Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Policy reuse for dialog management using action-relation probability

TUNG. T. NGUYEN[1], KOICHIRO YOSHINO[1,2], (MEMBER, IEEE), SAKRIANI SAKTI[1,3], (MEMBER, IEEE), AND SATOSHI NAKAMURA [1,3], (Fellow, IEEE)
[1]Nara Institute of Science and Technology, Ikoma, Nara, Japan 630-0192 Japan
[2]Japan Science and Technology Agency, Japan
[3]RIKEN, Center for Advanced Intelligence Project AIP, Japan

Corresponding author: Tung T. Nguyen (e-mail: nguyen.tung.np5@ is.naist.jp).

Parts of this work are supported by JSPS KAKENHI Grant Number 20H05567 and JP17H06101.

ABSTRACT We study the problem of policy adaptation for reinforcement-learning-based dialog management. Policy adaptation is a commonly used technique to alleviate the problem of data sparsity when training a goal-oriented dialog system for a new task (the target task) by using knowledge when learning policies in an existing task. The methods used by current works in dialog policy adaptation need much time and effort for adapting because they use reinforcement learning algorithms to train a new policy for the target task from scratch. In this paper, we show that a dialog policy can be learned without training by reinforcement learning in the target task. In contrast to existing works, our proposed method learns the relation in the form of probability distribution between the action sets of the source and the target tasks. Thus, we can immediately derive a policy for the target task, which significantly reduces the adaptation time. Our experiments show that the proposed method learns a new policy for the target task much more quickly. In addition, the learned policy achieves higher performance than policies created by fine-tuning when the amount of available data on the target task is limited.

INDEX TERMS dialog management, reinforcement learning, transfer learning, policy adaptation, mixture density network

I. INTRODUCTION

REINFORCEMENT learning (RL) is a widely used framework for modeling the decision-making process in such tasks as the dialog management of conversational systems [1]–[3]. In general, training a policy with reinforcement learning helps the system to learn a more robust behavior and its performance will not be upper-bounded by the human performance in the training dialog samples, which is sometimes not optimal. Nonetheless, the training process of reinforcement learning is usually tedious and needs a large number of samples to train an optimal policy. Policy adaptation, or policy transfer, is a very useful technique that can tackle this problem in reinforcement learning.

Policy adaptation refers to the process of reusing knowledge, i.e., a policy that is learned in one or multiple source tasks to a new target task. Various literatures which has studied about policy adaptation in RL proposed different techniques that show such promising results as the acceleration of the convergence rate and the reduction of data volume requirements [4], [5].

Within the scope of reinforcement learning-based dialog management, the application of policy adaptation remains very limited. Current works in dialog policy adaptation [6]–[8] follow the weight initialization strategy, which contains two steps: pre-training and fine-tuning. Pre-training refers to the process that trains a policy in the source task, where the policy is usually represented by a neural network. Some of the source policy's weight parameters are used to initialize the neural network's weights to train a policy in the target task. Next, we fine-tune the network weights by training with a reinforcement learning algorithm.

However, when we do not have enough data on the target task, this strategy does not work well because it barely uses the knowledge from the source task's policy, forcing the target task's policy to be basically trained from scratch. Consider a situation with a dialog policy that handles restaurant reservations and we want to adapt to manage the task of booking hotel rooms. Obviously, we expect that this adaptation can be performed with minor adjustment to the restaurant booking task's policy. Motivated by this observation, we proposed a novel method to adapt dialog policy called the *Dialog Policy Reuse Algorithm (DPRA)*. Unlike previous works, our proposed method learns the relation between the action sets of the source and the target tasks using a mixture density network [9], and then we can quickly infer the policy for the target task without RL training. In other words, DPRA allows us to "reuse" the source task policy for action decision-making in the target task. The following are the primary contributions of our work:

- We propose a novel method for the policy adaptation of reinforcement-learning-based dialog management. In particular, the source task's policy can be used for the action selection procedure in the target task through a special mapping called *action-relation probability*. We propose a mixture density network to model this distribution.
- Since our proposed method can learn a policy for the target task without RL training, it reduces the effort to construct dialog policies in the target task.
- Experimental results show that, when only a small amount of data is available for training, our proposed method achieves higher performance than the baseline adaptation method, which is based on fine-tuning, and also requires much less time for training.

The remainder of this paper is organized as follows. Section II provides background knowledge in reinforcement learning and dialog management. In Section III, we explain our proposed method, the DPRA algorithm. Section IV describes our experiment setting, its results, and analyses. Section V explains current works in dialog policy adaptation and their drawbacks. Finally, we summarize our work and discuss future directions in Section VI.

II. REINFORCEMENT LEARNING AND DIALOG MANAGEMENT

A. REINFORCEMENT LEARNING

Reinforcement learning is a popular framework for learning autonomous behavior. We consider the standard reinforcement learning setting where an agent interacts with an environment that follows a Markov decision process (MDP) over a number of discrete time steps [10]. At each time step t, the state, action, and reward are respectively denoted by $s_t \in S$, $a_t \in A$, and $r_t \in R$. The dynamics of the task (the environment) are characterized by two random variables: state transition probabilities $P^a_{ss'} = P(s_{t+1} = s'|s_t = s, a_t = a)$, and the expected reward, given by $R^a_s = E[r_{t+1}|s_t = s, a_t = a] = \sum_{r_{t+1}} r_{t+1} P(r_{t+1}|s_t = s, a_t = a)$. The agent's procedure for selecting action a given state s is the agent's policy, denoted by $\pi(s,a) = P(a|s)$. We define the *return*, which is the total rewards that received by the agent, as $R_t = \sum_{k=1}^{T-k} r_{t+k}$ where T is the final time step. The agent's objective is to maximize expected return $E[R_t|s_t, \pi]$ at each time step t when following policy π. If the agent-environment interactions do not stop, T goes to ∞. We describe our task as *continuing*. If the interactions eventually end when we reach certain terminal states, then our task is called *episodic*. In this setting, the interactions from the beginning until the agent reaches a terminal state is called an *episode*. Appropriate setting are chosen depends on the problem we want to solve using reinforcement learning.

Given policy π, the state-action value is defined as: $Q^\pi(s,a) = E[\sum_{k=1}^{T-k} r_{t+k}|s_t = s, a_t = a, \pi]$, which is the expectation of the return if action a is chosen at time step t. Similarly, we define the state value of policy π: $V^\pi(s) = E[\sum_{k=1}^{T-k} r_{t+k}|s_t = s, \pi]$. Note that the above definitions of state-action, and state value fall under the *undiscounted* setting.

Reinforcement learning algorithms can be divided into two classes: value-based and policy-based. In value-based reinforcement learning methods, we estimate action-state $Q^\pi(s,a)$ or the state value $V^\pi(s)$ by using a function approximator, such as neural networks or simple value tables. Classic Q-learning [11] or deep Q-network [12] are examples of this class of algorithms. In contrast to value-based methods, policy-based RL algorithms parameterize policy π by parameters θ, which we update by performing (typically approximate) gradient ascent to maximize $E[R_t|\pi]$. There are various policy-based RL studies, especially with policy gradient methods, such as Actor-Critic [13], [14] or REINFORCE [15], which we are using in our evaluation.

B. DIALOG MANAGEMENT USING REINFORCEMENT LEARNING

A dialog can be divided into multiple turns, where each turn contains a user utterance and a system response. We can formulate the problem of dialog management as an MDP and apply any RL algorithm to solve it. We can define a set of actions for the system to interact with the user and define a reward function based on the system's goal. Since all dialogs only have a finite number of turn (or time steps) we need to use the episodic and undiscounted setting as a formulation of dialog management problem.

At each time step t, the required information for the action selection procedure is defined as an *observation*, denoted by o_t. The type of information included in an observation depends entirely on the task that is being solved. For example, in dialog management, the observation may consist of recognition results of slot information [6], [7], [16], user's dialog actions [17], [18], or such high-level multimodal information as user's deception [19]. Recall that the policy is a conditional probability for selecting action a given state s, $P(a|s)$; thus, the state must include critical information for making the decision. A natural approach is to represent the state by the vector of observation or the concatenation of observations from multiple time steps. We call such state representation as *explicit state representation.* In modular-based dialog systems [1], [2], [17]–[19], the state is represented by this explicit representation.

In dialog management procedures, the system needs to consider the dialog history, which contains the user's utterances and the system's actions from the beginning. With such long-term tracking requirements, explicit representation becomes unsuitable if the observations are high-dimensional or the conversations are lengthy. A common solution is using a Recurrent Neural Network (RNN) to learn state embedding, which resembles a *latent state representation* that stores the dialog history and current observation [16], [20]. Such state representation is used in end-to-end dialog modeling [20]. Fig. 1 shows an example of this approach. The RNN plays the role of dialog state tracker (DST) and its output, the latent state representation, is used by the dialog policy module for management. With an end-to-end approach, we are free from designing a complicated explicit representation of the dialog states.

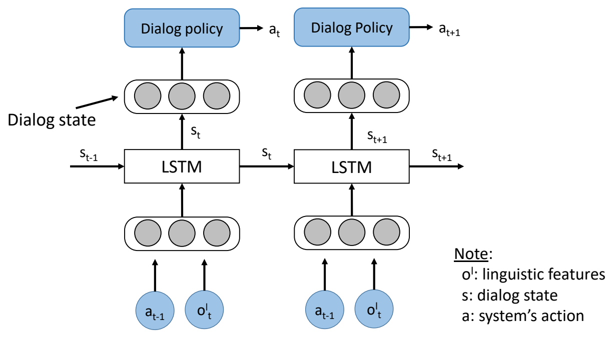


FIGURE 1. Dialog modeling with end-to-end approach.

RL training requires a huge amount of agent-environment interactions to learn a good policy. In dialog management, it is impossible to collect enough dialog samples to fulfill this requirement. Scheffler and Young proposed a *user simulator* as replacement for actual human users to train the policy [1]. This approach has become the standard for training dialog policies with RL. The user simulator is built from dialog samples by maximum likelihood or supervised learning methods, to imitate the human user's behavior. The user simulator can also be viewed as an approximation of the dialog management task's true state transition $P^a_{ss'}$, which is provided by the actual human user.

C. POLICY ADAPTATION

Humans are capable of learning a task better and faster by transferring the knowledge retained from solving similar tasks. This observation motivates the idea of transferring knowledge across different but related tasks to improve the performance of machine learning (ML) algorithms. The techniques that allow such knowledge transferring is called *transfer learning*.

Application of transfer learning to RL algorithms started to gain attention of the machine learning community from the middle of the 2000s. In reinforcement learning, the number of samples needed to learn an optimal solution is usually prohibitive, especially for dialog management, where data sparsity is a huge challenge [21]. Transfer learning can build prior information from the knowledge collected to solve a set of source tasks and be used for learning a policy in the target task.

Many types of knowledge can be used in transfer learning in RL, such as samples, representation, or parameters [4]. *Policy adaptation*, or *policy transfer*, refers to transfer learning methods that use knowledge of the policy from the source task for the transferring process. Policy adaptation methods are subclass of transfer learning solutions in reinforcement learning.

We usually face the problem of data sparsity when training policies for dialog tasks, because they do not require additional knowledge between the source and the target task. In this situation, the policy adaptation approach is a promising solution. Multiple studies investigate its application in reinforcement-learning-based dialog management [6]–[8]. All of these methods improve learning speed, reduce data requirements, or performance.

D. PROBLEM STATEMENT OF POLICY ADAPTATION

Current policy adaptation studies in reinforcement-learning-based dialog management follow the weight initialization approach [6]–[8]. As explained in Section I, this strategy requires us to train the target policy from scratch with an RL algorithm, which is a tedious process that requires great effort, especially for complex tasks. In addition, the construction of dialog policies usually involves user simulations. When we only have a small amount of data in the target dataset, the user simulator does not represent the behavior from the actual human users very well. Therefore, a policy that we train can have high performance against the simulator even though fails to work well versus actual human users.

Problem statement. Given a source task with the state space S_A and action set A, assume that we have trained policy $\pi(s, a)$ for the source task. The target task's state space and action set are denoted as S_B and B, our goal is to derive policy $\pi(s, b)$ for the target task from $\pi(s, a)$ without RL training.

III. POLICY REUSE BASED ON ACTION-RELATION PROBABILITY

In this section, we show that a policy can be adapted for the target task without RL training from the scratch. Instead of training a policy by interactions with a user simulator, we establish a connection between the policies of the source and target tasks through a special mapping distribution called *action-relation probability*. Our proposed adaptation method, DPRA, learns this distribution from dialog samples in both tasks, and can immediately derive a policy for the target task. Thus, DPRA can remarkbly reduce learning time. Since our proposed method does not use the user simulator, we can avoid the problem of low performance due to errors in constructing the user simulator.

A. POLICY ADAPTATION BY ACTION-RELATION PROBABILITY

We consider policy adaptation from a source task to a target task. First, we make the following assumptions, which allow the derivation of relation between the policies of the source and the target tasks :

Assumption 1: The source and the target tasks share identical state space S. This assumption is actually not restrictive. Union space $S = S_A \cup S_B$ is the state space that satisfies the assumption.

Assumption 2: The source and the target tasks have identical state representation.

We define *observation* as the information that is necessary for the agent's action selection in each time step. An example is the features extracted from the user's utterance at each turn. Obviously, in a policy adaptation setting, the source and the target tasks are different and they require distinctive sets of features. Thus, the state representation of the source task is also not identical to the target task's. However, we can define a unified set of features for those tasks and have the same observation and state representation across the source and the target tasks.

We establish a connection between the source and the target task's policy as follows. Denote the source task's policy as $\pi(s, a)$ and the target task's policy as $\pi(s, b)$, where $a \in A, b \in B$ are action sets in the source and target tasks.

$$\pi(s, b) = \sum_{a \in A} P(b|a, s)\pi(s, a). \tag{1}$$

Equation (1) argues that with any conditional distribution of $P(b|a, s)$; from source task's policy $\pi(s, a)$, we can directly infer a policy for target task $\pi(s, b)$. We call this $P(b|a, s)$ *action-relation probability*. Our proposed policy adaptation method models this distribution instead of performing RL training.

B. ACTION-RELATION MODELING WITH MIXTURE DENSITY NETWORK

This section explains the modeling of action-relation probability distribution based on a mixture model. First, denote the state and action at time step t as $s = s_t$ and $a = a_t$, where the state at the next time step is $s' = s_{t+1}$. The state transition in the target task is given:

$$\begin{aligned} P(s'|a, s) &= \sum_{b \in B} P(s', b|a, s) \quad \textit{(law of total probability)} \\ &= \sum_{b \in B} P(s'|b, a, s)P(b|a, s). \end{aligned} \tag{2}$$

The state transition of the source task has the form of a mixture model with the action-relation probability as the component weights. Mixture density network (MDN) [9], is a suitable approach for modeling state transition $P(s'|a, s)$. In principle, a mixture density network is a type of Gaussian mixture model (GMM) that utilizes an artificial neural network. Given multivariate random variables x, y, MDN models conditional probability density $p(y|x)$:

$$p(y|x) = \sum_{m=1}^{M} w_m(x) \cdot \mathcal{N}(y; \mu_m(x), \sigma_m^2(x)). \tag{3}$$

M is the number of components, and $w_m(x)$, $\mu_m(x)$, and $\sigma_m^2(x)$ are the component weight, mean, and standard deviation for component m. We assume that these mixture variables are functions of input x that are approximated by neural networks $f_m^w, f_m^\mu, f_m^\sigma$, parameterized by $\theta_m^w, \theta_m^\mu, \theta_m^\sigma$. With some slight abuse of notation, we have:

$$w_m(x) \approx \frac{\exp(f_m^w(x; \theta_m^w))}{\sum_{l=1}^{M} \exp(f_l^w(x; \theta_l^w))} \tag{4a}$$

$$\mu_m(x) \approx f_m^\mu(x; \theta_m^\mu) \tag{4b}$$

$$\sigma(x) \approx \exp(f_m^\sigma(x; \theta_m^\sigma)) \tag{4c}$$

With MDN, we assume that all the components in multivariate random variable y are mutually independent, and thus the covariance matrix is diagonal and can be represented by a vector with the same dimension as $f_m^\mu(x)$. Denote the dataset as $D = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}, i = 1..N$, where $\mathbf{x}, \mathbf{y}$ are the observed data for random variables x and y. The parameters are optimized using gradient descent with the following negative log-likelihood:

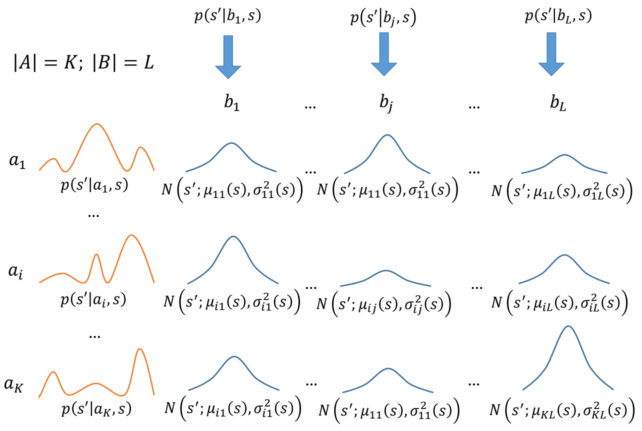
$$L = - log(\prod_{i=1}^{N} p(\sum_{m=1}^{M} w_m(\mathbf{x^{(i)}}) \cdot \mathcal{N}(\mathbf{y}^{(i)}; \mu_m(\mathbf{x^{(i)}}), \sigma_m^2(\mathbf{x^{(i)}})))) \tag{5}$$

By replacing the probabilities with probability density functions, the mixture model in (2) is now given:

$$\begin{aligned} p(s'|a_i, s) &= p_i(s'|s) \qquad (a_i \in A) \\ &= \sum_{j=1}^{|B|} P_{ij}(s) \cdot p_{ij}(s'|s) \\ &= \sum_{j=1}^{|B|} w_{ij}(s) \cdot \mathcal{N}(s'; \mu_{ij}(s), \sigma_{ij}^2(s)). \end{aligned} \tag{6}$$

An illustration of these mixture models is given in Fig. 2. In principle, the density of the state transition caused by a_i is a mixture model with each component $p_{ij}(s'|s) = p(s'|a_i, b_j, s)$ follows Gaussian distribution $\mathcal{N}(s'; \mu_{ij}(s), \sigma_{ij}^2(s))$.

For each action a_i in the source task, we can train its corresponding MDN with just the samples (s', a_i, s). Particularly, s is observed variable x and s' is the latent variable y in (5), a_i is the indicator to which Gaussian component corresponds to sample s, s'. The action-relation probability $P(b = b_j|a = a_i, s)$ is approximated by $f^w(x; \theta_{ij}^w))$, as shown in (4a). However, in that case, we cannot guarantee that $\mathcal{N}(s'; \mu_{ij}(s), \sigma_{ij}^2(s))$ truly models state transition $p(s'|a_i, b_j, s)$ since the source task samples do not contain



$p(s'|b_1,s)$
$p(s'|b_j,s)$
$p(s'|b_L,s)$
$|A|=K;\ |B|=L$
b_1 ... b_j ... b_L
a_1
$p(s'|a_1,s)$
$N(s';\mu_{11}(s),\sigma_{11}^2(s))$ ... $N(s';\mu_{11}(s),\sigma_{11}^2(s))$ ... $N(s';\mu_{1L}(s),\sigma_{1L}^2(s))$
...
a_i
$p(s'|a_i,s)$
$N(s';\mu_{i1}(s),\sigma_{i1}^2(s))$ ... $N(s';\mu_{ij}(s),\sigma_{ij}^2(s))$ ... $N(s';\mu_{iL}(s),\sigma_{iL}^2(s))$
...
a_K
$p(s'|a_K,s)$
$N(s';\mu_{i1}(s),\sigma_{i1}^2(s))$ ... $N(s';\mu_{11}(s),\sigma_{11}^2(s))$ ... $N(s';\mu_{KL}(s),\sigma_{KL}^2(s))$

FIGURE 2. State transition modeling by mixture density network. Components in each column correspond to an action in the target task.

any information of b_j. A natural solution is to additionally train the components using the samples (s', b_j, s) by *component matching* process, which actually "matches" the distribution of component $\mathcal{N}(s'; \mu_{ij}(s), \sigma_{ij}^2(s))$ to the transition of $p(s'|b_j, s)$.

In this work, we propose two methods of component matching. In the first, we assume that $p(s'|a_i, b_j, s) = p(s'|b_j, s) \forall a_i \in A, b_j \in B$. With this assumption, we can perform component matching by simply optimizing the networks' parameters using the negative log-likelihood of the target task's samples:

$$L = -log(\prod_{i=1}^{N} \mathcal{N}(\mathbf{s'}^{(i)}; \mu(\mathbf{s^{(i)}}), \sigma^2(\mathbf{s^{(i)}}))). \tag{7}$$

Since the training of this component matching method resembles the training process of a regression model, we define it *component matching by regression*. Algorithm 1 shows the pseudo code for the training process of the mixture model in 6 using component matching by regression.

Algorithm 1 Action-relation probability modeling by MDN with component matching by regression

```
Randomly initialize the network weights $\theta^w, \theta^\mu, \theta^\sigma$
Initialize MDN gradient $d\theta_{MDN} \leftarrow 0$
Initialize the gradient for component matching $d\theta_{CM} \leftarrow 0$
Initialize iteration counter $t \leftarrow 0$
repeat
  for all $a_i$ such that $a_i \in A$ do
    for all $b_j$ such that $b_j \in B$ do
      Calculate gradient $d\theta_{CM}$ by (7)
      Update parameters $\theta^\mu, \theta^\sigma$ with $d\theta_{CM}$
    end for
    Calculate gradient $d\theta_{MDN}$ by (5)
    Update parameters $\theta^w, \theta^\mu, \theta^\sigma$ with $d\theta_{MDN}$
  end for
  $t \leftarrow t + 1$
until $t > t_{max}$
```

The second component matching method stems from the following derivation:

$$\begin{aligned} P(s'|b,s) &= \sum_{a \in A} P(s', a|b, s) \quad (\textit{law of total probability}) \\ &= \sum_{a \in A} P(s'|b, a, s) P(a|b, s). \end{aligned} \tag{8}$$

Equation (8) argues that transition distribution $P(s'|b, s)$ also has the form of a mixture model that can be represented by MDN. The mixture in (8) has the same component $P(s'|a, b, s)$ as in (2), but different component weights $P(a|b, s)$. We define the network's parameters that approximates the component weights as $\hat{\theta}^w$.

Algorithm 2 Action-relation probability modeling by MDN with component matching by MDN

```
Randomly initialize the network weights $\theta^w, \hat{\theta}^w, \theta^\mu, \theta^\sigma$
Initialize MDN gradient $d\theta_{MDN} \leftarrow 0$
Initialize the gradient for component matching $d\theta_{CM} \leftarrow 0$
Initialize the iteration counter $t \leftarrow 0$
repeat
  for all $a_i$ such that $a_i \in A$ do
    for all $b_j$ such that $b_j \in B$ do
      Calculate gradient $d\theta_{CM}$ by (9)
      Update parameters $\hat{\theta}^w, \theta^\mu, \theta^\sigma$ with $d\theta_{CM}$
    end for
    Calculate gradient $d\theta_{MDN}$ by (5)
    Update parameters $\theta^w, \theta^\mu, \theta^\sigma$ with $d\theta_{MDN}$
  end for
  $t \leftarrow t + 1$
until $t > t_{max}$
```

In principle, the components in column of b_j in Fig. 2 form a mixture model for the density of state transition $p(s'|b_j, s)$. Similarly, we can train this mixture with a loss function that resembles (5) using the target task samples (s', b_j, s):

$$\begin{aligned} &L = \\ &- log(\prod_{i=1}^{N} p(\sum_{m=1}^{M} \hat{w}_m(\mathbf{x^{(i)}}) \cdot \mathcal{N}(\mathbf{y}^{(i)}; \mu_m(\mathbf{x^{(i)}}), \sigma_m^2(\mathbf{x^{(i)}})))). \end{aligned} \tag{9}$$

We call this method *component matching by MDN*. The pseudo code for action-relation probability modeling using MDN component matching is shown in Algorithm 2.

Finally, the procedure of our proposed method, the dialog policy reuse algorithm, is shown in Algorithm 3:

Algorithm 3 Dialog policy reuse algorithm: DPRA

```
Step 1: Train policy $\pi(s, a)$ for source task
Step 2: Model action-relation probability $P(b|s, a)$ using
either Algorithms 1 or 2
Step 3: Create a policy for target task $\pi(s, b)$ , by using 1
with the action-relation probability learned in Step 2.
```

We summarize this section by showing that resultant policy $\pi(s,b)$ of DPRA is proper:

$$\sum_{b \in B} \pi(s,b) = 1 \tag{10}$$

Intuitively, DPRA works as follows. Given policy $\pi(s,a)$ in the source task, assume that an agent with policy π selects action a given current state s. DPRA finds action b in the target task that makes similar transition $s \rightarrow s'$ to action a: in other words, $P(s'|s,a) \simeq P(s'|s,b)$. Instead of making a deterministic mapping, we learn a distribution that connects a to all the available actions in the target task, which is $P(b|a,s)$. This is why we use the term action-relation probability to define this special distribution. With the condition that the source and target tasks are also similar in the reward dynamic, if learned policy $\pi(s,a)$ in the source task is optimal, then policy $\pi(s,b)$ learned by DPRA is nearly optimal as well.

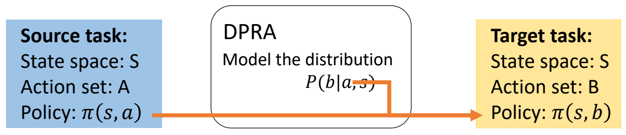


FIGURE 3. Working procedure of the proposed method.

An illustration of DPRA is shown in Fig. 3. DPRA requires identical state space and state representation of the source and target tasks (Assumption 1 and 2). Recall that all policy adaptation methods only work in the cases where the source and target tasks are similar. In this situation, even if the two tasks' state spaces and state representations are not completely identical, we expect that they still share considerable similarities, and thus, the proposed method can still learn a good policy for the target task without the ideal conditions in Assumption 1 and 2.

IV. EVALUATION

A. SETTING

We experimentally assessed the following hypotheses:

1) Our proposed adaptation algorithm, DPRA, requires much less training time than conventional fine-tuning methods.
2) DPRA learns policies that achieves equivalent or higher performance than those learned with current methods when limited data available in the target task.

For comparison with our proposed methods, we used policy adaptation by action embedding [8]. This method uses the same network as our end-to-end dialog modeling and changes its last layer of the network to connect to a new action space in the target task. This simple model does not require any prior information such as relations between actions. Since our proposed algorithm also does not require such prior information either, we selected this method as the baseline.

In our evaluation, we performed policy adaptation for a multimodal goal-oriented dialog system with an end-to-end approach. We chose a multimodal dialog setting because the available corpora for such conversations are mostly small-scale [21] and thus suitable to assess our second hypothesis. In particular, we augment the original end-to-end dialog model(Fig. 1) with a multimodal fusion component that uses the *Hierarchical Tensor Fusion Network* [22]. This component's role is to efficiently combine features from multiple modalities: linguistic, visual, and acoustic. Since it is fully connected to the dialog state tracker (the LSTM layer in Fig. 1), our dialog model still adheres to the end-to-end paradigm.

We formulated the problem of dialog management with an episodic and undiscounted reward setting and trained the dialog policies using REINFORCE [15], which updates policy parameters θ with the following gradient:

$$\partial(\theta) = E\left[\sum_{t=0}^{T} R_t \frac{\partial log\pi^{\theta}(s_t,a_t)}{\partial\theta}\bigg|\pi\right] \tag{11}$$

As in (11), the "vanilla" version of REINFORCE has high variance and slow convergence. To combat this problem, a *baseline* technique was introduced [23], [24]. [25] described how the most natural and effective baseline is the state-value function. Thus, we use $V^{\pi}(s,a)$ as the baseline and the gradient is given by:

$$\partial(\theta) = E\left[\sum_{t=0}^{T}(R_t - V^{\pi}(s_t,a_t)) \frac{\partial log\pi^{\theta}(s_t,a_t)}{\partial\theta}\bigg|\pi\right] \tag{12}$$

The baseline's role in (12) is to reduce the variance of the gradient estimation and smoothen the training. In principle, since any RL algorithm can be used for training the policies, we chose REINFORCE due to its simplicity and good performance.

The output layers of the multimodal fusion component and the DST both have 128 units. Therefore, the dialog state is represented by vector $\mathbf{s} \in \mathcal{R}^{128}$. The neural network that represents the dialog policy has one single hidden layer with 128 units and is fully connected to the input and output layers. We used the Adam optimizer for optimization of the networks' parameters and initialized the learning rate at $1e-3$. We trained the policies for the source and target tasks with 20,000 and 10,000 episodes, of which can be seen as a simulated dialog with the user simulator. The learning rate decreased by 10% every 1000 episodes.

The neural networks that approximate the mixture variables in Algorithms 1 and 2 have one hidden layer with 256 units. We also used the Adam optimizer for parameter optimization. The learning rate is fixed at $1e-4$, and the number of training epochs is 10. Note that in Algorithms 1 and 2, the network parameters are updated sequentially with two different gradients. Thus, the training process has large oscillation and converges slowly. To avoid this problem, we adopted the $p:q$ training scheme. Every epoch, we perform component matching for $p=2$ times and trained the mixture model for $q=2$ times.

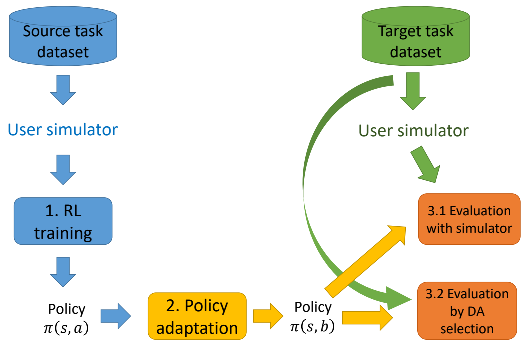


FIGURE 4. Experiment procedure.

B. PERFORMANCE METRICS

We measured the training time of all the methods in seconds. Only the time spent in adaptation is measured; the learning times of the source task's policy and for creating the simulator is not included. We run the adaptation algorithms on identical hardware [1] to ensure a fair comparison among the evaluated methods.

To assess out second hypothesis, we used two performance metrics: average reward per episode and the system's action selection accuracy. With the first metric, we used a user simulator created from the target dataset. We ran the learned policies against this simulator for 1000 episodes and measured the average total reward per episode. For the second metric, several human experts read the dialog transcript and selected the most appropriate system action for each turn. These actions are used as groundtruth for measuring the action selection accuracy of the policies. This metric shows the appropriateness of the learned policy behaviors on each turn.

C. DIALOG TASK

In this evaluation setting, both the source and the target tasks are negotiation task in healthcare domain [19]. The system tries to convince the users that their current living style is unhealthy and suggests that they adopt its proposed living habits.

We used the previously proposed healthcare consultation dataset [19], which contains conversations on six topics: sleeping, eating, working, exercising, social media usage, and leisure activities. The conversations of the first four topics (51 dialogs) are used for training the policies of the source task. The remaining 24 dialogs in two topics are used for training the target task's policy. In each turn, we split the recorded video of the users into 30 segments and randomly sampled one frame from each segment to be used for extracting visual features. We extracted 14 face action unit (AU) regressions and 6 AU classification values for each frame with the OpenFace toolkit [26]. The visual observation at each turn o^v is a vector that contains these extracted values. We extracted acoustic features from the audio using the OpenSMILE toolkit [27] with the Interspeech 2009 (IS09) emotion challenge standard feature-set as our feature template [28]. We used these extracted features to create acoustic observation o^a.

[1]CPU: Intel Xeon CPU E5-2630 v4, GPU: GTX Titan X.

The set of system's actions in the source task is identical as [19], which includes {*Offer, Framing, End*}. We changed the system's action set of the target task into {*Offer_New, Offer_Change, Framing_Argue, Framing_Answer, End_Dialog*}. The source policy never sees these actions during training.

For the RL training, we created a user simulator that generates labels of user action u and deception information d with the following intention and deception models:

$$\begin{aligned}\text{intention model} &= P(u_{t+1}|u_t, d_t, d_{t+1}, a_t)\\ \text{deception model} &= P(d_{t+1}|u_t, d_t, a_t)\end{aligned} \tag{13}$$

Recall that in each dialog turn, the system takes input features (observations) from three modalities: linguistic, visual, and acoustic. We employed the user action u as linguistic observation o^l. The visual and acoustic observations, o^v and o^a, are sampled uniformly from the dialog corpus using u and d.

D. EXPERIMENT RESULTS

We conduct the following experiments to assess the hypotheses raised in the beginning of this section.

1) Policy adaptation time

TABLE 1. Comparison of training time required for different policy adaptation methods.

Model	Training time
Policy adaptation by action embedding [8]	$\sim$ 350s
DPRA - regression component matching	$\sim$ 40s
DPRA - MDN component matching	$\sim$ 45s

Table 1 shows the training times required for all the policy adaptation methods. The numbers reported for DPRA are from a case in which all 24 dialogs of the target task are available for training. Cases with less data will obviously take less time for training with DPRA. With policy adaptation by action embedding [8], we chose the number of episodes (interactions with simulated users) for training the target policy based on the average rewards received per episode. As seen from Table 1, since DPRA requires significantly less time for training, our first hypothesis stands.

2) Dialog policy performance comparison

We recreated scenarios in test sets where the amount of data available for training in the target task is limited to assess the second hypothesis. In particular, we sampled k dialogs from the target task dataset, $k \in \{1, 2, 4, 8, 16\}$. We used these k dialog samples to create a user simulator for training

TABLE 2. Average reward per episode of learned policies with different amounts of available data. Numbers in brackets indicate 95% confidence intervals.

Model \ # available dialogs	1 dialog	2 dialogs	4 dialogs	8 dialogs	16 dialogs
NoAdapt - 2k	-52.93(±16.59)	-23.42(±13.93)	-26.61(±17.98)	-19.15(±18.29)	-11.77(±19.80)
NoAdapt - 10k	-36.95(±15.21)	-17.72(±16.27)	-9.02(±13.92)	-7.75(±14.40)	10.55(±19.95)
ActEmb - 2k	-33.56(±17.03)	-11.48(±10.33)	0.01(±11.45)	-8.56(±13.00)	12.86(±11.05)
ActEmb - 10k	-29.98(±13.12)	-18.07(±11.60)	-13.42(±12.22)	0.64(±11.99)	2.04(±13.72)
DPRA - MDN	-25.43 (±16.16)	-10.33 (±8.94)	-2.18 (±10.71)	**14.93 (±4.11)**	17.21 (±4.2)
DPRA - regression	**-12.96 (±11.05)**	**-0.01(±10.24)**	**6.11(±6.03)**	14.65(±6.27)	**19.41(±4.49)**

TABLE 3. Dialog act selection accuracy of learned policies with different amounts of available data. Numbers in brackets indicate 95% confidence intervals.

Model \ # available dialogs	1 dialog	2 dialogs	4 dialogs	8 dialogs	16 dialogs
NoAdapt - 2k	33.80% (±7.54%)	35.86% (±7.06%)	35.62% (±7.06%)	34.80% (±7.17%)	43.14% (±7.50%)
NoAdapt - 10k	37.39% (±6.91%)	44.50% (±8.01%)	47.86% (±7.61%)	44.41% (±6.31%)	52.16% (±7.76%)
ActEmb - 2k	33.83% (±4.88%)	33.25% (±4.73%)	41.41% (±5.92%)	39.41% (±5.75%)	43.64% (±5.41%)
ActEmb - 10k	28.04% (±5.24%)	24.79% (±5.04%)	26.11% (±6.78%)	25.62% (±6.06%)	29.30% (±5.04%)
DPRA - MDN	40.96% (±7.79%)	39.86% (±6.81%)	56.16% (±7.82%)	62.05% (±4.94%)	62.05% (±4.64%)
DPRA - regression	**45.41% (±5.99%)**	**52.79% (±5.18%)**	**60.48% (±6.02%)**	**62.98% (±5.08%)**	**69.30% (±3.22%)**

the target policy in the *ActEmb* adaptation method and for modeling the action-relation probability in DPRA. For each value of k, we sampled k dialogs five times, thus making five different datasets. With each dataset, we perform policy adaptation ten times for each method and conduct 50 runs of the policy adaptation experiment with each value of k.

Performance in terms of received rewards. Table 2 shows the performance of the dialog policies in terms of the average reward per episode. The details of the reward function for both the source and target tasks are provided in Appendix A. DPRA-MDN and DPRA-regression respectively refer to the proposed policy adaptation method with component matching by MDN and regression. ActEmb-2k and ActEmb-10k refer to policy adaptation by action embedding method, where the number of episodes for training in the target task is 2,000 and 10,000. Finally, NoAdapt refers to a policy that is trained on the source task without adaptation, where the notation for number of training episodes is identical as ActEmb. As seen in the table, the performance generally increases when more data are available. In Table 2, bold numbers indicate the policy with the highest average reward per episode in each scenario where the number of available dialogs k. Policies adapted by DPRA shows significantly higher performance than those from ActEmb and NoAdapt ($p < 0.05$) with all k, and the difference is bigger when k is small. ActEmb-2k and ActEmb-10k perform similarly; on the other hand, the performance of NoAdapt-10k is remarkably higher than NoAdapt-2k.

Performance in terms of DA selection. The performance in terms of dialog act selection accuracy for all the policies is shown in Table 3. Similarly, the dialog policies learned by DPRA outperformed those of ActEmb and NoAdapt with a large margin ($p < 0.05$) for all values of k. Surprisingly, when more data available the performance gap between DPRA and the other methods increases. In fact, there is only a subtle increase in the action selection accuracy of the policies learned by ActEmb and NoAdapt when k increases from 1 to 16. Recall that the results in Table 2 were reported under a setting where the policies were run against a simulator that is created from all 24 dialogs in the target dataset. Therefore, if we train a policy using a simulator created from 16 dialogs, the performance in terms of average reward per episode will be much higher than using a simulator from just one dialog. However, because ActEmbed and NoAdapt does not use full knowledge from the source task policy, 16 dialogs are insufficient to train a policy with high action selection accuracy. Thus, the gain is modest when increasing the amount of available data in Table 3. In contrast, DPRA can retain knowledge from the source task policy and effectively adapt it to the target task, thus achieving high performance in terms of action selection accuracy, especially when more data are available.

V. RELATED WORKS

Many studies have addressed policy adaptation for reinforcement-learning-based goal-oriented dialog management.

Chen et al. [6] proposed a policy adaptation method using a multi-agent dialog policy. They used an explicit representation of the dialog state, which contains of multiple slot information. For each slot, they built an “agent” that learns how to choose actions corresponding to this slot. The dialog policy is an ensemble of these agents. In the target task, for each new slot information, the network weights of its corresponding agent are initialized using the weights of the agents that have been trained in the source task. Although this adaptation method does not require the state representation to be similar, as in DPRA, it has a restriction: the state representation of both tasks must be explicit, such as slots or values. In addition, this method requires identical action sets for each agent to perform weight initialization. This makes Chen et al.’s method [6] less flexible than DPRA in terms of action space restriction.

Ilievski et al. [7] introduced a new policy adaptation method by using weight bootstrapping and also used slot information for the state representation. Their method shares the slots and actions across the source and target tasks and constructs a neural network where the input layer 's number of neurons equals to the number of unique slots in both tasks. Similarly, the output layer has the number of neurons that is equal to the number of total grouped actions. First, they trained a policy on a source task by reinforcement learning. The network weights are then fine-tuned by training on the target task. This adaptation method has the same restriction of state representation as in a previous work [6]. Furthermore, [7] also requires overlapping of the action sets of the source and target tasks. Our proposed method is more flexible and lacks this limitation.

Mendez et al. [8] proposed an adaptation of dialog policy using action embeddings. They argued that there is a set of action embeddings can be shared across the source and the target tasks. In practice, action embedding is represented by a hidden layer that is fully connected to both the input and output layers. After training the policies from all the source tasks, the weight parameters that connect the input and the hidden layers are used to initialize the corresponding weights in the target policy's neural network. We used this method as our baseline because this adaptation method requires no additional knowledge or relations between the source and target tasks and is comparable to our proposed method in terms of flexibility.

VI. CONCLUSIONS

We propose a novel method for policy adaptation in dialog management called the *dialog policy reuse algorithm – DPRA*. Our proposed method uses action-relation probability for adaptation, which allows the source task policy to be reused for the action selection of the target task. DPRA learns the action-relation probability from dialog samples in both tasks using a mixture density network, and can immediately derive a policy for the target task. Thus, DPRA learns the target task's policy much more quickly than conventional methods that require RL training and user simulation. Since our proposed method does not employ the user simulator, it can avoid the problem of low performance due to errors in constructing the user simulator.

Future work will conduct a deeper scrutiny of DPRA to determine in which adaptation setting it works and what kind of performance we should expect from it. In addition, DPRA currently does not take the changes in reward dynamics into consideration. We believe that if we can incorporate an estimation of such changes in reward functions of the source and target tasks, we can further improve the performance of the policies learned by DPRA. Finally, we also want to investigate the applications of our method to other settings, such as adaptation for autonomous control tasks.

.

APPENDIX A REWARD FUNCTIONS IN THE EXPERIMENT

Table 4 shows the reward in the source task that received by the agent when selecting an action given the user dialog act (u) and the deception label (d), which are generated by the user simulator.

TABLE 4. Reward definition for the source task.

Dialog state		Rewards		
User DA (u)	d	Offer	Framing	End
Accept	0	–10	–10	+100
	1	–10	+10	-100
Reject	0	+10	+10	–100
	1	–10	+10	–100
Question	0	–10	+10	–100
Hesitate	0	+10	+10	–100

The reward definition for the target task in Section IV is shown in Table 5. Note that *Offer_New* gives +10 reward only if it is selected in the first turn.

APPENDIX B PROOF OF EQUATION 1 AND EQUATION 10

First, we show the proof for (1):

$$\begin{aligned}
\pi(s,b) &= P(b|s)\\
&= \sum_{a\in A} P(b,a|s) \quad (\textit{law of total probability})\\
&= \sum_{a\in A} P(b|a,s)P(a|s)\\
&= \sum_{a\in A} P(b|a,s)\pi(s,a) \quad Q.E.D
\end{aligned}$$

The following is full proof for (9):

$$\begin{aligned}
\sum_{b\in B}\pi(s,b) &= \sum_{b_j\in B} P(b_j|s)\\
&= \sum_{b_j\in B}\sum_{a_i\in A} P(a_i,b_j|s)\\
&= \sum_{b_j\in B}\sum_{a_i\in A} P(a_i|s)P(b_j|a_i,s)\\
&= \sum_{a_i\in A} P(a_i|s)(\sum_{b_j\in B} P(b_j|a_i,s))\\
&= \sum_{a_i\in A} P(a_i|s)(\sum_{b_j\in B} w_{ij}(s))\\
&= \sum_{a_i\in A} P(a_i|s)\cdot 1\\
&\qquad (\textit{sum of component weights is 1})\\
&= \sum_{a\in A} P(a|s) = 1 \quad Q.E.D
\end{aligned}$$

TABLE 5. Reward definition for the target task. Note: *Offer_New* gives +10 reward only if it is selected in the first turn.

Dialog state		Rewards				
User DA (u)	d	Offer_Change	Offer_New	Framing_Answer	Framing_Argue	End
Accept	0	–10	–10	–10	–10	+100
	1	–10	–10	–10	+10	-100
Reject	0	+10	–10	–10	–10	–100
	1	–10	–10	–10	+10	–100
Question	0	–10	–10	+10	–10	–100
Hesitate	0	+10	–10	–10	+10	–100

TABLE 6. Experiment environment details.

	Description
Operating system	Ubuntu 16.04.6 LTS – "xenial"
CPU	Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz
GPU	NVIDIA Corporation GM200 [GeForce GTX TITAN X]
Programming language	Python 2.7.12
Frameworks	PyTorch 1.4.0, CUDA 9.0, NumPy 1.16.6

APPENDIX C DETAILS ON EXPERIMENT ENVIRONMENT

Table 6 shows the details of the environment that was used for our experiments described in Section IV.

REFERENCES

[1] K. Scheffler, & S. Young, "Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning," in *Proc. HLT,* San Diego, CA, USA, 2002, pp. 12–19.

[2] J.D. Williams, P. Poupart, and S. Young, "Partially Observable Markov Decision Processes with Continuous Observations for Dialogue Management," in *Proc.6th SigDial*, Lisbon, Portugal, 2005.

[3] J.D. Williams and S. Young (2005, November). Scaling up POMDPs for Dialog Management: The"Summary POMDP"Method. Presented at IEEE Workshop on Automatic Speech Recognition and Understanding.

[4] A. Lazaric,"Transfer in reinforcement learning: a framework and a survey, " in *Reinforcement Learning,* vol. 12, Springer, Berlin, Heidelberg, 2012, pp. 143–173.

[5] R. A. C. Bianchi, L. A. Celiberto Jr, P. E. Santos, J. P. Matsuura, , & R.L. de Mantaras, "Transferring knowledge as heuristics in reinforcement learning: A case-based approach," in *Artificial Intelligence*, vol. 226, Elsevier, pp. 102–121, 2015.

[6] L. Chen, C. Chang, Z. Chen, B. Tan, M. Gašić, & K. Yu (2018, April). "Policy adaptation for deep reinforcement learning-based dialogue management," in *Proc. ICASSP,* Calgary, AB, Canada, 2018, pp. 6074–6078.

[7] V. Ilievski, C. Musat, A. Hossmann, & M. Baeriswyl, "Goal-oriented chatbot dialog management bootstrapping with transfer learning," in *Proc. IJCAI,* Stockholm, Sweden, 2018, pp. 4115–4121.

[8] J.A. Mendez, A. Geramifard, M. Ghavamzadeh, and B. Liu. (2019, December). Reinforcement Learning of Multi-Domain Dialog Policies Via Action Embeddings. Presented at the 3rd Workshop on Conversational AI: Today's Practice Tomorrow's Potential, NeurIPS. Available: http://alborz-geramifard.com/workshops/neurips19-Conversational-AI/Papers/33.pdf

[9] C.M. Bishop, "Mixture Density Networks," Aston University. Birmingham, UK. February, 1994.

[10] R. Bellman, "A Markovian decision process," *Journal of mathematics and mechanics*, vol. 6, no. 5, pp 679–684, 2957. www.jstor.org/stable/24900506. Accessed on: June, 8, 2020.

[11] C.J.C.H Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no.3–4, pp. 279–292, 1992.

[12] V. Mnih, K. Kavukcuoglu, D. Silver *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. DOI: https://doi.org/10.1038/nature14236

[13] R.H. Crites and A.G. Barto, (1995, November). "An actor/critic algorithm that is equivalent to Q-learning," in *Proc. Advances in Neural Information Processing Systems*, Denver, CO, USA, 1995, pp. 401–408.

[14] V.R. Konda and J.N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Advances in neural information processing systems*, Denver, CO, USA, 2000, pp. 1008–1014.

[15] R.J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no.3-4, pp. 229–256, 1992.

[16] B. Dhingra, L. Li, X. Li, J. Gao, Y.N. Chen, F. Ahmad, & L. Deng, (2017, July), "Towards End-to-End Reinforcement Learning of Dialogue Agents for Information Access, " in *Proc. ACL,* Vancouver, BC, Canada, 2017, pp. 484–495.

[17] T. Hiraoka, G. Neubig,S. Sakti, T. Toda, and S. Nakamura. (2014, August). Reinforcement learning of cooperative persuasive dialogue policies using framing. Presented at COLING 2014, the 25th International Conference on Computational. Available: https://www.aclweb.org/anthology/C14-1161.pdf

[18] K. Yoshino, and T. Kawahara, "Conversational system for information navigation based on POMDP with user focus tracking," in *Computer Speech & Language* vol. 34, no. 1, pp. 275–291, 2015.

[19] T.T. Nguyen, K. Yoshino, S. Sakti, and S. Nakamura, "Dialog Management of Healthcare Consulting System by Utilizing Deceptive Information, " in *Transactions of the Japanese Society for Artificial Intelligence*, vol. 35, no. 1, 2020.

[20] T. Zhao and M. Eskenazi, "Towards End-to-End Learning for Dialog State Tracking and Management using Deep Reinforcement Learning," in *Proc. SIGDIAL,* Los Angeles, CA, USA, 2016, pp. 1–10.

[21] J.V. Serban et al, "A survey of available corpora for building data-driven dialogue systems: The journal version," in *Dialogue & Discourse*, vol. 9, no. 1, pp. 1–49, 2018.

[22] T.T. Nguyen, K. Yoshino, S. Sakti, and S. Nakamura,. (2019, December). Hierarchical Tensor Fusion Network for Deception Handling Negotiation Dialog Model. Presented at the 3rd Workshop on Conversational AI: Today's Practice Tomorrow's Potential, NeurIPS. Available: http://alborz-geramifard.com/workshops/neurips19-Conversational-AI/Papers/10.pdf

[23] J.R. Wilson "Variance reduction techniques for digital simulation," *American Journal of Mathematical and Management Sciences*, vol.4, no. 3-4, pp. 277-–312, 1984.

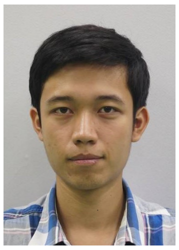
[24] P. L'Ecuyer, "Efficiency improvement and variance reduction," in *Proc. WSC*, pp. 122–132, 1994.

[25] R.S. Sutton, D.A. McAllester, S.P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc NeuRIPS*, Denver, CO, USA, pp. 1057–1063, 2000.

[26] T. Baltrušaitis, P. Robinson, and L.P. Morency, "Openface: an open source facial behavior analysis toolkit," in *Proc. WACV*, Lake Placid, NY, USA, 2016, pp. 1-10. doi: 10.1109/WACV.2016.7477553.

[27] F. Eyben, M. Wöllmer, and B. Schuller. (2010, October). Opensmile: the munich versatile and fast open-source audio feature extractor. Presented at ACM international conference on Multimedia. Available: https://dl.acm.org/doi/pdf/10.1145/1873951.1874246

[28] B. Schuller, S. Steidl, and A. Batliner. (2009). The interspeech 2009 emotion challenge. Presented at Tenth Annual Conference of the International Speech Communication Association. Available: https://mediatum.ub.tum.de/doc/980035/file.pdf



TUNG T. NGUYEN received his Bachelor degree in Computer Science from University of Engineering and Technology, Vietnam National University, in 2014 and the M.Eng degree in Information Science from Nara Institute of Science and Technology (NAIST), in 2017.

He is currently a doctoral candidate at Augmented Human Communication Laboratory, NAIST. His research area includes multimodal processing, reinforcement learning, and spoken dialog system.

Mr. Nguyen is a recipient of the scholarship from the Japanese Ministry of Education, Culture, Sport, Science, and Technology (MEXT) from 2017 to 2020. He is a member of the Japan Association for Natural Language Processing.

KOICHIRO YOSHINO received his B.A. degree in 2009 from Keio University, M.S. degree in informatics in 2011, and a Ph.D. degree in informatics in 2014 from Kyoto University, respectively. From 2014 to 2015, he was a research fellow (PD) of Japan Society for Promotion of Science. From 2015 to 2016, he was a research assistant professor of the Graduate School of Information Science, Nara Institute of Science and Technology (NAIST). Currently, he is an assistant professor of NAIST. He is also a researcher of PRESTO, JST, concurrently. He is working on areas of spoken and natural language processing, especially on spoken dialogue systems. Dr. Koichiro Yoshino received the JSAI SIG-research award in 2013. He is a member of IEEE, ISCA, IPSJ, and ANLP.

SAKRIANI SAKTI received her B.E. degree in Informatics (cum laude) from Bandung Institute of Technology, Indonesia, in 1999. In 2000, she received DAAD-Siemens Program Asia 21st Century Award to study in Communication Technology, University of Ulm, Germany, and received her MSc degree in 2002. During her thesis work, she worked with the Speech Understanding Department, Daimler Chrysler Research Center, Ulm, Germany. Between 2003-2009, she worked as a researcher at ATR SLC Labs, Japan, and during 2006-2011, she worked as an expert researcher at NICT SLC Groups, Japan. While working with ATR-NICT, Japan, she continued her study (2005-2008) with Dialog Systems Group University of Ulm, Germany, and received her Ph.D. degree in 2008. She actively involved in collaboration activities such as Asian Pacific Telecommunity Project (2003-2007), A-STAR, and U-STAR (2006-2011). In 2009-2011, she served as a visiting professor of the Computer Science Department, University of Indonesia (UI), Indonesia. In 2011-2017, she was an assistant professor at the Augmented Human Communication Laboratory, NAIST, Japan. She also served as a visiting scientific researcher of INRIA Paris-Rocquencourt, France, in 2015-2016, under "JSPS Strategic Young Researcher Overseas Visits Program for Accelerating Brain Circulation". Currently, she is a research associate professor at NAIST, as well as a research scientist at RIKEN, the Center of for Advanced Intelligent Project AIP, Japan. She is a member of JNS, SFN, ASJ, ISCA, IEICE, and IEEE. She is also the officer of ELRA/ISCA Special Interest Group on Under-resourced Languages (SIGUL) and a board Member of Spoken Language Technologies for Under-Resourced Languages (SLTU). Her research interests include statistical pattern recognition, graphical modeling framework, deep learning, multilingual speech recognition & synthesis, spoken language translation, affective dialog system, and cognitive communication.

SATOSHI NAKAMURA is Professor of Graduate School of Science and Technology, Nara Institute of Science and Technology, Japan, Project Leader of Tourism Information Analytics Team of RIKEN, Center for Advanced Intelligence Project AIP, Honorarprofessor of Karlsruhe Institute of Technology, Germany, and ATR Fellow. He received his B.S. from Kyoto Institute of Technology in 1981 and Ph.D. from Kyoto University in 1992. He was Associate Professor of Graduate School of Information Science at Nara Institute of Science and Technology in 1994-2000. He was Director of ATR Spoken Language Communication Research Laboratories in 2000-2008 and Vice president of ATR in 2007-2008. He was Director General of Keihanna Research Laboratories and the Executive Director of Knowledge Creating Communication Research Center, National Institute of Information and Communications Technology, Japan in 2009-2010. He is currently Director of Augmented Human Communication laboratory and a full professor of Graduate School of Information Science at Nara Institute of Science and Technology. He is interested in modeling and systems of speech-to-speech translation and speech recognition. He is one of the leaders of speech-to-speech translation research and has been serving for various speech-to-speech translation research projects in the world including C-STAR, IWSLT, and A-STAR. He received Yamashita Research Award, Kiyasu Award from the Information Processing Society of Japan, Telecom System Award, AAMT Nagao Award, Docomo Mobile Science Award in 2007, ASJ Award for Distinguished Achievements in Acoustics. He received the Commendation for Science and Technology by the Minister of Education, Science and Technology, and the Commendation for Science and Technology by the Minister of Internal Affair and Communications. He also received the LREC Antonio Zampolli Award 2012. He has been Elected Board Member of International Speech Communication Association, ISCA, 2011-2019, IEEE Signal Processing Magazine Editorial Board Member since April 2012, IEEE SPS Speech and Language Technical Committee Member since 2013-2016, IEEE Fellow since 2016 and ISCA Fellow since 2020.