Efficient Dialog Policy Learning with Hindsight, User Modeling, and Adaptation

Keting Lu*, Yan Cao*, Xiaoping Chen, Shiqi Zhang

Abstract—**Goal-oriented dialog systems aim to efficiently and accurately exchange information with people using natural language. A goal-oriented dialog policy is used for suggesting language actions for such dialog systems. Reinforcement learning have been used for computing dialog policies from the experience of language-based interaction. Learning efficiency is particular important in dialog policy learning, due to the considerable cost of interacting with human users, and the potentially very poor user experience from low-quality conversations. In this article, we develop deep reinforcement learning algorithms to improve the efficiency of dialog policy learning. Our contribution is threefold, aiming at the central goal of improving the efficiency of dialog policy learning. First, we present a novel "hindsight" approach to make use of unsuccessful dialog instances to provide the dialog learning agent with extra positive feedback. Second, we introduce user modeling, and enable the dialog agent to learn from simulated interaction experience. Third, we have developed a meta-learning algorithm that enables the dialog agent to adaptively learn from simulated users and hindsight experience at the same time. The threefold contribution altogether, for the first time, enables our dialog agent outperforming a number of state-of-the-art dialog policy learning methods, as demonstrated via our experimental results.**

Index Terms—**Dialog systems, Reinforcement learning, Deep learning, Intrinsically motivated learning, Meta learning**

I. Introduction

GOAL-ORIENTED dialog systems aim at assisting users to accomplish specific goals using natural language, and have been applied to a variety of domains [1], [2], [3], [4], [5], [6]. With the development of big data and deep learning techniques, the goal of creating automatic human-computer dialog systems, as our personal assistant or chat companion, is no longer an illusion. Notable products available on the market include Apple Siri, Microsoft XiaoIce, and Xiaodu DuerOS. Goal-oriented dialog systems typically include three components [7]: a language understanding component for recognizing and parsing the language inputs into inner representations, a belief state tracking component for predicting user intent and updating the dialog history, and a dialog management component that generates dialog actions. The dialog actions can be converted into spoken or text-based language using a language generator. On the one hand, goal-oriented dialog systems favor *concise* conversations; learning a policy for goal-oriented dialog management usually requires a large amount of real-user experience [6]. On the other hand, collecting trial-and-error dialog data from real users is difficult. The two observations together motivate the development of sample-efficient algorithms for goal-oriented dialog policy learning.

Reinforcement learning (RL) algorithms aim at learning action policies from trial-and-error experiences [8], and have been used for learning policies for dialog management in goal-oriented dialog systems [7], [9]. Recently, deep RL methods, e.g. [10], have been developed for dialog policy learning in dialog domains to support large state spaces and complex state representations. Despite all the advances in deep RL, dialog policy learning remains a challenge [11], [12] for at least the following two interdependent reasons.

- On the one hand, to learn a good-quality dialog policy, the dialog agent needs a lot of dialog experience with real users [6], [13]. However, when the dialog policy's quality is low, the user experience is poor, making it hard to gather the dialog experience with real users.
- On the other hand, the dialog agent needs positive feedback (reinforcements) to learn to interact with people. However, it is very hard to get such positive signals in the early learning phase [14], as most dialogs are unsuccessful.

As a result, it is critical to develop **sample-efficient** RL methods for learning high-quality dialog policies with limited conversational experiences. In particular, we aim to make use of the many unsuccessful dialogs and leverage user modeling to provide extra "artificial" reinforcements toward efficient dialog policy learning.

In this article, we develop an algorithm, called **learning with hindsight, user modeling, and adaptation** (LHUA), for sample-efficient dialog policy learning. LHUA enables a dialog agent to simultaneously learn from real, simulated, and hindsight experiences, which identifies the key contribution of this research. Simulated experience is generated using learned user models, and hindsight experience (of successful dialog samples) is generated by manipulating dialog segments and goals of the (potentially many) unsuccessful samples. Dialog experience from simulation and hindsight respectively provide more dialog samples and more positive feedback for dialog policy learning. To further improve the sample efficiency, we develop a **meta-agent** for LHUA that adaptively learns to switch between real and simulated users in the dialog-based interactions, which identifies the second contribution of this research. An overview of LHUA is shown in Figure 1.

LHUA is capable of adaptively adjusting its strategy of selecting between real and simulated users for policy learning. Without external rewards, our dialog agent is motivated to interact with real users more, when the learned user model is

* Equal contribution
Lu is with Baidu Inc (Email: ktlu@mail.ustc.edu.cn); Cao (Email: caotian@mail.ustc.edu.cn) and Chen (Email: xpchen@ustc.edu.cn) are with USTC; Zhang is with SUNY Binghamton (Email: zhangs@binghamton.edu)

Fig. 1. An overview of LHUA. A *dialog agent* interacts with both real and simulated users while learning a dialog policy from this interaction experience. A *simulated user* is modeled using real dialog samples, and interacting with this simulated user provides the dialog agent with simulated dialog samples. An *adaptive coordinator* learns from the dialog agent's recent performance to adaptively assign one user (real or simulated) for the dialog agent to interact with. A *hindsight manager* manipulates both real and simulated dialog samples (of mixed qualities) to "synthesize" successful dialog samples.

less mature. As the user model's quality is getting better, the dialog agent is more leaning toward learning from simulated users.

The adaptation capability of LHUA makes it an *intrinsically motivated* dialog policy learning approach, because the meta-agent is self-motivated, and generates intrinsic signals to guide itself to interact with the environment. Those signals help the dialog agent to switch between real users and simulated users throughout the dialog policy learning process. This switching mechanism enables the dialog agent to obtain suf cient dialog samples with quality assurance. More speci cally, when an LHUA agent starts a dialog, based on a self-evaluation of its dialog performance, the *adaptive coordinator* of LHUA intrinsically decides it should chat with a real user or a learned user model. This self-motivational behavior improves the dialog agent's learning strategy without requiring external guidance.

Experiments were conducted using a realistic dialog simulation platform [4]. LHUA has been compared with state-of-the-art methods [15], [16] in dialog policy learning tasks, as well as with its own ablations. Results suggest that ablations of LHUA produce comparable (or better) performances in comparison to competitive baselines in success rate, and LHUA as a whole performed the best.

This journal article is built on two recent conference papers from the same group of researchers. The rst paper described dialog policy learning with hindsight [14], including an approach called *stitching-based hindsight experience replay* (S-HER), which becomes an ablation of LHUA in this article. The second paper improved S-HER by enabling the user modeling and adaptation functionalities, and developed the initial version of LHUA [17]. Due to the space constraints of the conference papers, we could not have presented the technical details of LHUA (hindsight management, user modeling, and adaptive learning) and experimental results in detail, which serves as the main motivation of developing this journal article. The technical sections have been restructured to present the LHUA algorithm as a whole, including Figs. 1 and 3, Algorithms 1-4, and their descriptions, which together serves as the most important contribution of this journal article, c.f., the two conference papers. In addition, we have rede ned a number of variables to ensure the consistency in terminology, and generated new results (Fig. 7) to ll the gap in evaluation. Altogether, this journal paper, for the rst time, provides an uni ed, principled description, and systematic evaluations of the LHUA algorithm.

The rest of this article is structured as follows. We discuss related works, and identify the differences between our work and those from the literature in Section II. To make readers easily follow our paper, we make a brief introduction of technical background in Section III. After that, we present the main contribution of this article, the LHUA algorithm, in Section IV, as well as a full instantiation in Section V. Experiment setup and experimental results are reported to show the effectiveness of our algorithms in Section VI Finally, we conclude this article and discuss a few promising ways of improving and promoting this line of research in Section VII.

II. Related Work

WE discuss a few families of methods for improving the ef ciency of reinforcement learning, as applied to the problem of dialog policy learning. We qualitatively compare the discussed algorithms from the literature with our LHUA approach.

User Modeling: Researchers have developed "two-step" algorithms for improving the ef ciency of dialog policy learning [18], [19]. Their dialog agents rst builds user models through supervised learning with real conversational data, and then learn dialog policies by interacting with the simulated users. In those methods, user modeling must be conducted of ine before the start of dialog policy learning. As a result, the learned policies are potentially biased toward the historical conversational data. Toward online methods for dialog policy learning, researchers have developed algorithms for simultaneously constructing models of real users, and learning from the simulated interaction experience with user models [20], [21], [22], [23], [24], [4], [25], [26], [6]. Those methods enable agents to simultaneously build and leverage user models in dialog policy learning. However, the problem of learning high-quality user models by itself can be challenging. Our algorithms support user modeling, while further enabling agents to adaptively learn from both hindsight and real conversations.

Experience Replay: Deep Q-Network (DQN) has enabled an agent to achieve human-level performance on playing Atari games with many useful techniques, such as target network and experience replay (ER) [27]. RL algorithms, such as DQN, are generally data intensive and frequently require huge numbers of interactions with the environments. To better use the interaction experience, ER suggests storing and reusing samples at training time, has been widely used for speeding up the RL agent's training process [28], [29].

Prioritized ER (PER) further accelerates training by assigning a weight based on temporal difference error (TD-error) to each sample [30], so as to increase the likelihood of selecting samples with a high TD-error. While PER enables more effective sample selections [31], the applicability of PER is limited when very few successful samples are available. Hindsight experience replay (HER) methods have been developed to manipulate goals based on resulting states [32]. For instance, HER was initially applied to a robot manipulation domain. If the robot arm fails to reach a desired position, HER deems the trail as being successful by changing the original goal. However, the HER method is only applicable to domains where goals are explicit to the agents, e.g., target positions in manipulation tasks. In dialog domains, goals are not fully observable and must be identi ed via language actions, which motivates the development of complex HER methods in this work. In particular, the hindsight manager of our approach was inspired by the HER idea, and generates arti cial "successful" samples to improve the learning rate of dialog agents.

Reward Shaping: Reward shaping has been used for speeding up the learning of dialog policies by adding a new dense reward function [33]. It has been proved that a well designed (dense) reward function does not alter the optimal policy [34]. Recently, [35] applied Recurrent Neural Networks (RNNs) to predict the intermediate rewards for dialogs to reduce training time, and developed an on-line learning framework where dialog policy and reward function were jointly trained via active learning [3]. However, generating such complex reward functions require either considerable human effort or large datasets. Compared with the reward shaping methods, the user modeling component of our approach builds simulated users for generating dialogs in simulation.

Exploration in RL: Another line of research focuses on developing effective exploration strategies for RL algorithms, enabling more sample-ef cient dialog learning. For instance, researchers have integrated Least-Squares Policy Iteration [36] and Fitted-Q [37] for dialog policy optimization [38]. Other researchers have applied Kalman Temporal Differences to estimate Q-function and maintained its uncertainty to actively explore state-action space in dialog management [39]. Other examples include Gaussian Process (GP)-based sample-ef cient dialog learning [40], the Bayes-by-Backprop Q-network (BBQN) [41], and trust-region and gradient-based algorithms [42].

Our methods are orthogonal to those exploration algorithms from the perspective of RL, and have the potential to be combined with the above sample-ef cient RL methods to produce a more ef cient learning process, which is one of the directions to extend our work.

Pre-training in RL: Pre-training has been used in dialog learning for computing an initial policy from a corpus using supervised learning (SL) [43], [44]. After that, a dialog agent can further improve the policy via learning from interactions with users. In line with past research on dialog systems (as listed above), we use pre-training in this work (unless stated otherwise) to give our dialog agent a "warm start".

Intrinsic Motivation: Intrinsic motivation (IM) learning, sometimes known as "curiosity-driven learning," aims to discover how to produce "interesting" effects in the environment driven by self-generated motivational signals not related to speci c tasks or instructions [45]. Systems of IM usually need a mechanism for evaluating the degree of novelty or surprise, which is then used for reward design [46], [47], [48]. IM has been applied to various RL domains, such as Atari games with or without spare extrinsic rewards [49], learning robotic motor skills [50], reaching target objects using a two-armed robot [51]. In addition, the IM ideas have been used to further enhance the performance of existing algorithms for imitation learning [52], representation learning [53], [54], and multi-agent systems [55], [56].

While the main application domain of IM is robotics, researchers have recently investigated the application of IM to dialog systems [5]. In that work, state prediction error is evaluated through the predictions of the next belief state, and is considered an intrinsic reward. Our approach shares the IM spirit, as our dialog agents have a task-independent mechanism for self-motivation. Different from existing IM methods that use intrinsic rewards to guide exploration behaviors, our dialog agent is able to generate intrinsic dialog goals (potentially different from the original goals). In case the dialog agent fails in accomplish user goals, the agent uses intrinsic goals to synthesize successful "arti cial" dialogs for learning purposes.

To our best knowledge, LHUA (our dialog policy learning algorithm) is the rst of such that enables dialog agents to simultaneously learn from real, simulated, and hindsight experiences. Further, its performance is enhanced through a meta-learner that learns to switch between interacting with real and simulated users at runtime. Next, we briefly discuss some background knowledge of this research before presenting the key contribution of this article – the LHUA algorithm.

III. Background

In this section, we briefly introduce the two building blocks of this work, namely Markov decision process (MDP)-based dialog management, and Deep Q-Network (DQN).

A. MDP-based dialog Management

Dialog control is modeled using MDPs in this work. An MDP-based dialog manager [41] can be described as a tuple $\langle S, A, T, s_0, R \rangle$. S is the state set, where $s \in S$ represents the agent's current dialog state including the agent's last action, the user's current action, the distribution of each slot, and other domain variables as needed. A is the action set, where $a \in A$ represents the agent's response. T is the stationary, probabilistic transition function with conditional density $p(s_{t+1} | s_t, a_t)$ that satisfies the (first-order) Markov property. $s_0 \in S$ is the initial state. $R : S \times A \to \mathbb{R}$ is the reward function, where the agent receives a big bonus in successful dialogs, and has a small cost in each turn.

Solving an MDP-based dialog management problem produces π, a dialog policy. A dialog policy maps a dialog state to an action, $\pi : S \to A$, toward maximizing the discounted, accumulative reward in dialogs, i.e., $R_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$, where $\gamma \in [0, 1]$ is a discount factor that specifies how much the agent favors future rewards.

B. Deep Q-Network

Deep Q-Network (DQN) [27] is a model-free RL algorithm for discrete action space. DQN uses a neural network as an approximation of the optimal Q-function, $Q^* = Q(s, a; \theta)$, where a is an action executed at state s, and θ is a set of parameters. Its policy is defined either in a greedy way: $\pi_Q(s) = \arg\max_{a \in A} Q(s, a; \theta)$ or being ϵ-greedy, i.e., the agent takes a random action in probability ϵ and action $\pi_Q(s)$ otherwise. The loss function for minimization in DQN is usually defined using TD-error:

$$L = \mathbb{E}_{s,a,r,s'}[(Q(s, a; \theta) - y)^2], \qquad (1)$$

where $y = r + \gamma \max_{a' \in A} Q(s', a'; \theta^-)$.

Naive DQNs often suffer from overestimation and instability, and two techniques are widely used to alleviate the issues. One is called *target network* [27] whose parameters are updated by θ once every many iterations in the training phase. The other technique is *experience replay* [57], [27], where an experience pool E stores samples, each in the form of (s_t, a_t, r_t, s_{t+1}). In training the DQN, a mini-batch is uniformly sampled from E. Experience replay desensitizes DQN to the correlation among the samples, and increases the data efficiency via reusing the (potentially expensive) samples. Both techniques improve the performance of DQN and are used in this paper.

C. Hindsight Experience Replay

Hindsight experience replay (HER) was originally developed for multi-goal RL tasks with sparse rewards [32]. HER enables an agent to augment learning experience by replaying trajectories with a goal that is different from the goal that the agent was originally trying to achieve. This simple modification for post-processing RL agents' interaction experiences has been shown effective for sample-efficient learning from sparse rewards. Technically, given an experienced trajectory,

$$s_0, s_1, s_2, \dots, s_T,$$

and the corresponding goal state s_g, each transition $s_t \to s_{t+1}$ (of a particular trajectory) is associated to not only the original goal s_g, but also a subset of other goal states s_l (of the same trajectory), where $t + 1 < l < T$. Those additional goals can help generate many new, “successful” trajectories. Researchers have found that, “Not only does HER improve the sample efficiency in this setting, but more importantly, it makes learning possible even if the reward signal is sparse and binary” [32]. HER uses hindsight to leverage intermediate states to generate artificial (sub)goals, serving additional positive feedback. As a result, those states in new (synthetic) trajectories can potentially provide rewards denser than those of the states as in the original trajectory. Thanks to this characteristic, HER algorithms can be used for the implicit development of curriculum as those artificially generated goals could be designed in such a way that their complexity increases over time [32].

IV. Algorithm for Learning with Hindsight, User Modeling, and Adaptation

In this section, we formally introduce our proposed method, called learning with hindsight, user modeling, and adaptation (LHUA). Due to the algorithm complexity, we start with the subsystems of LHUA before presenting the whole algorithm. In particular, we first introduce learning with hindsight (LH), then learning with hindsight and user modeling (LHU), and finally present LHU with Adaptation (LHUA).

A. Concepts and Definitions

Goal-oriented dialog systems help users accomplish their goals via dialog, where a goal G includes a set of constraints C and a set of requests R: $G = (C, R)$ [58].

Consider a movie booking domain. A user may ask about the *name* and *time* of a movie starring *Jackie Chan*, of genre *action*, and running *today*, where the goal is in the form of:

$$\text{Goal} = \left(\begin{array}{l} C = \begin{bmatrix} \text{actor} = \text{Jackie Chan} \\ \text{genre} = \text{action} \\ \text{date} = \text{today} \end{bmatrix}, \\ R = \begin{pmatrix} \text{movie name} = \\ \text{start time} = \end{pmatrix} \end{array} \right)$$

Definition 1. *Subgoal:* Given $G = (C, R)$ and $G' = (C', R')$, we say G' is a subgoal of G, or $G' \sqsubset G$, if $C' \subseteq C$ and $R' \subseteq R$, where $G' \neq \emptyset$.[1]

[1]Our definition of subgoal is different from that in the dialog learning literature on hierarchical reinforcement learning [59] and state space factorization [60].

In successful dialogs, the agent must correctly identify all constraints and requests in G, so as to correctly provide the requested information via querying a database.

Figure 2 shows three example subgoals (out of many) in the movie booking example. It should be noted that some subgoals do not make sense to humans, but can be useful for dialog learning. For instance, Subgoal 2 corresponds to a query about *movie name* and *start time* without any constraints. Real users do not have such goals, but an agent can still learn from the experience of achieving such subgoals. Continuing the "Jackie Chan" example, if the agent misidentifies *genre*, *start time*, or both, the dialogs will be deemed unsuccessful, meaning that the agent cannot learn much from it, even though the agent has correctly identified the other entries of *movie name*, *actor*, etc. In this work, we make use of such unsuccessful dialogs in the training process, leveraging the fact that the agent has achieved subgoals (not all) in these dialogs.

Given dialogs D' and D, we say D' is a segment of D, if D' includes a consecutive sequence of turns of D. We introduce the concept of assessment function, $success(G, D)$, that outputs *true* or *false* representing whether dialog D accomplishes goal (or subgoal) G or not. Using the assessment function, we define the validity of dialog segments.[2]

Definition 2. *Validity of dialog segments:* Given dialog D, goal G, and dialog segment D' (of D), we say D' is a valid dialog segment of D, iff there exists a subgoal $G' \sqsubset G$, and $success(G', D')$ is *true*.

Using Definitions 1 and 2, one can assess the validity of dialog segment D', using the entire dialog D, the goal of this dialog G, and the provided subgoal G'. However, there exist many subgoals of the ultimate goal (combinatorial explosion), and it soon becomes infeasible to assess the validity of a dialog segment. Formally, given goal $G = (C, R)$, the number of subgoals is $\sum_{i=0}^{|C|}\binom{|C|}{i} \cdot \sum_{j=0}^{|R|}\binom{|R|}{j} - 2$, where we subtract the two extreme cases of the subgoal being $\emptyset$ and G. If $|C| = 5$ and $|R| = 5$, the number of subgoals is 1598.

Instead of assessing the validity of a dialog segment using all subgoals, we aim at using only the ones with the so-far-highest "cardinality". In line with previous research on dialog policy learning, e.g., [18], [19], [58], including the baseline methods used in experiments, we assume users are *cooperative* and *consistent* in dialogs, meaning that

1) During a dialog, the number of constraints and requests that have been identified is monotonically increasing; and

2) When a subgoal is accomplished, all its "subsubgoals" are automatically accomplished.

Definition 3. *Stitchability:* Consider two dialog segments

$$D = (s_0, a_0, s_1), (s_1, a_1, s_2), \cdots, (s_{M-1}, a_{M-1}, s_M),$$

$$D' = (s'_0, a'_0, s'_1), (s'_1, a'_1, s'_2), \cdots, (s'_{N-1}, a'_{N-1}, s'_N),$$

[2]Such assessment functions are provided by dialog simulators, e.g., TC-Bot (https://github.com/MiuLab/TC-Bot/blob/master/src/deep_dialog/usersims/usersim_rule.py#L228).

$$\text{Subgoal 1} = \left(C = \begin{pmatrix} actor = \text{Jackie Chan} \\ genre = \text{action} \end{pmatrix}, \; R = \emptyset \right);$$

$$\text{Subgoal 2} = \left(C = \emptyset, \; R = \begin{pmatrix} \text{movie name} = ? \\ \text{start time} = ? \end{pmatrix} \right);$$

$$\text{Subgoal 3} = \left(C = \begin{pmatrix} actor = \text{Jackie Chan} \\ date = \text{today} \end{pmatrix}, \; R = \begin{pmatrix} \text{movie name} = ? \\ \text{start time} = ? \end{pmatrix} \right);$$

Fig. 2. Example subgoals in the movie booking domain. For instance, Subgoal 3 corresponds to the request of "Please tell me the name and start time of the movie that is playing today and stars Jackie Chan."

where M and N are turn numbers (dialog segment lengths) of D and D', and each turn includes initial state, dialog action, and resulting state.

If the dialogs' corresponding (sub)goals are G and G', we say D and D' are stitchable, if and only if $G = G'$, and

$$D_{KL}(s_M \,\|\, s'_0) \leq \epsilon;$$

where $\epsilon \in \mathbb{R}$ is a stitchability threshold.

In the above definition, $D_{KL}(s_M \,\|\, s'_0)$ is the KL Divergence between the two dialog belief states (s_0 and s_1), each in the form of a probability distribution:

$$D_{KL}(s_0 \| s_1) = \sum_{i=0}^{n} s_0(i) \cdot \log \frac{s_0(i)}{s_1(i)}$$

B. Hindsight Experience Generation

In this subsection, we describe how to improve dialog agents' learning efficiency through hindsight experience replay (HER). While HER is not new [32], the application of the HER idea to dialog policy learning is new (first appeared in our conference paper [14]). The main idea of HER for dialog learning is to *manipulate dialog segments* of mixed qualities *to synthesize successful dialogs*. Accordingly, there are two key questions that must be addressed: 1) How to generate the dialog segments? and 2) How to use the segments to synthesize artificial dialogs? We answer the above-mentioned two questions in the remainder of this subsection.

Dialog Segmentation: Algorithm 1 describes how our *dialog segmentation* works. First, we initialize two collections P and Q. P stores the constraints and requests that have been identified during the dialogs, and Q stores the ones that have not been identified. Therefore, at the very beginning of dialogs, P is an empty set, and Q stores all constraints and requests. After each dialog turn, we generate a subgoal set $\mathcal{G}$ where all subgoals share the same (so far highest) cardinality. Formally, $\mathcal{G} = \{G' \mid G' = P \cup \{q\}, \forall q \in Q\}$. If $G' \in \mathcal{G}$ is accomplished by the current dialog segment, the corresponding q is removed from Q and added into P, which is used to generate the subgoal set for the next dialog segment. So at each dialog turn, only a small set of subgoals (i.e., $\mathcal{G}$) is used to assess a dialog segment. Compared to exhaustive subgoal identification that

Algorithm 1 Dialog Segmentation

```
Input:
    Goal $G$ that includes constraint set $C$ and request set $R$;
    Assessment function $success(\ ;\ )$;
Output:
    A collection of pairs of a valid (head) dialog segment and a correspond-
    ing subgoal, ;
 1: Initialize  = ;
 2: Initialize $P = \ ;$ and $Q = C [ R$
 3: while Dialog $D$ is not ended do
 4:     Outcome  ag $segment\_outcome = False$
 5:     for $q$ 2 $Q$ do
 6:         Construct a subgoal $G^0 = P [ q$
 7:         if $success(G^0; D)$ then
 8:             $segment\_outcome = True$
 9:             $P$   $P [ q$ and $Q$   $Q n q$
10:         end if
11:     end for
12:     if $segment\_outcome == True$ then
13:            [ $<D; P>$
14:     end if
15: end while
```

Algorithm 2 Hindsight Experience Generation (HEG)

```
Input:
    Dialog $D$, and its goal $G$
    A collection of pairs of a valid (tail) dialog segment and a corresponding
    subgoal,
    KL-divergence threshold ",
    Assessment function, $success(\ ;\ )$
Output:
    A collection of stitched dialogs $D^{st}$
    An updated collection of pairs of a valid (tail) dialogue segment and a
    corresponding subgoal,
 1: Assess input dialog (either from real users or simulated users):
            $success\_ag = success(G; D)$
 2: Call Algorithm 1 to compute , using $D$, $G$, and $success(\ ;\ )$, where
       is a collection of pairs of a valid (head) dialog segment and a
    corresponding subgoal
 3: for $< D^0; G^0 >$ 2   do
 4:     if $success\_ag$ is $True$ then
 5:            [ $< D$  $D^0; G^0 >$, where   is a dialog subtraction operator

 6:     else
 7:         for $< D^{00}; G^{00} >$ 2   do
 8:             if $G^0$ is $G^{00}$ and $D_{KL}(D^0 jj D^{00})$   " then
 9:                 $D^{stitched} = concatenate(D^0; D^{00})$
10:                 $D^{st}$   $D^{st}$ [ $D^{stitched}$
11:                 Break
12:             end if
13:         end for
14:     end if
15: end for
16: Return $D^{st}$,
```

suffers from combinatorial explosion, our dialog segmentation algorithm has $O(jCj + jRj)$ time complexity.

We use *head (tail) dialog segment* to refer to the segment that includes the early (late) turns in the resulting dialog. In Definition 3 (Section IV-A), D and D^0 are head and tail segments respectively. Further, a head segment set consists of head dialog segments D with the corresponding completed subgoal G_{sub},

$$= f(D; G_{sub})g \tag{2}$$

and a tail segment set consists of tail dialog segments D^0 with the corresponding completed subgoal G^0_{sub}.

$$= f(D^0; G^0_{sub})g \tag{3}$$

Next, we describe how to use the *head* dialog segments (stored in set) and the *tail* dialog segments (stored in set) to synthesize successful dialogs.

Generating Hindsight Experience: Algorithm 2 presents our algorithm, called Hindsight Experience Generation (HEG). Given a new user dialog D, HEG first calls Algorithm 1 to generate all valid dialog segments that start from the beginning of D, and stores them in set . After that, HEG assesses D's successfulness using its associated goal, and stores the result in $success\_ag$. If successful, HEG subtracts segments in from D to generate the tail segments that eventually lead to successful dialogs. The tail segments are used to augment .

 in Line 5 is a dialog subtraction operator, i.e., D D^0 produces a dialog segment by removing D^0 from D, and this operation is valid, only if D^0 is a segment of D. If the new user dialog is unsuccessful, HEG concatenates one head segment from to one tail segment from to form a new, successful dialog $D^{stitched}$, and add it into set D^{st}, which is also used for storing the algorithm output.

Remarks: HEG for generating successful dialog samples is inspired by Hindsight Experience Replay (HER) [32]. However, the original HER method [32] is not applicable to dialog domains, because goals in dialogs are not explicitly given (c.f., path planning for robot arms) and must be identified in dialog turns. We say HEG is a "complex" HER approach, because HEG manipulates dialog experiences based on the resulting dialog states (which is more difficult than goal manipulation). HEG generates successful, artificial dialogs using dialogs from users (successful or not), and are particularly useful in early learning phase, where successful dialogs are rare.

C. Learning with Hindsight and User Modeling (LHU)

In dialog policy learning, dialog agents can learn from interactions with real users, where the generated real experience is stored in reply buffer B^R. To provide more experience, we develop a simulated user for generating simulated dialog experience to further speed up the learning of dialog policies. The simulated user is of the form:

$$s^0; r \quad M(s; a; \ _M)$$

where, $M(s; a; \ _M)$ takes the current dialog state s and the last dialog agent action a as input, and generates the next dialog state s^0, and reward r. M is implemented by a Multi-Layer Perceptron (MLP) parameterized by $_M$, and refined via stochastic gradient descent (SGD) using real experience in B^R to improve the quality of simulated experience.

Simulated experience generated from interactions between the dialog agent and the simulated user is stored in the simulated replay buffer B^S, which is also manipulated by the hindsight manager to synthesize hindsight experience. Based on hindsight manager and user modeling, we next present the learning process of LHU, where our dialog agent interacts with a real user for one dialog, and a simulated user for k dialogs. In addition to parameter k, there is a KL-divergence threshold as a part of the input. We refer to this algorithm using LHU(k).

Algorithm 3 LH with User Modeling (LHU)

```
Input: K, the times of interactions with the simulated user; KL-divergence threshold "
Output: the success rate $SR^{Dlg}$, and average rewards $R^{Dlg}$ of agent$^{Dlg}$; $Q( )$ for agent$^{Dlg}$
 1: Initialize $Q(s; a; _Q)$ of agent$^{Dlg}$ and $M(s; a; _M)$ of the simulated user via pre-training on human conversational data
 2: Initialize experience replay buffers $B^R$ and $B^S$ for the interaction of agent$^{Dlg}$ with real and simulated users
 3: Initialize tail dialog segment set:
                                        ;
 4: Collect initial state, s, by interacting with a real user following goal $G^{Real}$
 5: Initialize $D^{Real}$ ;   for storing dialog turns (real)
 6: while $s \notin term$ do                          // Start a dialog with real user
 7:     $D^{st}$ ;
 8:     Select a  argmax$_{a^0} Q(s; a^0; _Q)$, and execute a
 9:     Collect next state $s^0$, and reward r
10:     Add dialog turn $d = (s; a; r; s^0)$ to $B^R$ and $D^{Real}$
11:     Call Algorithm 2 to compute $D^{st}$, and update , using $D^{Real}$, $G^{Real}$, and "
12:     Add turns of $D^{st}$ into $B^R$
13:     s   $s^0$
14: end while
15: for k = 1 : K do                         // K interactions with simulated user
16:     Sample goal $G^{Sim}$, and initial state s
17:     Initialize $D^{Sim}$ ;   for storing dialog turns (sim)
18:     while $s \notin term$ do                    // The $k^{th}$ dialog with sim user
19:         $D^{st}$ ;
20:         a  argmax$_{a^0} Q(s; a^0; _Q)$, and execute a
21:         Collect next state $s^0$, and reward r from $M(s; a; _M)$
22:         Add dialog turn $d = (s; a; r; s^0)$ to $B^S$ and $D^{Sim}$
23:         Call Algorithm 2 to compute $D^{st}$, and update , using $D^{Sim}$, $G^{Sim}$,  and "
24:         Add turns of $D^{st}$ into $B^S$
25:         s   $s^0$
26:     end while
27: end for
28: Calculate the success rate $SR^{Dlg}$ and average rewards $R^{Dlg}$ of total interactions
29: Randomly sample a minibatch from both $B^R$ and $B^S$, and update agent$^{Dlg}$ via DQN     // agent$^{Dlg}$ training
30: Randomly sample a minibatch from $B^R$, and update simulated user
31: return $SR^{Dlg}$, $R^{Dlg}$, $Q( )$                       // User modeling
```

Algorithm 3 starts with an initialization of the dialog agent's real and simulated experience replay buffers (B^R and B^S respectively), the model of the simulated user ($M(_M)$), and a tail segment sets for hindsight manager (i.e.,). In the first while loop (starting in Line 6), the dialog agent interacts with a real user and stores the real experience in B^R. Then, k dialogs with the simulated user are conducted in the for loop, where simulated experience is stored in B^S. During interactions with both real and simulated users, stitched dialogs and the tail segment set are simultaneously collected and updated (Lines 11 and 23), and if the set of stitched dialogs D^{st} is not empty, all turns of stitched dialogs are added into corresponding replay buffer (Lines 12 and 24). Finally, the dialog agent is trained on B^R and B^S, and the simulated user is trained on B^R. Note that we use stitching-based HER as our hindsight manager's implementation since it has better performance than trimming-based HER, which can be evaluated in experiments.

LH and LHU: The red lines of Algorithm 3 (Lines 15-27) correspond to the interactions with simulated users. We refer to the version of Algorithm 3 with Lines 15-27 commented out, as Learning with Hindsight (LH). In experiments, we evaluate LH as a subsystem (ablation) of LHU.

Algorithm 4 LHU with Adaptation (LHUA)

```
Input: H, the max length of adaptation episode; KL-divergence threshold; N, training times
Output:  , the dialog policy;
 1: Initialize $A(s^A; k; _A)$ of agent$^{Adp}$, and replay buffer $B^A$ as empty
 2: for i = 1 : N do
 3:     Initialize adaptation state $s^A$ using Eqn. 4
 4:     Initialize turn counter h: h = 0
 5:     while h  H do
 6:         Select action k: k  argmax$_{k^0} A(s^A; k^0; _A)$
 7:         Execute action k: Call Alogrithm 3 to compute $SR^{Dlg}; R^{Dlg}; Q( )$, using k;
 8:         Collect reward $r^A$ via Eqn. 5, and next adaptation state $\hat{s}^A$ using Eqn. 4
 9:         $B^A$  $B^A$ [ $(s^A; k; r^A; \hat{s}^A)$, $s^A$  $\hat{s}^A$, and h  h + 1
10:     end while
11:     Sample a minibatch from $B^A$, and update $_A$ via DQN
12: end for
13: for all $s \in S$: ( s)  argmax$_{a^0} Q(s; a^0; _Q)$
14: return ( )
```

The output of Algorithm 3 is used in the next section, where we introduce how to further enable the dialog agent to learn a meta-policy for adaptively determining which user (real or simulated) to interact with.

D. LHU with Adaptation (LHUA)

Adaptively determining which user (real or simulated) the LHU agent should interact with can further speed up the dialog policy learning process. The idea behind it is that, if a simulated user can generate high-quality, realistic dialog experience, interactions with the simulated user should be encouraged. To enable this adapative "switching" behaviors, we develop an *adaptive coordinator* that learns a meta-policy for selecting between real and simulated users for collecting interaction experience. We learn this adaptive coordinator using reinforcement learning, producing the LHUA algorithm, which is described next.

Meta-agent for Adaptation Learning: In each turn of interaction with the LHU agent, *adaptive coordinator* updates the adaptation state s^A using the equation below:

$$s_i^A = \begin{cases} [0; 0; 0; 0] & i = 0 \\ [SR_i; R_i; SR_i - SR_{i-1}; R_i - R_{i-1}] & i > 0 \end{cases} \quad (4)$$

where SR_i and R_i are respectively average success rate and rewards from LHU agent's training performance at i^{th} episode. In practice, R is normalized to have values between 0 and 1, same as SR. This form of adaptation state provides accessible information on different training phrases to represent LHU agent 's current performance.

Based on the state s^A, *adaptive coordinator* chooses action k to determine, after each dialog with the real user, how many dialogs should be conducted with the simulated user. The value of action k ranges from 1 to K. Adaptive coordinator receives immediate rewards after executing an action k, i.e. LHU(k), each time. We use success rate increment of LHU agent to design the reward function, as shown below:

$$r_i^A = \frac{SR_i - SR_{i-1}}{SR_i} \cdot \frac{k_i}{L_i} \quad (0 < i \leq H) \tag{5}$$

where k_i is the i^{th} action chosen by *adaptive coordinator*, and L_i means the total number of times of interactions with both real and simulated users, i.e., $L_i = \sum k_i + 1$. Reward is continuously harvested, until the H^{th} turn.

Instantiation of Adaptive Coordinator: Due to the continuous state space, the approximated value function of *adaptive coordinator* is implemented using a two-layer fully connected neural network, $Q^A(s^A; k; \theta_A)$, parameterized by θ_A. Interactions between the *adaptive coordinator* and the LHU agent start with an initial state. In each turn, the *adaptive coordinator* obtains the state s^A using Eqn. 4, and selects the action k via ϵ-greedy policy to execute. Then, the current training performance of LHU agent is used for acquiring the reward r^A using Eqn. 5, and updating the next state s'^A. Finally, the experience $(s^A; k; r^A; s'^A)$ is stored for meta-policy learning. We improve the value function by adjusting θ_A to minimize the mean-squared loss function.

The LHUA Algorithm: Algorithm 4 presents the dialog policy learning process, where our dialog agent adaptively learns from both simulated and real users. In addition to parameter ϵ for KL-divergence threshold, there is parameter H representing the length of one episode for adaptive coordinator as a part of the input. Algorithm 4 starts with an initialization of replay buffer B^A for adaptive coordinator, and the value function $Q^A(s^A; k; \theta_A)$. Before the start of each episode, a turn counter h is initialized as zero for turn counting. Adaptive coordinator interacts with LHU agent for H turns while collecting and saving experience in B^A. At the end of each adaptation episode, we use DQN to update θ_A.

LHUA enables the dialog agent to simultaneously learn from the dialogs with both real and simulated users. At the same time, *hindsight manager* manipulates both real and simulated dialog samples to synthesize more successful dialog samples. The *adaptive coordinator* is learned at runtime for adaptively generating self-motivational signals, in the form of a probability that the agent interacts with simulated users, in the dialog policy learning process. This intrinsically motivated adaptive behavior enables the dialog agent to further improve its performance in sample efficiency. For instance, when the quality of the learned user model is poor, the intrinsic signals tend to encourage the dialog agent to interact with real users.

E. Summary of Section IV

Given the length and importance of this section, it is necessary to summarize the key points made in this section. The key contribution of this article is an algorithm call LHUA. Due to the complexity of LHUA, we present the algorithm using a bottom-up strategy. We first describe the "H" component for learning from hindsight experiences (Section IV-B). With hindsight, the agent learns from both real interaction experience and the synthetic experience generated by our novel hindsight manager. The "U" component for user modeling is described in Section IV-C, which further enables the dialog agents to learn from simulated dialog experiences. In learning from simulated users, there is the fundamental exploration-exploitation trade-off between the quality of user models, and the amount of simulated experience. More specifically, to ensure the user model's quality, we want to delay the process of learning from simulated users, whereas, to increase the amount of simulated dialog experience, we want to interact with the simulated users as early as possible. Finally, the "A" component for adaptation learning aims to address this fundamental trade-off by learning to dynamcially adjust the strategy of switching between real and simulated users.

V. Algorithm Instantiation

THE algorithms presented in Section IV are generally applicable to goal-oriented dialog policy learning domains. In this section, we aim to provide the practitioners of dialog systems with technical details of system implementations.

A. Simulation Platform

All experiments were evaluated using a realistic dialog simulation environment, where a dialog agent communicates with simulated users on tasks of *booking movie tickets* [19], [4]. This movie-booking domain consists of 29 slots of two types. One type is on *search constraints* (e.g., number of people, and date), and the other is on *system-informable* properties that are needed for database queries (e.g., critic rating, and start time). The two types of dialog slots are formally defined in Section IV-A. A *dialog state* consists of five components:

1) one-hot representations of the current user action and mentioned slots;
2) one-hot representations of last system action and mentioned slots;
3) the belief distribution of possible values for each slot;
4) both a scalar and one-hot representation of the current turn number; and
5) a scalar representation indicating the number of results which can be found in a database according to current search constraints.

The system (dialog agent) has eleven *dialog actions* representing the system intent. These actions can be mapped into a natural language response according to the dialog belief states and heuristic rules. Once a dialog reaches the end, the system receives a big bonus 80, if it successfully helps users book tickets. Otherwise, it receives -40 as a penalty. In each turn, the system receives a fixed reward -1 (a cost of 1) to encourage shorter dialogs. The maximum number of dialog turns allowed is 40, where a dialog is deemed unsuccessful once this maximum is reached. Our previous work has studied how such rewards affect dialog behaviors. For instance, higher success bonus and/or lower failure penalty encourage more risk-seeking behaviors, while higher QA costs result in less accurate, shorter conversations [2]. Those parameters are intentionally left unchanged in this article.

The DQN architecture used in this paper is a single-layer neural network of size 80 (default setting of [19]), and greedy policy is used, where is initialized with 0.3, and decayed to 0.01 during training. The output layer of DQN has 11 units corresponding to the same number of dialog actions. The techniques of target network and experience replay are applied (see Section II). The size of experience pool is 100k, and experience replay strategy is uniform sampling. The settings of DQN are standard and well in line with references from the literature.

B. User Modeling

The simulated user model, $M()$, is a multi-task neural network [61], and contains two shared hidden layers and three task-speci c hidden layers, where each layer has 80 nodes. Stitching threshold of *hindsight manager* is set 0.2, which is based on an observation from a threshold experiment for stitching-based HER method. The policy network of *adaptive coordinator* is a single-layer neural network of size 64. Parameters k and H are described in Algorithm 4, and have the value of $k = 20$ and $H = 8$.

C. Two Versions of Hindsight Manager

We have implemented two versions of the hindsight manager that provide different trade-offs between computational ef ciency and learning ef cacy. The two versions are distinguished based on whether the tail segments (de ned in Section IV-A) are used for synthesizing arti cial dialogs. The rst version fully exploits the hindsight management capability described in Section IV-B. The second version (still an LH algorithm) does not use tail segments, but only uses the head segments to generate new, short, successful dialogs. The second version is referred to as "LH Light" to highlight its lightweight feature. To distinguish from LH Light, we refer to the original LH method using "LH Full". By default, LH and LH Full are used interchangeably unless suggested otherwise.[3]

Next, we use a case study to illustrate the difference between LH Full and LH Light before experimentally evaluating our hypotheses in the next section.

Case Illustration: As shown in Figure 3, the goal of the unsuccessful dialog (Left) includes three constraints (*start time*, *city*, and *movie name*), and one request of *ticket*. The goal of successful dialog (Right) includes six constraints (*start time*, *city*, *movie name*, *theater*, *date*, and *number of people*), and one request of *ticket*. Next, we demonstrate how the full and light versions of the hindsight manager use the two dialogs to synthesize arti cial, successful dialog instances.

LH Light selects valid dialog segments according to the achieved subgoals. On the left, the dialog segment in blue color achieves the subgoal that includes three constraints (*start time*, *city*, and *movie name*), and also its "subsubgoals". Accordingly, LH Light can generate three valid dialog segments (new, successful dialogs), including the ones from T_0 to T_4, from T_0 to T_5, and from T_0 to T_8. Turns after T_8 are not considered, because our assessment function does not allow the agent asking useless questions.

LH Full generates new successful dialogs by stitching head and tail dialog segments. In this example, the two dialog segments in blue color achieved the same subgoal, although the two dialog segments have very different ows. LH Full enables our agent to generate a new successful dialog by stitching the dialog segment in blue color on the left, and the dialog segment in green color on the right. The newly generated, successful dialog achieves the goal that is originally from the dialog on the right.

[3]In our previous work, those two Hindsight Experience Replay versions were referred to as Trimming HER (T-HER) and Stitching HER (S-HER), respectively [14]. Those HER methods are considered ablations of the LHUA algorithm from this article, and are renamed accordingly.

VI. Experiment

GIVEN the full instantiation described in Section V, we focus on evaluating the performance of LHUA by comparing it to competitive baselines from the literature and its ablation methods. Here we use moving average to smooth all learning curves in experiments.

A. Hypotheses and Experimental Setup

In order to comprehensively demonstrate the bene ts and improvements brought by the proposed algorithm, LHUA, we conducted a lot of experiments, from a horizontal and vertical perspective. From a horizontal perspective, we want to evaluate the most important hypothesis, namely that LHUA outperforms many exists state-of-the-art RL based algorithms in the goal-oriented dialog task. Beside this high-level hypothesis, we have done many experiments from a vertical perspective to explain the improvements brought from different components of LHUA, i.e., user modeling, hindsight manager and adaptive coordinator. Speci cally, we conducted an ablation test to show the effects of different components.

Each experiment includes 1000 epochs. Each epoch includes 100 dialog episodes. By the end of each epoch, we update the weights of target network using the current behavior network, and this update operation executes once every epoch. Every data point in the gures is an average of 500 dialogs from 5 runs (100 dialogs in each run). For instance, a data point of 0.6 success rate at episode 70 means that 60% of the 70th dialog episodes (500 in total) were successful. In line with previous research [43], [44], we use supervised learning to give our dialog agent a "warm start" in each set of experiments, unless speci ed otherwise.

B. Evaluations of LHUA and Its Ablations

LHUA vs. Baselines Our key hypothesis is that adaptively learning from real, simulated, and hindsight experiences at the same time performs better than baselines from the literature. To evaluate this hypothesis, we have selected two strong baselines for goal-oriented dialog policy learning, including DDQ [16], D3Q [6]. In implementing the DDQ agent, the ratio of interaction experiences between simulated and real users is ten, which is consistent with the original implementation [16].

Fig. 3. Examples of an unsuccessful dialog and a successful dialog, where we use “U” and “S” to indicate user and system turns respectively.

Fig. 4. The performances of LHUA (ours), and two strong methods, including DDQ [16], D3Q [6]. We see that, except for the very early phase (rst 50 episodes), LHUA outperformed all baselines.

Figure 4 presents the key results of this research on the quantitative comparisons between LHUA and two strong baselines. We can see that, except for the very early learning phase, LHUA performed consistently better than other methods. In particular, LHUA reached the success rate of 0.75 after about 70 episodes, whereas none of the baselines were able to achieve comparable performance within 150 episodes.

Ablations of LHUA Results reported in Figure 4 have shown the advantage of LHUA over the two strong baseline methods. However, it is still unclear how much each component of LHUA contributes to its performance. We removed components from LHUA, and generated four different ablations of LHUA, including DQN, DDQ (LU, or Learning with User modeling), LH Full, LHU, and LHUA. Since DDQ is learning with user modeling, so we refer to LU as DDQ in this ablation experiment.

Figure 5 shows the ablation experiment's results. From the results, we see that LHUA performed much better than no-hindsight (LU), and no-user-modeling (LH Full) ablations. When both “hindsight” and “user modeling” are activated, there is LHUA's ablation of LHU, which performed better than all the other ablations. LHU still cannot generate comparable performance, c.f., LHUA, which justi ed the necessity of the adapative coordinator.

Fig. 5. Comparisons between LHUA and its ablations: DQN (no hindsight manager, no user modeling, and no adaptive coordinator), LU (no hindsight manager, and no adaptive coordinator), LH Full (no user modeling, and no adaptive coordinator), and LHU (no adaptive coordinator). A complete LHUA includes all the components, including DQN (for naive dialog policy learning), hindsight manager, user modeling, and adaptive coordinator.

Fig. 6. In extreme situations (cold start and small experience pool), LH Full performed better than LH Light in learning rate, and the combination of LH Full and LH Light performed the best.

C. Hindsight Management Strategies

These experiments aimed to evaluate two hypotheses mentioned before, Hypothesis-I: whether or not using tail segments, we assume that both head segments and tail segments are useful for accelerating learning rate of training, and Hypothesis-II: effects of different stitching thresholds, we assume that too small or too large stitching threshold are negative on either learning rate or policy performance. Note that in order to have an obvious observation on performance when evaluate Hypothesis-I, we removed the warm-start policy (from supervised learning) that has been widely used in the literature [4], [41], and reduced the experience replay pool from 100k to 1k, resulting in an extremely challenging task to the dialog learners. And we also report the performance of prioritized experience replay (PER) in these experiments to further evaluate the signi cant improvement of our hindsight method.

We assume that too small stitching threshold generates too few stitched dialogs to speed up the learning ef ciency, though the newly generated dialogs are of very good quality, and too big one may generate many dialogs with different quality, even vary unrealistic ones, producing negative effects to the learning process. As for more understanding the roles of adaptive coordinator played in LHUA, we hypothesize that the behaviors (i.e., the value of k) of adaptive coordinator should change accordingly during training to help dialog agent suf ciently exploit the user modeling.

Two Versions of Hindsight Manager: Figure 6 shows the results of Hypothesis-I evaluation. We can see that uniform experience replay (ER) [27], and PER [30] could not accomplish any task. Under such challenging settings, our complex HER methods, LH Full and LH Light, achieved > 0.5 success rates. All versions of LH (including LH Full, LH Light and the combination of LH Full and LH Light) can ultimately reach comparable success rates. However, their learning rates are different. One observation is that LH Full outperformed LH Light. This is intuitive, because LH Full utilizes both head and tail segments in hindsight manipulation, and can synthesize more trials to accelerate policy learning. Another observation is that the combination of LH Full and LH Light produced the best performance on learning rate. This is because LH Full synthesized the most dialog trials, including those from the head segments of dialogs, and those that connect head and tail segments.

Selection of Stitching Thresholds: The above experiment has shown that LH is necessary for hindsight manager to generate more successful and complete dialogs for speed up the training. However, stitching is allowed in LH only if the KL divergence between two connecting states is below a "stitchability" threshold (details in Algorithm 2). Intuitively, if the threshold is too small, there will be very few dialog segments being stitched (though the newly generated dialogs are of very good quality). If the threshold is too big, many dialog segments are stitched, but the quality of generated, arti cial dialogs will be poor, producing negative effects to the learning process.

Figure 7 depicts the in uences of different thresholds. As expected, when the threshold is too high (e.g., $\epsilon = 1.0$), there is a side effect on policy convergence of the RL agent, as the nal performance is lower than RL agents with smaller thresholds (i.e., $\epsilon = 0.2$); when the threshold is too small (e.g., $\epsilon = 0$), the improvement to the learning rate is not as good as using a threshold within the range of $[0.2, 0.4]$. Results here can serve as a reference to LH Full practitioners.

D. Adaptive Coordination Strategies

Results reported in Figure 5 have shown the necessity of our adaptive coordinator in LHUA. In this experiment, we look into the learning process of the adaptive coordinator, to better know its effects and behaviors (i.e., different values of k). More speci cally, we are interested in how the value of k is selected (see Algorithm 4). We have implemented LHU with six different values of k, and their performances are reported in Figure 8, where the left sub gure is on success rate, and the middle is on Area under Curve (AUC). The AUC metric has been used for the evaluation of learning speed [62], [63].

Fig. 7. Stitchability in LH: KL divergence threshold' around 0.2 performs the best.

We see that, in early learning phase (within 100 episodes), the k value of 10 produced the best performance overall, though the performance is comparable to that with $k = 12$ to some level.

The right subfigure in Figure 8 reports the selection of k values by our adaptive coordinator. Each bar corresponds to an average over the k values of 25 episodes. We see that the value of k was suggested to be around 10 within the first 100 episodes, which is consistent to our observation from the results of Figure 8. The consistency further justified our adaptive coordinator's capability of learning the interaction strategy in switching between real and simulated users.

VII. Conclusion

IN this article, we develop an algorithm called Learning with Hindsight, User modeling, and Adaptation (LHUA) for sample-efficient dialog policy learning. LHUA enables dialog agents to adaptively learn with hindsight from both simulated and real users. Simulation and hindsight provide the dialog agent with more experience and more (positive) reinforcements respectively. Experimental results suggest that LHUA outperforms competitive baselines from the literature in both learning speed, and the quality of resulting policies. We have conducted ablation studies by evaluating the performances of LHUA's no-simulation, no-adaptation, and no-hindsight counterparts. This is the first article that enables a dialog agent to adaptively learn from real, simulated, and hindsight experiences all at the same time.

In the future, we plan to evaluate our algorithm using other dialog simulation platform, e.g., PyDial [64]. Another direction is to combine other efficient exploration strategies, including learning directed exploration policies with different trade-offs between exploration and exploitation [65]. We will also focus on generating more synthetic dialog experience of different quality [66], to further improve the dialog learning efficiency. There are other sensory modalities that can be incorporated into goal-oriented dialog systems. For instance, recent research has demonstrated the promises of visual-dialog navigation [67], [68]. We plan to implement our dialog system on mobile robots, and evaluate the co-optimization of dialog and motion strategies. Finally, there is a very interesting research direction for future work on developing an intrinsically motivated simulated user that intentionally synthesizes dialog samples different from those from real users. This "open-ended learning" mechanism [69] will enable the dialog agent to learn skills from artificial experience. The learned diverse skills can help the agent deal with new dialog situations when they arise, and improve the robustness of the learned dialog policies.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China under grant number U1613216, and a portion of this work has taken place in the Autonomous Intelligent Robotics (AIR) Group at SUNY Binghamton. AIR research is supported in part by grants from the National Science Foundation (NRI-1925044), Ford Motor Company (URP Awards 2019 and 2020), OPPO (Faculty Research Award 2020), and SUNY Research Foundation.

References

[1] M. A. Walker, "An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email," *Journal of Artificial Intelligence Research*, 2000.

[2] S. Zhang and P. Stone, "Corpp: commonsense reasoning and probabilistic planning, as applied to dialog with a mobile robot," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[3] P.-H. Su, M. Gasic, N. Mrkšić et al., "On-line active reward learning for policy optimisation in spoken dialogue systems," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016.

[4] X. Li, Y.-N. Chen, L. Li et al., "End-to-end task-completion neural dialogue systems," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2017.

[5] P. Wesselmann, Y.-C. Wu, and M. Gašić, "Curiosity-driven reinforcement learning for dialogue management," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 7210–7214.

[6] Y. Wu, X. Li, J. Liu et al., "Switch-based active deep dyna-q: Efficient adaptive planning for task-completion dialogue policy learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[7] S. Young, M. Gašić, B. Thomson, and J. D. Williams, "Pomdp-based statistical spoken dialog systems: A review," *Proceedings of the IEEE*, 2013.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[9] E. Levin, R. Pieraccini, and W. Eckert, "Learning dialogue strategies within the markov decision process framework," in *IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, 1997.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.

[11] J. D. Williams and G. Zweig, "End-to-end lstm-based dialog control optimized with supervised and reinforcement learning," *arXiv preprint arXiv:1606.01269*, 2016.

[12] H. Cuayáhuitl, "Simpleds: A simple deep reinforcement learning dialogue system," in *Dialogues with Social Robots*. Springer, 2017.

[13] J. Rajendran, J. Ganhotra, and L. C. Polymenakos, "Learning end-to-end goal-oriented dialog with maximal user task success and minimal human agent use," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 375–386, 2019.

[14] K. Lu, S. Zhang, and X. Chen, "Goal-oriented dialogue policy learning from failures," *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[15] B. Peng, X. Li, J. Gao et al., "Deep Dyna-Q: Integrating planning for task-completion dialogue policy learning," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018. [Online]. Available: https://www.aclweb.org/anthology/P18-1203

[16] S.-Y. Su, X. Li, J. Gao et al., "Discriminative deep dyna-q: Robust planning for dialogue policy learning," *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018. [Online]. Available: http://dx.doi.org/10.18653/v1/d18-1416

Fig. 8. Success rate on the left, and Area under Curve (AUC) on the right, where we implemented six different versions of LHU with different *k* values, ranging from 6 to 16 at an interval of 2.

[17] Y. Cao, K. Lu, X. Chen, and S. Zhang, "Adaptive dialog policy learning with hindsight and user modeling," in *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, Jul. 2020, pp. 329–338.

[18] J. Schatzmann, B. Thomson, K. Weilhammer *et al.*, "Agenda-based user simulation for bootstrapping a pomdp dialogue system," in *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*. Association for Computational Linguistics, 2007.

[19] X. Li, Z. C. Lipton, B. Dhingra *et al.*, "A user simulator for task-completion dialogues," *arXiv preprint arXiv:1612.05688*, 2016.

[20] P.-H. Su, M. Gašić, N. Mrkšić *et al.*, "On-line active reward learning for policy optimisation in spoken dialogue systems," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016. [Online]. Available: https://www.aclweb.org/anthology/P16-1230

[21] Z. C. Lipton, J. Gao, L. Li, X. Li, F. Ahmed, and L. Deng, "Ef- cient exploration for dialogue policy learning with bbq networks replay buffer spiking," Tech. Rep. MSR-TR-2016-62, August 2016, arXiv:1608.05081.

[22] T. Zhao and M. Eskenazi, "Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning," in *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2016. [Online]. Available: https://www.aclweb.org/anthology/W16-3601

[23] J. D. Williams, K. Asadi, and G. Zweig, "Hybrid code networks: practical and ef cient end-to-end dialog control with supervised and reinforcement learning," *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017. [Online]. Available: http://dx.doi.org/10.18653/v1/P17-1062

[24] B. Dhingra, L. Li, X. Li *et al.*, "Towards end-to-end reinforcement learning of dialogue agents for information access," *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017. [Online]. Available: http://dx.doi.org/10.18653/v1/P17-1045

[25] B. Liu and I. Lane, "Iterative policy learning in end-to-end trainable task-oriented neural dialog models," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017.

[26] B. Peng, X. Li, L. Li *et al.*, "Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning," *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2017. [Online]. Available: https://www.aclweb.org/anthology/D17-1237

[27] V. Mnih, K. Kavukcuoglu, D. Silver *et al.*, "Human-level control through deep reinforcement learning," *Nature*, 2015.

[28] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek *et al.*, "Massively parallel methods for deep reinforcement learning," *arXiv preprint arXiv:1507.04296*, 2015.

[29] Z. Wang, V. Bapst, N. Heess, V. Mnih *et al.*, "Sample ef cient actor-critic with experience replay," *arXiv preprint arXiv:1611.01224*, 2016.

[30] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[31] S. Han and Y. Sung, "Multi-batch experience replay for fast convergence of continuous action control," *arXiv preprint arXiv:1710.04423*, 2017.

[32] M. Andrychowicz, F. Wolski, A. Ray *et al.*, "Hindsight experience replay," in *Advances in Neural Information Processing Systems*, 2017, pp. 5048–5058.

[33] E. Ferreira and F. Lefevre, "Reinforcement-learning based dialogue system for human–robot interactions with socially-inspired rewards," *Computer Speech & Language*, 2015.

[34] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*, 1999.

[35] P.-H. Su, D. Vandyke, M. Gasic *et al.*, "Reward shaping with recurrent neural networks for speeding up on-line policy learning in spoken dialogue systems," in *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2015.

[36] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of machine learning research*, 2003.

[37] S. Chandramohan, M. Geist, and O. Pietquin, "Optimizing spoken dialogue management with tted value iteration," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[38] O. Pietquin, M. Geist, S. Chandramohan, and H. Frezza-Buet, "Sample-ef cient batch reinforcement learning for dialogue management optimization," ACM Transactions on Speech and Language Processing (TSLP) 2011.

[39] O. Pietquin, M. Geist, S. Chandramohan *et al.*, "Sample ef cient on-line learning of optimal dialogue policies with kalman temporal differences," in *IJCAI Proceedings-International Joint Conference on Arti cial Intelligence*, 2011.

[40] M. Gašić and S. Young, "Gaussian processes for pomdp-based dialogue manager optimization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2014.

[41] Z. Lipton, X. Li, J. Gao, L. Li *et al.*, "Bbq-networks: Ef cient exploration in deep reinforcement learning for task-oriented dialogue systems," *arXiv preprint arXiv:1711.05715*, 2017.

[42] P.-H. Su, P. Budzianowski, S. Ultes, M. Gasic, and S. Young, "Sample-ef cient actor-critic reinforcement learning with supervised data for dialogue management," in *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, 2017.

[43] P.-H. Su, M. Gasic, N. Mrksic *et al.*, "Continuously learning neural dialogue management," *arXiv preprint arXiv:1606.02689*, 2016.

[44] B. Peng, X. Li, L. Li *et al.*, "Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.

[45] V. Santucci, P.-Y. Oudeyer, A. Barto, and G. Baldassarre, "Editorial: Intrinsically motivated open-ended learning in autonomous robots," *Frontiers in Neurorobotics*, vol. 13, p. 115, 01 2020.

[46] P. Y. Oudeyer, F. Kaplan, and V. V. Hafner, "Intrinsic motivation systems for autonomous mental development," *IEEE Trans Evolutionary Computation*, vol. 11, no. 2, pp. 265–286, 2007.

[47] F. Mannella, V. G. Santucci, E. Somogyi, L. Jacquey, K. J. O'Regan, and G. Baldassarre, "Know your body through intrinsic goals," *Frontiers in neurorobotics*, vol. 12, p. 30, 2018.

[48] V. G. Santucci, B. Gianluca, and M. Marco, "Which is the best intrinsic motivation signal for learning multiple skills?" *Frontiers in Neurorobotics*, vol. 7, no. 22, 2013.

[49] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 16–17.

[50] M. B. Hafez, C. Weber, and S. Wermter, "Curiosity-driven exploration enhances motor skills of continuous actor-critic learner," in *2017 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*. IEEE, 2017, pp. 39–46.