POLICY ADAPTATION FOR DEEP REINFORCEMENT LEARNING-BASED DIALOGUE MANAGEMENT

Lu Chen[1], *Cheng Chang*[1], *Zhi Chen*[1], *Bowen Tan*[1], *Milica Gašić*[2], *Kai Yu*[1]

[1]Key Lab. of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering
SpeechLab, Department of Computer Science and Engineering
Brain Science and Technology Research Center
Shanghai Jiao Tong University, Shanghai, China
[2] Engineering Department, University of Cambridge, Cambridge, UK

ABSTRACT

Policy optimization is the core part of statistical dialogue management. Deep reinforcement learning has been successfully used for dialogue policy optimization for a static pre-defined domain. However, when the domain changes dynamically, e.g. a new previously unseen concept (or slot) which can be then used as a database search constraint is added, or the policy for one domain is transferred to another domain, the dialogue state space and action sets both will change. Therefore, the model structures for different domains have to be different. This makes dialogue policy adaptation/transfer challenging. Here a multi-agent dialogue policy (MADP) is proposed to tackle these problems. MADP consists of some *slot-dependent* agents (**S-Agent**s) and a *slot-independent* agent (**G-Agent**). S-Agents have shared parameters in addition to private parameters for each one. During policy transfer, the shared parameters in S-Agents and the parameters in G-Agent can be directly transferred to the agents in extended/new domain. Simulation experiments showed that MADP can significantly speed up the policy learning and facilitate policy adaptation.

Index Terms— dialogue policy, deep reinforcement learning, adaptation, multi-agent

1. INTRODUCTION

A task-oriented spoken dialogue system (SDS) is a system that can continuously interact with a human to accomplish a predefined task (e.g. finding a restaurant or booking a flight). These systems are typically designed according to a structured *ontology* which consists of some concepts (or slots) that a user might wish to use to frame a query. Each slot possesses two attributes: whether it is *requestable* and *informable*. A slot is requestable if the user can request the value of it. An informable slot is one that the user can provide a value for to use as a constraint on their search. Figure 1 is a dialogue example of finding a restaurant. Here *food* and *area* are informable slots, and *phone* is requestable slot.

S: Hello, how may I help you?
U: I want to find a Chinese [*food*] restaurant in the south [*area*] of the town.
S: Hakka is a nice Chinese restaurant in the south area.
U: What is the phone number [*phone*]?
S: 01223568988.
U: Thank you, goodbye!
S: Bye!

S: System
U: User

Fig. 1. An example of task-oriented dialogue.

At every dialogue turn, a dialogue state tracker is to maintain the dialogue belief state for each informable slot, i.e. a distribution of possible values. These belief states with the database query results form the dialogue state $\mathbf{b}$, based on which a dialogue policy π, a mapping function from $\mathbf{b}$ to dialogue action $a = \pi(\mathbf{b})$, is to decide how to respond to the user. Reinforcement learning (RL) methods are usually used to automatically optimize the policy π [1]. More recently, deep reinforcement learning (DRL) methods are adopted for dialogue policies [2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. These policies are often represented by *fully connected* neural networks including deep Q-networks and policy networks, and work well in a static pre-defined domain. However, they are not well suited to the situation where the ontology changes dynamically, unlike the Gaussian process-based approaches [12, 13]. For example, if a new informable slot *pricerange* is added, the dialogue state space and action sets will change, therefore the model structures have to be different, which may significantly degrade performance. Furthermore, policy transfer between different domains is also challenging, because their

This work has been supported by the National Key Research and Development Program of China under Grant No.2017YFB1002102, the China NSFC projects (No. 61573241), and Shanghai International Science and Technology Cooperation Fund (No. 16550720300). Experiments have been carried out on the PI supercomputer at Shanghai Jiao Tong University. The corresponding author is Kai Yu.

state space and action sets both are fundamentally different.

In this paper, we propose a multi-agent [14] dialogue policy (MADP) which facilitates policy adaptation/transfer. MADP consists of some *slot-dependent* agents (**S-Agent**s) and a *slot-independent* agent (**G-Agent**). Each S-Agent focuses on a different informable slot, and G-Agent focuses on slot irrelevant aspects. When making decision, each agent first chooses a candidate action according to its own policy. Final action is then selected from these candidate actions. S-Agent has *shared* parameters in addition to its *private* parameters. The private parameters capture the specific characteristics for each slot, and the shared parameters capture the common characteristics of all slots. With the shared parameters, the skills from one S-Agent can be transferred to another S-Agent, which may speed up the learning process. Moreover, when a new slot in added, the shared parameters can be used to initialize the corresponding S-Agent, i.e. the new S-Agent can be thus transferred some common skills from other S-Agents. To the best of our knowledge, this paper is the first attempt to investigate the policy adaptation/transfer in DRL-based dialogue policy.

2. PROPOSED FRAMEWORK

In this section, we will first present MADP in detail, where S-Agents only have shared parameters. Then we introduce an approach to integrate private parameters and shared parameters for S-Agents, followed by the concrete procedures to do dialogue policy adaptation under MADP framework.

Note that the proposed MADP does not depend on any specific DRL algorithm, hence is compatible with all existing DRL algorithms. Here, we use it within deep Q-network (DQN) framework and call it multi-agent DQN (MADQN).

2.1. Multi-Agent Dialogue Policy

Supposing in a domain there are n informable slots, and the dialogue state $\mathbf{b}$ usually can be decomposed into sub-states $\mathbf{b}_1, \cdots, \mathbf{b}_n$ and $\mathbf{b}_g$, i.e. $\mathbf{b} = \mathbf{b}_1 \oplus \cdots \oplus \mathbf{b}_n \oplus \mathbf{b}_g$. $\mathbf{b}_j (1 \le j \le n)$ is the belief state corresponding to j-th informable slot, and $\mathbf{b}_g$ represents the slot-independent state, e.g. database search results. Similarly, the summary actions can be divided into $n+1$ sets including n slot-dependent action sets $\mathbb{A}_j (1 \le j \le n)$, e.g. *request_slot*$_j$, *confirm_slot*$_j$, *select_slot*$_j$, and one slot-independent action set $\mathbb{A}_g$, e.g. *repeat, offer*.

MADP consists of n slot-dependent agents, i.e. S-Agents $\mathcal{A}_j (1 \le j \le n)$, each one corresponding to an informable slot, and a slot-independent agent, i.e. G-Agent $\mathcal{A}_g$. Figure 2(a) provides an overview of MADP with DQN as the DRL algorithm.

- The input of $\mathcal{A}_j$ is $\mathbf{b}_j$, and the output is the Q-values $\mathbf{q}_j$ corresponding to actions in $\mathbb{A}_j$, i.e. $\mathbf{q}_j = [Q(\mathbf{b}_j, a_{j1}), \cdots, Q(\mathbf{b}_j, a_{jm_s})]$, where $a_{jk} (1 \le k \le m_s) \in \mathbb{A}_j$.

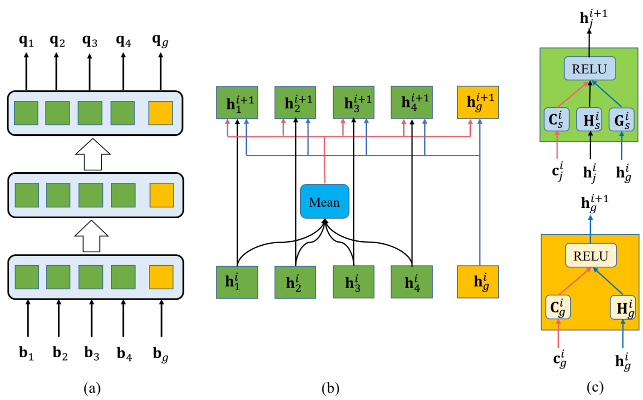

Fig. 2. MADQN with 4 S-Agents (green) for 4 informable slots and the G-Agent (yellow). (a) The overview of MADQN with 3 hidden layers, i.e. two communication steps. (b) A single communication step, i.e. the details of connection between two hidden layers in (a). (c) The hidden layers for a S-Agent (top) and the G-Agent (bottom).

- The input of $\mathcal{A}_g$ is $\mathbf{b}_g$, and the output is the Q-values $\mathbf{q}_g$ corresponding to actions in $\mathbb{A}_g$, i.e. $\mathbf{q}_g = [Q(\mathbf{b}_g, a_{g1}), \cdots, Q(\mathbf{b}_g, a_{gm_g})]$, where $a_{gk} (1 \le k \le m_g) \in \mathbb{A}_g$.

To obtain the Q-values $\mathbf{q}$ for all actions, the outputs of all agents are concatenated as shown in Figure 2(a), i.e. $\mathbf{q} = \mathbf{q}_1 \oplus \cdots \oplus \mathbf{q}_n \oplus \mathbf{q}_g$. When making decision, the action is chosen according to $\mathbf{q}$.

These agents have some internal messages exchange [15, 16] when they calculate their own Q-values. As shown in Figure 2(b), after i-th hidden layer, both $\mathcal{A}_j$ and $\mathcal{A}_g$ will output some messages. Here, we just use the output of hidden layer as messages, i.e. $\mathbf{h}_j^i$ for $\mathcal{A}_j$ and $\mathbf{h}_g^i$ for $\mathcal{A}_g$. At (i+1)-th hidden layer, the input of $\mathcal{A}_j$ includes the output of its previous layer $\mathbf{h}_j$, the message from other S-Agents $\mathbf{c}_j^i$,

$$\mathbf{c}_j^i = \frac{1}{n-1} \sum_{1 \le l \le n, l \ne j} \mathbf{h}_l^i,$$

and the message from G-Agent $\mathbf{g}^i$, $\mathbf{g}^i = \mathbf{h}_g^i$. Based on $\mathbf{h}_j^i$, $\mathbf{c}_j^i$ and $\mathbf{g}^i$, the output of (i+1)-th hidden layer of $\mathcal{A}_j$ is shown at the top of Figure 2(c) [1], i.e.

$$\mathbf{h}_j^{i+1} = \sigma(\mathbf{H}_s^i \mathbf{h}_j^i + \mathbf{C}_s^i \mathbf{c}_j^i + \mathbf{G}_s^i \mathbf{g}^i), \tag{1}$$

where σ is a non-linear activation function, e.g. RELU. $\theta_s \triangleq \{\mathbf{H}_s^i, \mathbf{C}_s^i, \mathbf{G}_s^i\}_{i=1}^L$ are weight matrices, i.e. parameters shared across all slot-dependent agents.

Similarly, at (i+1)-th hidden layer of $\mathcal{A}_g$, the input includes the output of its previous layer $\mathbf{h}_g^i$ and the message from S-Agents $\mathbf{c}_g^i$,

$$\mathbf{c}_g^i = \frac{1}{n} \sum_{1 \le j \le n} \mathbf{h}_j^i.$$

[1] For simplicity, the bias term is omitted.

Based on $\mathbf{h}_g^i$ and $\mathbf{c}_g^i$, the output of (i+1)-th hidden layer of $\mathcal{A}_g$ is shown at the bottom of Figure 2(c), i.e.

$$\mathbf{h}_g^{i+1} = \sigma(\mathbf{H}_g^i \mathbf{h}_g^i + \mathbf{C}_g^i \mathbf{c}_g^i), \tag{2}$$

where $\theta_g \triangleq \{\mathbf{H}_g^i, \mathbf{C}_g^i\}_{i=1}^L$ are weight matrices.

In summary, MADQN can be viewed as a structured DQN with hidden layers $\mathbf{h}^{i+1} = \sigma(\mathbf{T}^i \mathbf{h}^i)$ where $\mathbf{h}^i$ is the concatenation of all $\mathbf{h}_j^i (1 \leq j \leq n)$ and $\mathbf{h}_g^i$, i.e. $\mathbf{h}^i = \mathbf{h}_1^i \oplus \cdots \oplus \mathbf{h}_n^i \oplus \mathbf{h}_g^i$. $\mathbf{T}^i$ takes the form

$$\mathbf{T}^i = \begin{pmatrix} \mathbf{H}_s^i & \overline{\mathbf{C}}_s^i & \cdots & \overline{\mathbf{C}}_s^i & \mathbf{G}_s^i \\ \overline{\mathbf{C}}_s^i & \mathbf{H}_s^i & \cdots & \overline{\mathbf{C}}_s^i & \mathbf{G}_s^i \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \overline{\mathbf{C}}_s^i & \overline{\mathbf{C}}_s^i & \cdots & \mathbf{H}_s^i & \mathbf{G}_s^i \\ \overline{\mathbf{C}}_g^i & \overline{\mathbf{C}}_g^i & \cdots & \overline{\mathbf{C}}_g^i & \mathbf{H}_g^i \end{pmatrix}, \tag{3}$$

where $\overline{\mathbf{C}}_s^i = \frac{1}{n-1}\mathbf{C}_s^i$ and $\overline{\mathbf{C}}_g^i = \frac{1}{n}\mathbf{C}_g^i$. The training process of MADQN is similar to vanilla DQN [17] except that the weight matrix is structured as equation (3).

2.2. Shared-Private Weighted Network

In simple domains where slots have similar characteristics, shared parameters are sufficient to capture the differences between different slots. However, in more complex domains, private parameters are needed to capture their characteristics. Here, we propose a shared-private weighted network (SPWN) to introduce private parameters in S-Agent.

In SPWN, each S-Agent $\mathcal{A}_j$ has its own private parameters $\theta_j \triangleq \{\mathbf{H}_j^i, \mathbf{C}_j^i, \mathbf{G}_j^i\}_{i=1}^L$ in addition to the shared parameters $\theta_s \triangleq \{\mathbf{H}_s^i, \mathbf{C}_s^i, \mathbf{G}_s^i\}_{i=1}^L$ across all slots. For each input $\mathbf{b}_j$, the agent first computes the outputs with θ_j and θ_s in parallel, then takes the weighted average of two outputs to obtain the final output $\mathbf{q}_j$, i.e.

$$\mathbf{q}_j = \alpha \text{Net}(\mathbf{b}_j; \theta_j) + (1-\alpha)\text{Net}(\mathbf{b}_j; \theta_s), \tag{4}$$

where $\alpha \in [0, 1]$ is the weight. The more complex the domain/task, the larger α should be.

2.3. Policy Adaptation

The general procedure of MADP-based policy adaptation, namely Shared-Private-Adaptation (**SP-Adapt**): (1) Initialize the shared parameters θ_s, the private parameters θ_j for S-Agent $\mathcal{A}_j$ and the parameters θ_g for G-Agent $\mathcal{A}_g$. (2) Train the multi-agent policy in original domain. (3) When domain is extended, the private parameters for new S-Agent θ_j are initialized by θ_s, or when it's transferred to a new domain, the shared parameters θ'_s for S-Agents and the θ'_g for G-Agent are initialized by the corresponding parameters in the original domain, i.e. $\theta'_s \leftarrow \theta_s$ and $\theta'_g \leftarrow \theta_g$. The private parameters θ'_j are initialized by θ_s with added noise, i.e. $\theta'_j \leftarrow \theta_s + \mathcal{N}(\mathbf{0}, \sigma_{noise}\mathbf{I})$. (4) Continuously train the policy in the extended/new domain.

In addition, if the original domain and the extended/new domain are relatively simple and their interactive environments follow similar patterns, it would be sufficient to use a simplified MADP framework with no private parameters for S-Agents to achieve satisfying results. For an extended/new domain, the shared parameters for S-Agents and the parameters for G-Agent are initialized by the corresponding parameters in the original domain. This is called Shared-Adaptation (**S-Adapt**) procedure.

3. EXPERIMENTS

Two objectives are set for the experiments: (1) Comparing the performances of policy learning on single domain between our proposed MADP and traditional models. (2) Comparing the policy adaptation performances of different models and investigating the benefits of our proposed MADP framework.

Here the purpose of the user's interacting with SDS is to find restaurant/tourist information in the Cambridge (UK) area [18, 19]. There are three domains: *DSTC2_Simple*, *DSTC2*, and *DSTC3*. *DSTC2_Simple* and *DSTC2* are both restaurant information domain. *DSTC2_Simple* has 6 slots of which 3 can be used by the system to constrain the database search. *DSTC2* has an additional *pricerange* slot. *DSTC3* is tourist information domain, and it has all slots on *DSTC2* and 5 new slots. An agenda-based user simulator [20] with semantically conditioned LSTM-based natural language generator (SC-LSTM-NLG) [21] was implemented to emulate the behavior of the human user. With SC-LSTM-NLG, the semantics-level dialogue acts from user simulator are converted to N-based utterance list as ASR results. An SVM-based semantic parser [22] was trained on DSTC2/3 datasets. The semantic error rates on *DSTC2* and on *DSTC3* are ∼0.15 and ∼0.40 respectively.

For the reward, at each turn, a reward of -0.05 is given to the policy. At the end of the dialogue, a reward of $+1$ is given for dialogue success.

3.1. Fast Policy Learning

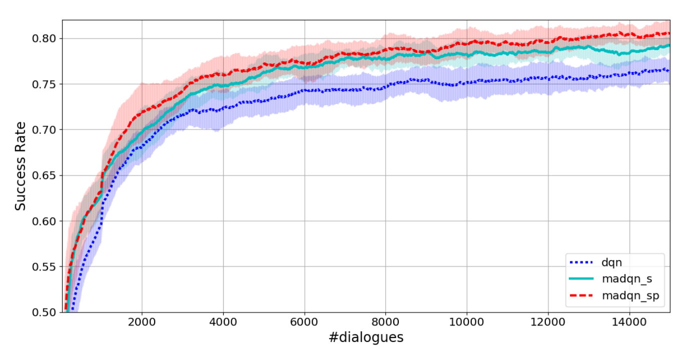

Fig. 3. The learning curves in *DSTC2* domain.

In this section, our proposed multi-agent methods are compared with other methods on *DSTC2* when there is no adaptation. As shown in Figure 3, three models are compared: (1) `dqn` is a dropout DQN proposed in [8], which performs much better than vanilla DQN. It has two hidden layers, each with 128 nodes. The dropout rate is 0.2. (2) `madqn_s` is MADQN as shown in Figure 2. Each S-Agent has no private parameters. All agents have three hidden layers, i.e. two communications steps. The sizes of each hidden layer for S-Agents and G-Agent are 32 and 62 respectively. Similarly, it has dropout layers, and the dropout rate is 0.1. (3) `madqn_sp` is similar to `madqn_s` except that each S-Agent has private parameters in addition to the shared parameters as described in section 2.2.

We can find that multi-agent models (`madqn_s` and `madqn_sp`) achieve faster learning speed at the early stage of learning and better convergence performance. While comparing `madqn_s` with `madqn_sp`, they have little difference in learning speed and final performance, which indicates that the shared parameters are sufficient to capture the different characteristics of slots on *DSTC2*.

3.2. Policy Adaptation

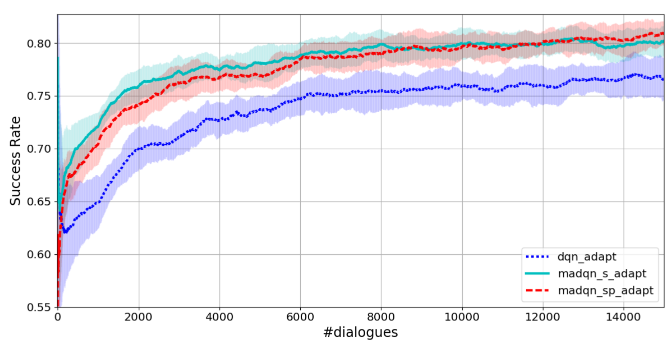

Fig. 4. The learning curves of policy adaptation from *DSTC2_Simple* to *DSTC2*.

In this section, we will compare MADQN with DQN for policy adaptation. Figure 4 and Figure 5 are the results of policy transfer from *DSTC2_Simple* to *DSTC2* and the results of policy transfer from *DSTC2* to *DSTC3* respectively. Here, three policies are compared: For `dqn_adapt`, `dqn` is first pre-trained in the original domain with 15000 dialogues. When the domain is extended to the new domain, the number of input features and the summary action space both increase. The corresponding new weights to input layer and output layer are randomly initialized with $\mathcal{N}(0, 0.01)$ before continuously trained in the new domain. For `madqn_s_adapt` (or `madqn_sp_adapt`), `madqn_s` (or `madqn_sp`) is first pre-trained in the original domain with 15000 dialogues. Then it is transferred to a new domain following the procedure of *S-Adapt* (or *SP-Adapt*) described in section 2.3.

From Figure 4, we can find that MADQN-based models (`madqn_s_adapt` and `madqn_sp_adapt`) learn much

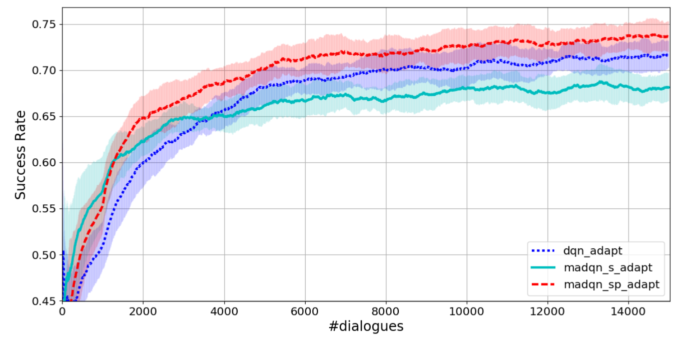

Fig. 5. The learning curves of policy adaptation from *DSTC2* to *DSTC3*.

faster than `dqn_adapt` when they are transferred from *DSTC2_Simple* to *DSTC2*. The newly added S-Agent for new slot *pricerange* can use the shared parameters, i.e. transfer some skills from other agents. Comparing `madqn_s_adapt` in Figure 4 with `madqn_s` in Figure 3, obvious improvement at the early learning process can be observed, which demonstrates the effectiveness of knowledge transfer through shared parameters.

Comparing `madqn_sp_adapt` with `madqn_s_adapt`, we can find that introducing private parameters in MADQN does not lead to any improvement on *DSTC2*. The reason lies in that the shared parameters are sufficient to capture the differences between 4 slots

In Figure 5, MADQN without private parameters in S-Agents (`madqn_s_adapt`) also learns much faster than `dqn_adapt` at the beginning. However, it reaches a sub-optimal convergence at the end. On *DSTC3*, there are 8 informable slots, and the shared parameters are not sufficient to capture the differences between these slots. So introducing private parameters (`madqn_sp_adapt`) can significantly boost the performance.

4. CONCLUSION

This paper proposed a DRL-based multi-agent dialogue policy (MADP) framework, consisting of a *slot-independent* agent, G-Agent, and some *slot-dependent* agents, S-Agents. Under this framework, shared parameters in S-Agents can be easily transferred from one domain to another, ensuring a good initialization and fast subsequent learning in the new domain. Experiments showed that the proposed MADP-based models learn faster than traditional models in the single domain, and achieve efficient and effective policy adaptation from original domain to extended/new domain. However, the results also indicated that in complex domains, e.g. *DSTC3*, policy transfer is still challenging. Although policies are pre-trained in the original domain, their initial success rates on DSTC3 are low. Further investigation of methods for improving the efficiency of policy transfer is needed.

5. REFERENCES

[1] Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams, "Pomdp-based statistical spoken dialog systems: A review," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1160–1179, 2013.

[2] Heriberto Cuayáhuitl, Simon Keizer, and Oliver Lemon, "Strategic dialogue management via deep reinforcement learning," *NIPS DRL Workshop*, 2015.

[3] Mehdi Fatemi, Layla El Asri, Hannes Schulz, Jing He, and Kaheer Suleman, "Policy networks with two-stage training for dialogue systems," in *Proceedings of SIGDIAL*, 2016, pp. 101–110.

[4] Tiancheng Zhao and Maxine Eskenazi, "Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning," in *Proceedings of SIGDIAL*, 2016, pp. 1–10.

[5] Zachary C Lipton, Jianfeng Gao, Lihong Li, Xiujun Li, Faisal Ahmed, and Li Deng, "Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking," *arXiv preprint arXiv:1608.05081*, 2016.

[6] Pei-Hao Su, Pawel Budzianowski, Stefan Ultes, Milica Gasic, and Steve Young, "Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management," in *Proceedings of SIGDIAL*, 2017.

[7] Jason D Williams, Kavosh Asadi, and Geoffrey Zweig, "Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning," in *Proceedings of ACL*, 2017.

[8] Lu Chen, Xiang Zhou, Cheng Chang, Runzhe Yang, and Kai Yu, "Agent-aware dropout dqn for safe and efficient on-line dialogue policy learning," in *Proceedings of EMNLP*, 2017.

[9] Cheng Chang, Runzhe Yang, Lu Chen, Xiang Zhou, and Kai Yu, "Affordable on-line dialogue policy learning," in *Proceedings of EMNLP*, 2017, pp. 2190–2199.

[10] Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz, "End-to-end task-completion neural dialogue systems," in *Proceedings of IJCNLP*, 2017, vol. 1, pp. 733–743.

[11] Bing Liu, Gokhan Tur, Dilek Hakkani-Tur, Pararth Shah, and Larry Heck, "End-to-end optimization of task-oriented dialogue model with deep reinforcement learning," *NIPS Workshop in Conversational AI*, 2017.

[12] Milica Gašic, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young, "Pomdp-based dialogue manager adaptation to extended domains," in *Proceedings of SIGDIAL*, 2013.

[13] Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young, "Dialogue manager domain adaptation using gaussian process reinforcement learning," *Computer Speech & Language*, vol. 45, no. Supplement C, pp. 552 – 569, 2017.

[14] Lucian Busoniu, Robert Babuska, and Bart De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2)*, 2008.

[15] Sainbayar Sukhbaatar, Rob Fergus, et al., "Learning multiagent communication with backpropagation," in *Proceedings of NIPS*, 2016, pp. 2244–2252.

[16] Jakob Foerster, Yannis M Assael, Nando de Freitas, and Shimon Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proceedings of NIPS*, 2016, pp. 2137–2145.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[18] Matthew Henderson and Blaise Thomson, "The second dialog state tracking challenge," in *Proceedings of SIGDIAL*, Stroudsburg, PA, USA, 2014, vol. 263, ACL.

[19] Matthew Henderson, Blaise Thomson, and Jason D Williams, "The third dialog state tracking challenge," in *Proceedings of IEEE SLT*. IEEE, 2014, pp. 324–329.

[20] Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young, "Agenda-based user simulation for bootstrapping a pomdp dialogue system," in *Proceedings of NAACL*, Morristown, NJ, USA, 2007, pp. 149–152, ACL.

[21] Tsung-Hsien Wen, Milica Gašić, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve J Young, "Semantically conditioned lstm-based natural language generation for spoken dialogue systems," *Proceedings of EMNLP*, 2015.

[22] Su Zhu, Lu Chen, Kai Sun, Da Zheng, and Kai Yu, "Semantic parser enhancement for dialogue domain extension with little data," in *Proceedings of IEEE SLT*, 2014, pp. 336–341.