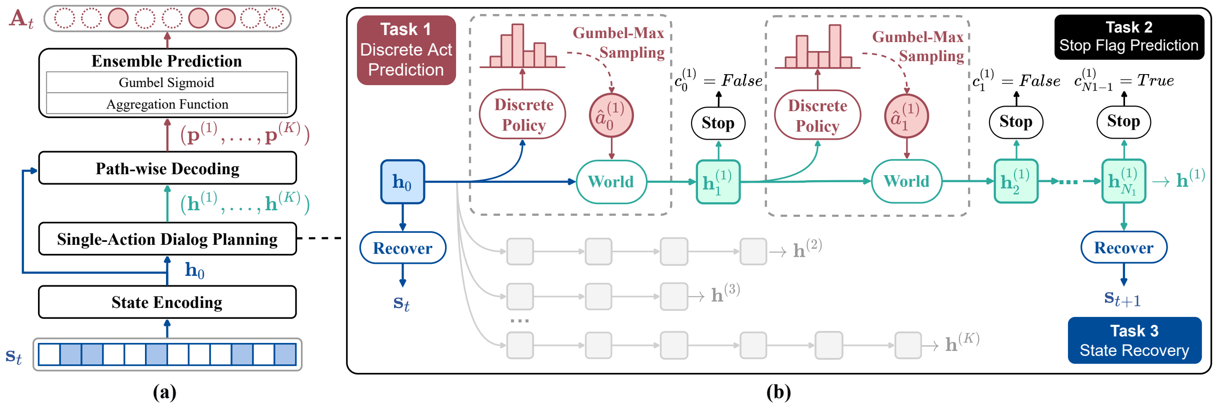

Figure 2: (a) The Planning-Enhanced Dialog Policy (PEDP) framework. It utilizes a *single action dialog planning* module (b) to incorporate contextually relevant contents before multi-action prediction. A total of K single-action dialog procedures are planned, with the k-th path looking ahead N_k steps under single-action dialog dynamics. At each step, the discrete policy model predicts an atomic dialog action a_n given the previous dialog state embedding $\mathbf{h}_{n-1}$. The world model, which simulates user behavior, responds to the predicted action a_n and updates the dialog state embedding from $\mathbf{h}_{n-1}$ to $\mathbf{h}_n$.

3.2 Overview

The PEDP framework is inspired by the human dialog strategy of "think before you speak" that conceives relevant content to express before deciding the current response. We first discover contextually relevant dialog actions by looking ahead several turns under the single-action dialog dynamics (i.e., *Single Action Dialog Planning*) and then use the planning result as guidance to predict the atomic dialog actions that should be executed concurrently (i.e., macro-action).

Formally, let $\mathbf{s}_t$ denote the current dialog state representing the dialog history up to the current dialog turn. We encode $\mathbf{s}_t$ into a dialog state embedding $\mathbf{h}_t$. Given the current state embedding $\mathbf{h}_t$, we plan K independent single-action dialog paths, and the k-th dialog path is represented by a vector $\mathbf{h}^{(k)}$, $k = 1, \ldots, K$. Our model then decodes each dialog path to a probability distribution over atomic dialog actions, i.e., $\mathbf{p}^{(k)}$. Finally, these distributions $\{\mathbf{p}^{(k)}\}_{k=1}^{K}$ are aggregated to form a unified distribution, from which atomic dialog actions in the macro-action $\mathbf{A}_t$ are sampled.

Figure 2 depicts the architecture of PEDP, which mainly consists of four modules: 1) *State Encoding*, which encodes the dialog state into a compact vector; 2) *Single-Action Dialog Planning*, which plans several independent dialog paths with each one looking ahead several steps considering only the single-action dialog dynamics; 3) *Path-wise Decoding*, which predicts the multi-action probability distribution for each dialog path; and 4) *Ensemble Prediction*, which aggregates the probabilities and samples the final macro-action. In what follows, we elaborate on each of these modules in PEDP.

3.3 State Encoding

The dialog state is a summary of the dialog history. Inspired by [Li *et al.*, 2020], we propose to include the following information into dialog state $\mathbf{s}_t$ at time step t: 1) embedding of returned entities for a query, 2) the last user action, 3) the last system action, and 4) the belief state that contains the requested slots and informed slots of user and agent.

To densify the dialog state, we apply a Fully-connected Feed-forward Network (FFN) to obtain a compact dialog state embedding $\mathbf{h}_t$. The FFN consists of two linear transformations with a ReLU activation in the middle layer, i.e.,

$$\mathbf{h}_t = \mathtt{FFN}_{enc}(\mathbf{s}_t) = \mathtt{ReLU}(\mathbf{s}_t W_1 + b_1) W_2 + b_2.$$

In what follows, this dialog state embedding $\mathbf{h}_t$ will serve as the initial dialog state embedding for planning.

3.4 Single-Action Dialog Planning

As discussed in the introduction, the straight-forward multi-label imitation of action co-occurrence suffers from the poor generalization ability issue due to the lack of labeled multi-action training data. To address this issue, we propose a *single-action dialog planning* module that further leverages single-action dialog samples to model contextual action co-occurrence patterns. It looks ahead several steps under single-action dialog dynamics to predict the next dialog state, which will serve as the additional information to enhance macro-action prediction. In what follows, we describe how to model the dialog state transition in a single-action dialog.

Given the initial dialog state embedding $\mathbf{h}_t$ at the current dialog turn t, we plan K independent single-action dialogs to discover contextually related atomic dialog actions. In each dialog, a *discrete policy model* and a *world model* interact to look ahead several steps. Let $\mathbf{h}_{t,n}^{(k)}$ denote the dialog state embedding at the n-th step in the k-th dialog for $n = 0, \ldots, N_k$ where N_k is the length of the k-th dialog, and $\mathbf{h}_{t,0}^{(k)} = \mathbf{h}_t$, $\mathbf{h}_{t,N_k}^{(k)} = \mathbf{h}^{(k)}$. The last dialog state embedding $\mathbf{h}^{(k)}$ estimates the hidden vector of the future dialog state $\mathbf{s}_{t+1}$ and summarizes the planned single-action dialog. In what follows, we describe how to plan a single step from $\mathbf{h}_{t,n}^{(k)}$ to obtain $\mathbf{h}_{t,n+1}^{(k)}$. To simplify the notations, we omit the dialog index k and dialog turn t.

Given the dialog state embedding $\mathbf{h}_n$ at step n, we want first to predict a single action a_n and then use it to update $\mathbf{h}_n$ to obtain $\mathbf{h}_{n+1}$. This can be achieved by cyclical interactions between a discrete policy model (DP) that predicts an atomic dialog action under the current-step dialog state and a world model that simulates user behavior to update the current-step dialog state, i.e.,

$$a_n = \texttt{DP}(\mathbf{h}_n) \triangleq \text{GumbelSoftmax}^{(\tau_d)}(\mathbf{h}_n W_d + b_d)$$
$$\mathbf{h}_{n+1} = \texttt{World}(\mathbf{h}_n, a_n) \triangleq \texttt{GRU}(\mathbf{h}_n, \texttt{Emb}(a_n)).$$

Here, DP is implemented as a single linear layer followed by a Gumbel-Softmax function [Jang *et al.*, 2016] parameterized by τ_d. The Gumbel-Softmax function draws an atomic dialog action sample from a categorical distribution, diversifying the planned dialogs. τ_d is selected to balance the approximation bias and the magnitude of gradient variance. The world model is implemented using a GRU to model dialog state transitions, and $\texttt{Emb}(a_n)$ denotes the embedding vector of atomic dialog action a_n.

We observe that both executing a macro-action and sequentially executing the corresponding atomic dialog actions will lead to a similar (or identical) state transition (see Fig. 1). Accordingly, we assume that the planned information is adequate once such a state transition is achieved, and propose to determine if to stop planning by comparing the initial and the **next** dialog state embeddings, i.e., $\mathbf{h}_0$ and $\mathbf{h}_{n+1}$ (see Fig. 2b). This is also modeled using a neural network, i.e.,

$$c_n = \text{GumbelSoftmax}^{(\tau_s)}(\texttt{FFN}_{st}([\mathbf{h}_0 : \mathbf{h}_{n+1}]))$$

where c_n is a binary variable, ":" denotes vector concatenation, and FFN is a 2-layer fully-connected feed-forward network using the ReLU activation function in the middle layer.

Note that the single-action dialog planning module estimates dialog state transitions in the dialog state embedding space from $\mathbf{h}_0$ to $\mathbf{h}_N$, and it demands supervision from the labeled structural dialog state transition samples from $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$. To enable model training, we incorporate a recovery model for mapping the state embeddings back to the structured state-space. Formally, we have,

$$\mathbf{s}_t = \texttt{Recover}(\mathbf{h}_0)$$
$$\mathbf{s}_{t+1} = \texttt{Recover}(\mathbf{h}_N)$$

Here, Recover is implemented by a 2-layer FFN and is only used during the training stage.

3.5 Path-wise Decoding

Leveraging the planned single-action dialog, we predict the probability distribution over atomic dialog actions.

Given the initial and the last dialog state embeddings $\mathbf{h}_0$ and $\mathbf{h}^{(k)}$, we decode the multi-action probability distribution $\mathbf{p}^{(k)}$ with a neural network applied to each path separately and identically. Since our task formulation posits no dependency among atomic dialog actions, following [Li *et al.*, 2020], we perform binary classification for each specific action in the atomic dialog action space independently to decide whether it is selected. Specifically, we instantiate the decoder $\mathbf{p}^{(k)} = [\mathbf{p}_1^{(k)} : \ldots : \mathbf{p}_M^{(k)}]$, where k refers to the planned path and M is the size of the action space. Each $\mathbf{p}_m^{(k)}, m = 1, \ldots, M$ is a vector computed as:

$$\mathbf{p}_m^{(k)} = \texttt{FFN}_m^{dec}([\mathbf{h}_0 : \mathbf{h}^{(k)}]).$$

Future efforts at exploring different decoders may further improve the performance.

3.6 Ensemble Prediction

The performance of the proposed model is affected by the quality of the planned dialogs. To reduce the impact of low-quality dialogs, we propose to ensemble the multi-action probability distributions with an aggregation function, i.e.,

$$\mathbf{P}_t = \text{Aggr}(\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(K)})$$

where $\text{Aggr}(\cdot)$ is the mean average in our case.

Task-oriented dialogs share a critical "one-to-many" nature that different responses may be proper under the same context, making it essential to incorporate stochastic factors to build a multi-action dialog policy. We empirically verified that this stochasticity could be achieved with probability sampling. In our case, we apply the Gumbel-Sigmoid function to sample the macro-action, that is,

$$\mathbf{A}_t = \text{GumbelSigmoid}(\mathbf{P}_t) = \frac{e^{(\mathbf{P}_t + g_1)/\tau}}{e^{(\mathbf{P}_t + g_1)/\tau} + e^{(\mathbf{P}_t + g_2)/\tau}}$$

Here $\text{GumbelSigmoid}(\cdot)$ is a modification of the Gumbel-Softmax function, regarding sigmoid as a softmax with two logits p and 0. τ denotes the temperature factor, g_1 and g_2 are Gumbel noises.

3.7 Training

We use a multi-task objective to train all the modules jointly.

Task 1: Discrete Act Prediction (DAP) For the planned single-action sequence $\boldsymbol{a} = (a_0, \ldots, a_{N-1})$, we aim to maximize the log-likelihood (MLE) for the joint probability $p(\boldsymbol{a}|\mathbf{h}_0)$, which can be factorized as:

$$p(\boldsymbol{a}|\mathbf{h}_0) = p_\theta(a_0|\mathbf{h}_0) \prod_{n=1}^{N-1} \underbrace{p_\theta(a_n|\mathbf{h}_n)}_{\text{DAP}} \underbrace{p_\phi(\mathbf{h}_n|a_{n-1}, \mathbf{h}_{n-1})}_{\text{state transition}}$$

where θ and ϕ denotes trainable parameters for the discrete dialog policy model and the world model, respectively.

Task 2: Stop Flag Prediction (SFP) We define the objective of predicting the stop flag sequence $\boldsymbol{c} = (c_0, \ldots, c_{N-1})$ as MLE for the joint probability $p(\boldsymbol{c}|\mathbf{h}_0)$, which can be factorized as:

$$p(\boldsymbol{c}|\mathbf{h}_0) = \prod_{n=0}^{N-1} \underbrace{p_\gamma(c_n|\mathbf{h}_{n+1}, \mathbf{h}_0)}_{\text{SFP}} \underbrace{p_{\phi,\theta}(\mathbf{h}_{n+1}|\mathbf{h}_n)}_{\text{1-step planning}}$$

where γ parameterizes the stop prediction model, the joint probability of $p_{\phi,\theta}(\mathbf{h}_{n+1}|\mathbf{h}_n)$ is factorized as $p_\phi(\mathbf{h}_{n+1}|a_n, \mathbf{h}_n)p_\theta(a_n|\mathbf{h}_n)$ of state transition and discrete act prediction.

Task 3: State Recovery (SR) We consider a state recovery objective to supervise state encoding and state transition. Precisely, we predict the current dialog state $\mathbf{s}_t$ and next dialog state $\mathbf{s}_{t+1}$ with the initial dialog state embedding $\mathbf{h}_0$ and the last state embedding $\mathbf{h}_N$, respectively, i.e.,

$$p(\mathbf{s}_t) = \underbrace{p_\zeta(\mathbf{s}_t|\mathbf{h}_0)}_{\text{SR}} \underbrace{p_\eta(\mathbf{h}_0|\mathbf{s}_t)}_{\text{state encoding}}$$

$$p(\mathbf{s}_{t+1}|\mathbf{s}_t) = \underbrace{p_\zeta(\mathbf{s}_{t+1}|\mathbf{h}_N)}_{\text{SR}} \underbrace{p_\eta(\mathbf{h}_0|\mathbf{s}_t)}_{\text{state encoding}} \prod_{n=0}^{N-1} \underbrace{p_{\phi,\theta}(\mathbf{h}_{n+1}|\mathbf{h}_n)}_{\text{1-step planning}}$$

where η and ζ denotes trainable parameters for state encoder and the `Recover`, respectively. The joint probability $p_{\phi,\theta}(\mathbf{h}_{n+1}|\mathbf{h}_n)$ is the same as explained in Task 2.

Task 4: Multi-Action Prediction (MAP) Finally, we introduce a multi-action prediction objective to supervise all the modules in our PEDP framework. That is:

$$p(\mathbf{A}_t|\mathbf{s}_t) = \underbrace{p_\omega(\mathbf{A}_t|\mathbf{h}_0, \mathbf{h}_N)}_{\text{MAP}} \underbrace{p_\eta(\mathbf{h}_0|\mathbf{s}_t)}_{\text{state encoding}} \prod_{n=0}^{N-1} \underbrace{p_{\phi,\theta}(\mathbf{h}_{n+1}|\mathbf{h}_n)}_{\text{1-step planning}}$$

where ω denotes trainable parameters for the decoder. The rest is the same as explained in Task 3.

Full Objective The overall loss is the weighted sum of four losses: two cross-entropy losses of DAP and SFP and two binary cross-entropy losses of SR and MAP[3].

4 Experiments

4.1 Settings

Evaluation Metrics We perform the automatic evaluation from two perspectives:

1) *Interactive Evaluation*, which evaluates task completion quality through dialog interaction with the user simulator. We use *Inform F1*, *Inform Recall*, *Match*, *Turn*, and *Success* as evaluation metrics. *Inform F1* and *Inform Recall* evaluate whether all the requested information (e.g., hotel address) has been provided. *Match* evaluates whether the booked entities match the indicated constraints (e.g., the cheapest restaurant in town). *Turn* and *Success* evaluate the efficiency and level of task completion of dialog agents, respectively. Specifically, a dialog is considered successful if all the requested information is provided (i.e., *Inform Recall* $= 1$) and all the entities are correctly booked (i.e., *Match* $= 1$).

2) *Standard Evaluation*, which measures the generalization ability with the test set of the training corpus. We report the sample-wise *Precision*, *Recall*, and F_1 score of the macro-actions, where each atomic dialog action is assumed to be correct if it is included in the ground truth.

Datasets We conduct experiments on two dialog datasets.

1) *MultiWOZ* [Budzianowski *et al.*, 2018] is a multi-domain dialog dataset with $10,438$ dialogs covering 7 distinct domains and 24 slots. We apply the agenda-based user simulator as the interaction environment for interactive evaluation.

2) *SGD* [Rastogi *et al.*, 2020] is published to build scalable multi-domain conversational agents. It has $16,142$ dialogs spanning 16 domains and 214 slots. We conduct a standard evaluation on SL-based methods to address the scaling concerns. We do not report interactive evaluation results since the corresponding official user simulator is unavailable.

Baselines Three types of baselines are explored: 1) Supervised Learning (SL), 2) Reinforcement Learning (RL), and 3) Adversarial Learning (AL). We ignore DQN-based methods because these methods do not suit the multi-label classification setting in our task formulation (see Sec.2 for details).

	MultiWOZ				
Agent	**Turn**	**Match**	**Rec**	**F1**	**Success**
DiaMultiClass	11.46 $\pm$0.56	0.68 $\pm$3.9%	0.81 $\pm$3.2%	0.81 $\pm$2.1%	67.3 $\pm$3.69
+ sample	9.23 $\pm$0.2	0.82 $\pm$1.1%	0.90 $\pm$1.8%	0.77 $\pm$1.2%	81.4 $\pm$1.78
DiaSeq (beam)	9.06 $\pm$0.67	0.81 $\pm$0.4%	0.9 $\pm$1.2%	0.86 $\pm$0.9%	81.4 $\pm$0.16
greedy	10.35 $\pm$0.04	0.68 $\pm$1.5%	0.80 $\pm$0.5%	0.77 $\pm$0.5%	67.7 $\pm$1.02
+ sample	8.82 $\pm$0.1	0.86 $\pm$0.6%	0.93 $\pm$0.4%	0.81 $\pm$0.5%	86.9 $\pm$0.49
DiaMultiDense	9.66 $\pm$0.15	0.85 $\pm$0.6%	0.94 $\pm$0.4%	0.87 $\pm$0.6%	86.3 $\pm$0.64
- sample	12.75 $\pm$0.77	0.61 $\pm$6%	0.72 $\pm$5.4%	0.80 $\pm$2.3%	58.4 $\pm$6.05
gCAS	11.69 $\pm$0.53	0.56 $\pm$1.4%	0.72 $\pm$0.4%	0.76 $\pm$1.4%	58.8 $\pm$2.82
GP-MBCM[4]	**2.99**	0.44	-	0.19	28.9
ACER[4]	10.49	0.62	-	0.78	50.8
PPO [4]	15.56	0.60	0.72	0.77	57.4
ALDM[4]	12.47	0.69	-	0.81	61.2
GDPL	7.54 $\pm$0.43	0.84 $\pm$0.9%	0.89 $\pm$2.2%	**0.88** $\pm$1.2%	83.2 $\pm$1.48
DiaAdv	8.90 $\pm$0.18	0.87 $\pm$0.9%	0.94 $\pm$0.75%	0.85 $\pm$0.58%	87.6 $\pm$0.9
- sample	11.9 $\pm$0.88	0.62 $\pm$5.9%	0.73 $\pm$4.6%	0.80 $\pm$2.1%	61.7 $\pm$5.59
PEDP	8.69 $\pm$0.15	**0.88** $\pm$1.3%	**0.97** $\pm$0.4%	0.87 $\pm$1.1%	**90.6** $\pm$0.68
- planning	9.66 $\pm$0.15	0.85 $\pm$0.6%	0.94 $\pm$0.4%	0.87 $\pm$0.6%	86.3 $\pm$0.64
- ensemble	9.25 $\pm$0.43	0.88 $\pm$1.97%	0.96 $\pm$0.8%	0.85 $\pm$2.5%	89.1 $\pm$1.74
- sample	8.85 $\pm$0.22	0.82 $\pm$2.5%	0.93 $\pm$1.4%	0.86 $\pm$1.6%	83.4 $\pm$1.01

Table 1: Interactive evaluation results. We simulate 1,000 dialogs per run and report the mean and standard deviation over 5 runs.

(SL) *DiaMultiClass* [Li *et al.*, 2020] is a FFN with Sigmoid as the last activation function.

(SL) *DiaSeq* [Li *et al.*, 2020] encodes the multi-hot dialog state vector with a 2-layer FFN and sequentially decodes the macro-action with an RNN.

(SL) *DiaMultiDense* [Li *et al.*, 2020] performs binary classification for each specific action with the same network described in Section 3.7.

(SL) *gCAS* [Shu *et al.*, 2019] is a novel DiaSeq variant that uses tangled RNNs to predicts the macro-action by an auto-regressive generation of intent and slots.

(RL) *GP-MBCM* [Gašić *et al.*, 2015] is a Gaussian process based Multi-domain Bayesian Committe Machine that decomposes the policy into domain-specific policies.

(RL) *ACER* [Wang *et al.*, 2016] is the Actor-Critic RL policy with Experience Replay. ACER is sample efficient and is well adapted for large discrete action spaces.

(RL) *PPO* [Schulman *et al.*, 2017] denotes Proximal Policy Optimization, a simple and stable RL algorithm that applies a constant clipping mechanism.

(AL) *ALDM* [Liu and Lane, 2018] is the Adversarial Learning Dialog Model. ALDM estimates a reward at the end of the session to optimize the dialog policy.

(AL) *GDPL* [Takanobu *et al.*, 2019] equips PPO with a "rewarder" for joint reward estimation and policy optimization. GDPL is regarded as the SOTA method.

(AL) *DiaAdv* [Li *et al.*, 2020] further improves the DiaMultiDense model with adversarial fine-tuning.

4.2 Model Evaluation

Interactive Evaluation We report the performance of each approach that interacts with the user simulator in Table 1.

We observe that our PEDP framework achieves the highest performance in the task success by 3% compared to the second-best model DiaAdv and by 7.4% compared to the SOTA GDPL method on account of the substantial improvement in match rate and inform recall. This demonstrates that

[3]We refer the reader to Appendix A and B for data usage and model implementation details.

[4]Results reused from [Li *et al.*, 2020]

	MultiWOZ			**SGD** (scaling)		
Agent	**F1%**	**Precision%**	**Recall%**	**F1%**	**Precision%**	**Recall%**
DiaMultiClass	39.41 ±1.08	54.59 ±1.71	34.32 ±1.32	58.09 ±0.63	81.29 ±1.13	46.29 ±0.57
+ sample	38.91 ±0.74	47.28 ±0.68	37.56 ±1.08	58.03 ±0.64	81.48 ±0.18	46.14 ±0.80
DiaSeq (beam)	44.64 ±2.08	51.91 ±0.99	43.66 ±2.27	63.13 ±0.18	86.04 ±0.5	50.83 ±0.30
greedy	48.34 ±0.45	54.71 ±0.21	48.84 ±0.84	63.21 ±0.35	86.31 ±0.7	50.85 ±0.40
+ sample	37.82 ±0.45	43.02 ±0.48	38.91 ±0.64	62.64 ±1.03	85.54 ±1.62	50.40 ±0.76
DiaMultiDense	35.92 ±0.54	51.93 ±0.33	30.10 ±0.69	57.85 ±0.68	80.64 ±0.43	46.21 ±0.89
- sample	34.35 ±0.62	52.14 ±0.19	27.74 ±0.74	56.69 ±0.62	79.54 ±0.88	45.19 ±0.75
gCAS	50.01 ±0.62	55.56 ±0.59	51.21 ±1.74	76.37 ±1.60	77.70 ±1.46	79.99 ±1.03
GDPL	31.89±0.96	50.14 ±0.79	24.99 ±1.14	-	-	-
+ sample	34.60 ±0.47	45.01 ±0.24	31.54 ±0.80	-	-	-
DiaAdv	40.97 ±0.95	53.44 ±0.50	36.84 ±1.30	-	-	-
- sample	41.71 ±0.47	56.46 ±0.45	36.28 ±1.48	-	-	-
PEDP	64.63 ±0.16	77.03 ±1.39	61.77 ±1.01	84.12 ±0.38	91.66 ±0.52	81.19 ±0.4
- planning	35.92 ±0.54	51.93 ±0.33	30.10 ±0.69	57.85 ±0.68	80.64 ±0.43	46.21 ±0.89
- ensemble	64.34 ±0.29	77.63 ±2.04	60.85 ±1.54	83.31 ±0.55	91.66 ±0.78	80.10 ±0.55
- sample	**66.95** ±0.45	**78.11** ±3.03	**65.02** ±1.22	**84.74** ±0.55	**92.07** ±0.97	**81.30** ±0.82

Table 2: Standard evaluation results. We report the mean and standard deviation over 5 runs.

PEDP can effectively incorporate more task-relevant information to enrich the response and lead to task success.

We also observe that PEDP performs moderately in terms of average turn. A possible reason is that the model fails to generate denser dialog action combinations. As shown in Table 1, PEDP reaches a comparable F1 score and a higher recall, indicating a lower precision score and implying the incorporation of redundant information that does not contribute to task completion. We report how this affects user experience in the human evaluation.

Standard Evaluation We report the standard performance of the state-of-the-art GDPL and DiaAdv and all the SL-based methods in Table 2.

We observe that our PEDP framework outperforms the others in terms of all evaluation metrics. For the F1 score, PEDP achieves $8.37 \sim 32.74\%$ improvements on account of the substantial improvement in precision ($6.03 \sim 26.89\%$) and recall ($1.31 \sim 36.78\%$) over the baselines.

We also observe that the standard and interactive performance are negatively correlated for the baseline methods (see Table 1). We attribute this to the fact that standard and interactive objectives are fundamentally in conflict. While multiple action combinations can be considered appropriate during dialog interaction, they may not match the ground truth in the test samples. Unlike existing methods, PEDP achieves the best of both worlds, demonstrating our method's effectiveness on MADPL.

Ablation Study We conduct ablation experiments to investigate the effects of two components in our model (Single-action Dialog Planning and Ensemble Prediction) and study how stochastic factors affect model performance (action sampling). We merge the results in Table 1 and 2.

The results show that: 1) The single-action dialog planning module contributes more significantly for performance improvement while the ensemble prediction module effectively stabilizes the model and leads to a lower standard deviation, and please see Appendix C for hyper-parameter sensitivity experiments for the number of paths K. 2) Sampling usually leads to a performance gain of match rate and inform recall but reduces the inform precision, indicating the incorporation of redundant information in the response. For GDPL, sampling harms the performance. A possible reason is that GDPL, which learns from the user simulator for evaluation, has adapted to the simulated dialog scenarios. Sampling instead may lead to sub-optimal decisions.

Dialog pair	**Win**	**Lose**	**Tie**	α
PEDP vs. DiaSeq	41.7	31.3	27.0	0.820
PEDP vs. DiaAdv	36.5	27.6	35.9	0.856
PEDP vs. GDPL	32.6	26.5	40.9	0.839

Table 3: Human evaluation results. We report the mean over 9 judges and Krippendorff's alpha (α) that measures the inter-rater reliability. Typically, results are considered reliable if $\alpha > 0.800$.

Human Evaluation Automatic evaluation only measures part of the agent's performance (e.g., *Success* for the level of task completion). It may fail to consider other aspects for assisting real users (e.g., inappropriate reply). Thus, we conduct a human study to fully evaluate system performance. Our PEDP method is compared with three state-of-the-art dialog policy models: GDPL, DiaAdv, and DiaSeq (reported to offer an elegant user experience [Li *et al.*, 2020]). Following [Takanobu *et al.*, 2019; Li *et al.*, 2020], for each comparison pair, we randomly sample 100 user goals and present the generated dialog samples from the two agents. The dialog actions are translated into human-readable utterances with the language generation module from ConvLab [Zhu *et al.*, 2020]. We hire nine postgraduates as judges to select the dialogs that provide a better user experience (each sample is graded nine times rather than three in existing works [Takanobu *et al.*, 2019; Li *et al.*, 2020]), "Tie" can also be selected for the equally performed ones. We refer the reader to Appendix D and E for the case study and the corresponding error analysis.

In Table 3, we see that our PEDP framework outperforms the baseline models. As mentioned in robust evaluation (see Section 5.4), our PEDP model shares a lower inform precision (0.79) than DiaSeq (0.82) and GDPL (0.87), which indicates the incorporation of redundant information. However, we find that the performance gap does not severely harm the user experience. A possible reason is that human users tend to focus on goal-related information during task-oriented dialog. Compared to dealing with redundant information, they are more intolerant towards the failure of providing the requested information.

5 Conclusion

This paper presents a supervised learning-based PEDP framework, which incorporates model-based planning for conceiving what to express before deciding the current response through simulating single-action dialogs. With multi-task learning of single-action dialog dynamics and multi-action prediction, our method fully uses the training corpus and makes reasonable and explainable decisions, thus improving performance towards unseen task-oriented human-machine dialog flows. Extensive experiments show that our method significantly and consistently outperforms state-of-the-arts with a favorable task success rate of 90.6%.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China (61902305, 61922067), Shenzhen Basic Research Grant (JCYJ20170816100819428), and MoE-CMCC "Artifical Intelligence" Project (MCM20190701).

References

[Budzianowski *et al.*, 2018] Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. MultiWOZ: A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv:1810.00278*, 2018.

[Gašić *et al.*, 2015] M Gašić, N Mrkšić, Pei-hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. Policy committee for adaptation in multi-domain spoken dialogue systems. In *ASRU*, pages 806–812. IEEE, 2015.

[Gordon-Hall *et al.*, 2020] Gabriel Gordon-Hall, Philip John Gorinski, and Shay B Cohen. Learning dialog policies from weak demonstrations. *arXiv:2004.11054*, 2020.

[Huang *et al.*, 2020] Xinting Huang, Jianzhong Qi, Yu Sun, and Rui Zhang. Semi-supervised dialogue policy learning via stochastic reward estimation. *arXiv:2005.04379*, 2020.

[Jang *et al.*, 2016] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv:1611.01144*, 2016.

[Jhunjhunwala *et al.*, 2020] Megha Jhunjhunwala, Caleb Bryant, and Pararth Shah. Multi-action dialog policy learning with interactive human teaching. In *SIGdial*, pages 290–296, 2020.

[Li *et al.*, 2017] Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. End-to-end task-completion neural dialogue systems. *arXiv:1703.01008*, 2017.

[Li *et al.*, 2020] Ziming Li, Julia Kiseleva, and Maarten de Rijke. Rethinking supervised learning and reinforcement learning in task-oriented dialogue systems. *arXiv:2009.09781*, 2020.

[Liu and Lane, 2018] Bing Liu and Ian Lane. Adversarial learning of task-oriented neural dialog models. *arXiv:1805.11762*, 2018.

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

[Peng *et al.*, 2018] Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, Kam-Fai Wong, and Shang-Yu Su. Deep Dyna-Q: Integrating planning for task-completion dialogue policy learning. *arXiv:1801.06176*, 2018.

[Qin *et al.*, 2021] Xiaoyu Qin, Xiaowang Zhang, Muhammad Qasim Yasin, Shujun Wang, Zhiyong Feng, and Guohui Xiao. Suma: A partial materialization-based scalable query answering in owl 2 dl. *Data Science and Engineering*, 6(2):229–245, 2021.

[Rastogi *et al.*, 2020] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *AAAI*, volume 34, pages 8689–8696, 2020.

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

[Shu *et al.*, 2019] Lei Shu, Hu Xu, Bing Liu, and Piero Molino. Modeling multi-action policy for task-oriented dialogues. *arXiv:1908.11546*, 2019.

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

[Suzuki and Ogata, 2020] K. Suzuki and T. Ogata. *Stable Deep Reinforcement Learning Method by Predicting Uncertainty in Rewards as a Subtask*. ICONIP, Proceedings, Part II, 2020.

[Takanobu *et al.*, 2019] Ryuichi Takanobu, Hanlin Zhu, and Minlie Huang. Guided dialog policy learning: Reward estimation for multi-domain task-oriented dialog. *arXiv:1908.10719*, 2019.

[Tian *et al.*, 2020] Shan Tian, Songsong Mo, Liwei Wang, and Zhiyong Peng. Deep reinforcement learning-based approach to tackle topic-aware influence maximization. *Data Science and Engineering*, 5(1):1–11, 2020.

[Wang *et al.*, 2016] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *arXiv:1611.01224*, 2016.

[Wang *et al.*, 2020] Sihan Wang, Kaijie Zhou, Kunfeng Lai, and Jianping Shen. Task-completion dialogue policy learning via monte carlo tree search with dueling network. In *EMNLP*, pages 3461–3471, 2020.

[Wu *et al.*, 2019] Yuexin Wu, Xiujun Li, Jingjing Liu, Jianfeng Gao, and Yiming Yang. Switch-based active deep dyna-q: Efficient adaptive planning for task-completion dialogue policy learning. In *AAAI*, volume 33, pages 7289–7296, 2019.

[Zhang *et al.*, 2020] Zheng Zhang, Ryuichi Takanobu, Qi Zhu, Minlie Huang, and Xiaoyan Zhu. Recent advances and challenges in task-oriented dialog systems. *Science China Technological Sciences*, pages 1–17, 2020.

[Zhang *et al.*, 2021] Shuo Zhang, Junzhou Zhao, Pinghui Wang, Nuo Xu, Yang Yang, Yiting Liu, Yi Huang, and Junlan Feng. Learning to check contract inconsistencies. In *AAAI*, volume 35, pages 14446–14453, 2021.

[Zhu *et al.*, 2020] Qi Zhu, Zheng Zhang, Yan Fang, Xiang Li, Ryuichi Takanobu, Jinchao Li, Baolin Peng, Jianfeng Gao, Xiaoyan Zhu, and Minlie Huang. Convlab-2: An open-source toolkit for building, evaluating, and diagnosing dialogue systems. *arXiv:2002.04793*, 2020.