INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Domain-Specific Multi-Agent Dialog Policy Learning in Multi-Domain Task-Oriented Scenarios

Li Tang[1], *Yuke Si*[1], *Longbiao Wang*[1,2,*], *Jianwu Dang*[1,2,3,*]

[1]Tianjin Key Laboratory of Cognitive Computing and Application,
College of Intelligence and Computing, Tianjin University, Tianjin, China
[2]Tianjin University-Huiyan Technology Joint AI Lab, Tianjin University, Tianjin, China
[3]Japan Advanced Institute of Science and Technology, Ishikawa, Japan
{tangli1208,siyuke,longbiao_wang}@tju.edu.cn, jdang@jaist.ac.jp

Abstract

Traditional dialog policy learning methods train a generic dialog agent to address all situations. However, when the dialog agent encounters a complicated task that involves more than one domain, it becomes difficult to perform concordant actions due to the hybrid information in the multi-domain ontology. Inspired by a real-life scenario at a bank, there are always several specialized departments that deal with different businesses. In this paper, we propose Domain-Specific Multi-Agent Dialog Policy Learning (DSMADPL), in which the dialog system is composed of a set of agents where each agent represents a specialized skill in a particular domain. Every domain-specific agent is first pretrained with supervised learning using a dialog corpus, and then they are jointly improved with multi-agent reinforcement learning. When the dialog system interacts with the user, in each turn the system action is decided by the actions of relevant agents. Experiments conducted on the commonly used MultiWOZ dataset prove the effectiveness of the proposed method, in which dialog success rate increases from 55.0% for the traditional method to 67.2% for our method in multi-domain scenarios.

Index Terms: task-oriented, dialog policy, domain-specific, multi-agent, reinforcement learning

1. Introduction

In traditional research, a dialog system is usually built in a complex pipeline [1] that contains five components, namely, natural language understanding (NLU) [2, 3], dialog state tracking (DST) [4, 5], dialog policy learning (DPL) [6, 7] and natural language generation (NLG) [8, 9]. Dialog policy, which decides the next system action, plays a decisive role in pipeline structured dialog systems. More recently, dialog modeling has been viewed as a decision-making problem and formulated as a Partially Observable Markov Decision Problem (POMDP) [10, 11], which can be addressed by Reinforcement Learning (RL) [12, 13, 14]. RL based dialog agents require substantial interaction with real users and further adjust policy in an online fashion [15, 16, 8]. Researchers firstly proposed to train a user simulator which learns from large amounts of human conversations to imitate real users and interacts with the dialog agent in an offline scenario Later, to avoid explicitly building the user simulator with heavy domain expertise, some studies have proposed regarding the user simulator as another dialog agent and jointly training the user agent and the system agent [17, 18, 19]. The user agent and the system agent are first bootstrapped by

*Corresponding author.

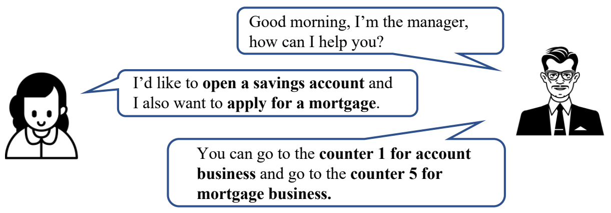
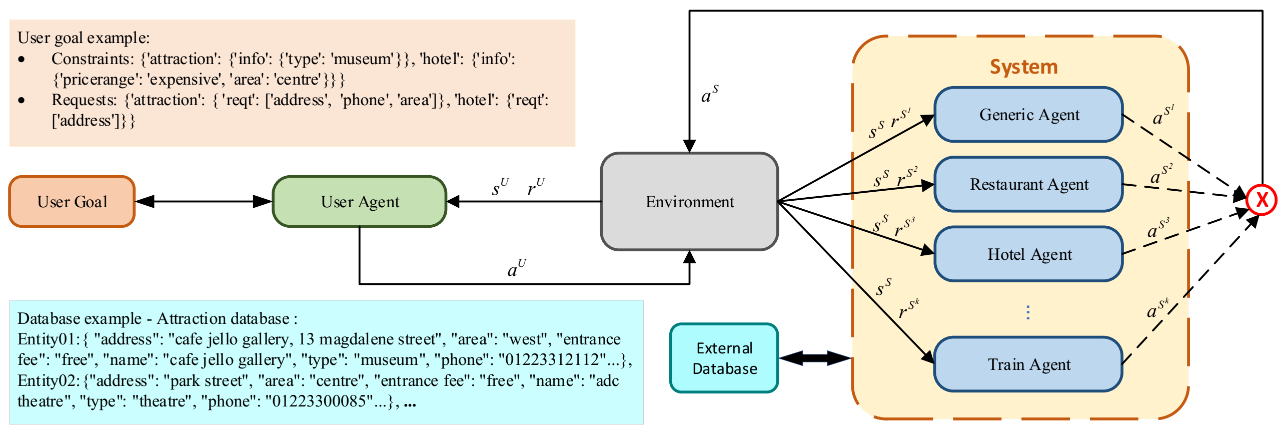

Good morning, I'm the manager, how can I help you?
I'd like to **open a savings account** and I also want to **apply for a mortgage**.
You can go to the **counter 1 for account business** and go to the **counter 5 for mortgage business.**

Figure 1: *Dialog between a client and a lobby manager at a bank: a real-life scenario.*

learning directly from human dialogs with supervised learning, and then they interact with each other in a collaborative pattern, in which the user agent and system agent are trained simultaneously.

In the past, task-oriented dialog agents were applied in simple scenarios which generally concerned only one domain, such as flight booking or restaurant reservation [20]. However, with the development of virtual assistants and smart life, there are widespread demands for dialog agents for application to complex scenarios. Considering a situation in which one is making a travel schedule, a traveler must order flight tickets, book hotels, choose attractions and plan other aspects. With so many subtasks to complete, a dialog agent must acquire multi-domain knowledge to fulfill the user goal. However, with more domains involved in the dialog, there are more contradictions and conflicts crossing the domain ontology. Traditional methods [17, 18, 19] generally train a universal agent to solve all the situations even if the user goal involves multiple domains. Although the universal agent can learn and accomplish tasks in many different domains, if the user and the agent do not share the correct domain information, they may misunderstand each other in many cases since the same words have different meanings in different domains.

To handle the problem mentioned above, in this paper, we propose a multi-agent learning method to train a set of domain-specific agents where each agent concentrates on its own domain ontology and provides relevant domain suggestions to the user. The intuition is from the real-life scenario of going to the bank to transact some business: as presented in Figure 1, the lobby manager will first ask some questions and then tell us to go to specific counters to execute transactions. With professional staffs in their own fields to deal with our business, our questions will receive more specific and more detailed answers. Similar with the above situation, we proposed Domain-Specific Multi-Agent Dialog Policy Learning (DSMADPL) to deal with


User goal example:

- Constraints: {'attraction': {'info': {'type': 'museum'}}, 'hotel': {'info': {'pricerange': 'expensive', 'area': 'centre'}}}
- Requests: {'attraction': { 'reqt': ['address', 'phone', 'area']}, 'hotel': {'reqt': ['address']}}

Database example - Attraction database :
Entity01:{ "address": "cafe jello gallery, 13 magdalene street", "area": "west", "entrance fee": "free", "name": "cafe jello gallery", "type": "museum", "phone": "01223312112"...}, Entity02:{"address": "park street", "area": "centre", "entrance fee": "free", "name": "adc theatre", "type": "theatre", "phone": "01223300085"...}, ...

Figure 2: *Illustration of the proposed DSMADPL. The user agent interacts with the system based on the user goal. The dialog system is composed of 8 domain-specific agents that concentrate on specific domain knowledge. The system actions at each turn are decided by the related specialized agents.*

complex traveling scenarios and demonstrated the effectiveness in the commonly used MultiWOZ [21] dataset, which is a travel booking scenario with multiple domains. Specifically, we design individual dialog agent for each domain of the multi-domain corpus, and these agents first learn their own domain knowledge from the corpus with supervised learning, and then are jointly trained using multi-agent reinforcement learning to achieve maximal multi-domain dialog success. Since the domain-specific agent concentrates on its own domain knowledge, each agent only shares relevant domain information with the user, which avoids confusing messages being conveyed and misunderstood.

2. Domain-Specific Multi-Agent Dialog Policy Learning (DSMADPL)

Our DSMADPL model is illustrated in Figure 2 and consists of two parts: the user agent and the system agent, which is composed of several domain-specific agents. We deal with eight domains but only plot four domain-specific agents in the figure.

2.1. Multi-Domain Dialog Task

In this part, we describe the entire workflow of the multi-domain dialog task. Specifically, the user is first given a goal *G=(C,R)*, where *C* represents the user constraints and *R* the user requests, and the system can access an external database *DB* containing available entities and their corresponding information. Taking the hotel booking task as an example, the constraints given to the user may include finding an expensive hotel in the center of the city, and the requests may include asking for the address of the hotel. When *G* contains constraints and requests from more than one domain, the dialog becomes complicated and it becomes a multi-domain task. The goal is fulfilled only when all subtasks in each domain are completed. The dialog process can be viewed as a trajectory of state-action pairs $\left\{\left(s_0^U, a_0^U, s_0^S, a_0^S,\right);\left(s_1^U, a_1^U, s_1^S, a_1^S,\right);\ldots\right\}$[1], where the user agent and the system agent make decisions according to their respective dialog policy $\pi_U(a^U|s^U), \pi_S(a^S|s^S)$. In our proposed method, the system agent is composed of several domain-specific agents, so the system action is given from these domain-specific agents together according to the domains involved at the current turn. The details are presented in Section 2.3.

[1]In this paper, the U and S in the upper right of variables represent the user and the system, respectively, and the number/letter in the bottom right denotes the turn of dialog.

2.2. User Agent

2.2.1. *User Policy*

The policy of user agent π_U takes the user dialog state $s_t^U = [a_{t-1}^S; a_{t-1}^U; g_t; c_t]$ as the input, which concatenates four parts: (I) last turn system action a_{t-1}^S; (II) last turn user action a_{t-1}^U; (III) the goal state g_t that represents for indicates the remaining constraints C and requests R that need to be mentioned by the user; (IV) inconsistency vector c_t [22] that denotes the inconsistency between the system action and user constraints C, and outputs the current turn user action a^U and a terminal signal T that suggests that the dialog is finished when the user goal is fulfilled.

$$\pi_U = \pi(a_t^U, T|s_t^U), \quad T = \begin{cases} 1, & \text{dialog finished} \\ 0, & \text{dialog continues} \end{cases} \tag{1}$$

2.2.2. *User Reward*

In the reinforcement learning algorithm, the agent policy updates according to the reward it receives, and its goal is to maximize the accumulative reward of a trajectory. In this paper, we handcraft the reward for each agent considering its unique characteristics. The reward of the user is designed considering the following two parts: (I) turn-level reward: a penalty score of -5 will be assigned if the user does not give any actions at the turn, and a penalty score of -1 will be assigned for every constraint not mentioned before requesting information; (II) dialog-level reward: the user agent will receive a score of 20 if it gives all the constraints C and requests R at the end of the dialog or a -5 penalty score if this is not the case.

2.2.3. *Optimization*

The optimization of the user agent includes two stages. First, the user policy is pretrained with supervised learning using dialogs from the MultiWOZ dataset. We improve the user policy with β-weighted logistic regression as follow:

$$L(X,Y;\beta) = -[\beta \cdot Y^T log\sigma(X) + (I-Y)^T log(I-\sigma(X))], \quad (2)$$

where X is the user state and Y is the user action. During the second stage, we use *Actor-Critic* reinforcement learning to improve the user policy. In *Actor-Critic*, the actor is the user policy π_U that generates the user actions, and the critic is a value function $V(s_t)$ that evaluates the behavior of the actor. We improve the value function by minimizing the mean squared loss function, defined as follow:

$$L_V^U = (r^U + \gamma V_{\theta'}^U(s^{'U}) - V_\theta^U(s^U))^2, \quad (3)$$

where $\gamma \in [0,1]$ is a discount factor, and $V_{\theta'}^U(s^{'U})$ is the target value function [23] that is only periodically updated. Then we can obtain the *Actor-Critic* loss as follow:

$$L_{\pi_U}(\phi) = log\pi_\phi(a^U|s^U)[r + \gamma V_{\theta'}^U(s^{'U}) - V_\theta^U(s^U)], \quad (4)$$

where the user policy is parameterized by ϕ.

2.3. Domain-Specific System Agents

2.3.1. System Policy of Domain-Specific agents

The policy of system agent π_S takes system dialog state $s_t^S = [a_{t-1}^S; a_t^U; b_t; q_t]$ as input, which is the concatenation of four parts: (I) last turn system action a_{t-1}^S; (II) current turn user action a_t^U; (III) the belief state b_t [24] that keeps track of constraints and requests mentioned by the user in dialog history; (IV) embedding vectors of the number of query results q_t from *DB*, and outputs system action a^S at the current turn. The system action is composed of a set of *dialog act*, which is an abstract representation of an intention and is represented in the form of *domain-intent-slot-value* (e.g., hotel-inform-number-00001111).

Traditional methods train a universal agent to process all the domain knowledge, which treats the inputs from different domains identically. However, in this way it is difficult for the system agent to distinguish and utilize the information from different domains. Therefore, in this paper, we propose to use several domain-specific agents to cooperatively determine the system actions for every turn. Specifically, we design 7 agents for each domain in the ontology of the MultiWOZ dataset and a generic agent for those that do not belong to these 7 domains (such as general-greet-none). At every turn of the conversation with the user agent, each domain-specific agent will generate actions a^{S^k} based on the system state s^S, and the system action a^S of the current turn is composed of the domain-specific agents involved at the current turn. We consider whether the domain-specific agent is involved at every turn according to the domains mentioned in the current turn user action:

$$\pi_{S^k} = \pi(a_t^{S^k}|s_t^S), \quad (5)$$

$$a^S = (a^{S^1} \cup a^{S^2} ... \cup a^{S^k}), k \in [1,8] \quad (6)$$

2.3.2. System Reward for Domain-Specific agents

Considering that system agents aim to fulfill the user's goal by querying DB, system rewards are designed for the following three situations: (I) for the domain not involved in the current turn, the domain-specific agent will receive a penalty score of -5 if it gives actions; (II) for the domain involved in the current turn, the domain-specific agent will receive a penalty score of -5 if it gives an empty action, and it will receive a penalty score of -1 for not answering the user's request; (III) at the end of dialogs, the involved domain-specific agent will receive 20 scores for successful dialog or a -5 penalty score for failure.

2.3.3. Optimization for Domain-Specific agents

The optimization of the system agents includes two stages. Firstly, each domain-specific agent is pretrained with supervised learning using a dialog corpus. The loss used to improve the agents' policy is the same as formula (1). The policy of each domain-specific agent is then improved with *Actor-Critic* reinforcement learning by interacting with the user agent and generating new dialog. The value network for each domain-specific agent is the same: $V_\beta^S(s^S)$, and the critic optimization aims is to minimize the squared error between the temporal difference (TD) target and the estimated value:

$$L_V^{S^k} = (r^{S^k} + \gamma V_{\beta'}^S(s^{'S}) - V_\beta^S(s^S))^2, \quad (7)$$

where $k \in [1,8]$ represents the k^{th} domain-specific agent, $\gamma \in [0,1]$ is a discount factor, and $V_{\theta'}^{S^k}(s^{'S^k})$ is the target value function that is only periodically updated. We then obtain the *Actor-Critic* loss for each domain-specific agent as follow:

$$L_{\pi_{S^k}}(\omega^{S^k}) = log\pi_{\omega^{S^k}}(a^{S^k}|s^S)[r^{S^k} + \gamma V_{\beta'}^S(s^{'S}) - V_\beta^S(s^S)], \quad (8)$$

where the policy for each domain-specific agent is parameterized by ω^{S^k}.

3. Experiments

3.1. Dataset

Experiments are conducted on MultiWOZ [21], a multi-domain task-oriented dialog corpus that contains 7 domains: restaurant, hotel, attraction, taxi, train, hospital and police. All the dialog turns are split into 7 specific domains based on the domain tags. The dialog turns that do not belong to these 7 domains are included into a generic domain. The system is thus composed of a total of 8 domain-specific agents in our experimental setting.

3.2. Experimental Settings

SL: Supervised learning directly uses the dialog act annotations and trains the user and the system agents simply by behavior cloning.
MADPL: Multi-agent dialog policy learning [19] trains a user agent and a generic system agent with RL.
SL-DSMADPL (our model): The user and 8 domain-specific system agents are trained with supervised learning using Eq (2). Every agent policy is implemented with two-hidden-layer MLPs. The action space for user agents is 80, and 18, 32, 33, 41, 3, 31, 10, and 4 for domain-specific agents with the respective order of generic, restaurant, attraction, hotel, hospital, train, taxi, and police.
DSMADPL (our model): Based on the SL, we improve the user and domain-specific system agents with *Actor-Critic* RL using Eq (4) and Eq (8), respectively. The value function in *Actor-Critic* is implemented in two hidden layers, and the activation function is $Relu$. $RMSprop$ is used for optimization, and batch size is set to 32. Learning rate for user policy and

Table 1: *Performance of the interaction between user agent and system agents according to the different numbers of domains in the user goal. There are 746 and 252 test cases containing two domains and three domains, respectively. The overall results include all 1000 test cases.*

	Two domains					**Three domains**					**Overall**				
	Turn	**Success**	**Match**	**Inform**		**Turn**	**Success**	**Match**	**Inform**		**Turn**	**Success**	**Match**	**Inform**	
				Rec	**F1**				**Rec**	**F1**				**Rec**	**F1**
SL	7.353	36.6	82.613	74.215	68.377	10.778	13.1	57.333	61.019	71.442	8.214	30.7	76.688	69.534	69.343
SL-DSMDPL (our)	**6.393**	42.2	74.456	78.523	64.986	**8.071**	19.0	55.647	66.889	69.505	**6.819**	36.4	70.049	74.408	66.408
MADPL	9.826	66.4	**94.274**	90.369	**70.951**	18.373	21.0	71.333	75.083	**78.586**	11.975	55.0	**88.906**	84.945	**73.245**
DSMADPL (our)	6.899	**72.5**	89.404	**94.308**	67.757	10.611	**51.1**	**76.174**	**90.753**	78.234	7.831	**67.2**	86.321	**93.054**	71.110

Table 2: *Performance of the interaction between benchmark user simulator Agenda and the system agents according to the different numbers of domains in the user goal. The same user goals are sampled as Table 1.*

	Two domains					**Three domains**					**Overall**				
	Turn	**Success**	**Match**	**Inform**		**Turn**	**Success**	**Match**	**Inform**		**Turn**	**Success**	**Match**	**Inform**	
				Rec	**F1**				**Rec**	**F1**				**Rec**	**F1**
SL	7.453	94.5	95.473	98.646	**80.240**	15.611	42.5	55.180	79.181	82.971	9.509	81.4	86.089	91.713	**81.089**
SL-DSMDPL (our)	9.106	81.9	85.871	92.062	69.727	11.571	68.7	75.862	88.538	80.432	9.726	78.5	83.597	90.805	73.146
MADPL	7.704	95.4	96.005	98.615	75.465	13.607	58.0	65.146	85.382	**83.827**	9.193	86.0	88.819	93.903	78.016
DSMADPL (our)	**6.595**	**96.1**	**96.907**	**99.077**	71.184	**10.718**	**83.7**	**92.188**	**95.847**	80.568	**7.636**	**93.0**	**95.833**	**97.928**	74.230

domain-specific agents' policy are 0.0001 and 0.00001, respectively. The discount factor γ is 0.99, and the target network is updated every $C = 400$ training iterations.

3.3. Evaluation Metrics

In this paper, we use the following 5 metrics to automatically evaluate the performance of dialog agents:
1. Dialog turn: Dialog turn indicates the costs of a dialog agent accomplishing the user's tasks.
2. Match rate: Match rate checks whether the booked entities match all the constraints *C* given by the user.
3. Inform recall: Inform recall measures whether all the user requests *R* are informed by the system.
4. Inform F1: Inform F1 is averaged with inform recall and precision.
5. Success rate: Success rate is the most important metric to automatically evaluate the performance of a dialog agent. We consider the overall dialog tasks to be successful if and only if both inform recall and match rate are 1.

3.4. Experimental Results and Analysis

To evaluate the ability of dialog agents in dealing with multi-domain scenarios, we use a set of 1000 user goals which contains 746 and 252 sample tests for two domains and three domains, respectively. The user agent starts the conversation with the user goal and the system agent interacts with the user to collaboratively complete the user goal. The results of interaction between user agent and system agents with different numbers of domains contained in the goal are presented in Table 1. During the supervised learning stage, compared to the baseline of SL, our domain-specific system agent spends less dialog turns and achieves higher success rates in both two and three domain scenarios: in particular, the success rates exceed the baseline by 6% on average. SL-DSMADPL requires the least dialog turns to complete user goals among all the methods. DSMADPL improves the success rate of the SL pretrained method from 36.4% to 67.2% overall, and achieves the highest success rate among all the methods. Especially in three domain scenarios, DSMADPL exceeds MADPL by approximately 30% of success rate. Among all the methods, DSMADPL achieves the highest inform recall, which shows that our model can better understand the user requests and provide answers. The results in multi-domain scenarios indicate that our proposed model builds a dialog system that can effectively handle complicated tasks.

To evaluate the scalability of dialog agents, we use the benchmark policy in the standardized task-oriented dialog system platform Convlab [25] which is implemented with Agenda [26] to interact with the system agents in all the methods. Table 3 shows the results of interaction between Agenda and the system agents. Our proposed method DSMADPL spends the least dialog turns and achieves the highest success rate, match rate and inform recall in both two and three domains among all the methods, which indicates that our proposed domain-specific dialog agents can implicitly share correct domain information with the user. In three-domain scenarios, DSMADPL exceeds the success rate of the baseline MADPL by about 25%, which highlights the great advantages of DSMADPL in dealing with complex multi-domain tasks.

4. Conclusions and Future Work

In this paper, we proposed a novel approach to build a high-quality multi-domain dialog system based on multi-agent reinforcement learning. To handle the problem of generic agent inability to effectively learn the knowledge in multi-domain ontology, we build several specialized domain-specific agents that only concentrate on their own specific domain knowledge. These agents are first bootstrapped from the dialog corpus, and then *Actor-Critic* reinforcement learning is utilized to train these agents simultaneously. When the system agents interact with the user, the actions from domain-specific agents related to the current topic are together considered as the current turn system actions. In future work, we will focus on how to enable all domain-specific agents to collaborate better.

5. Acknowledgements

This work was supported by the National Key RD Program of China under Grant 2018YFB1305200, the Tianjin Municipal Science and Technology Project under Grant 18ZXZNGX00330 and the National Natural Science Foundation of China under Grant 61771333.

6. References

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

[2] R. Sarikaya, G. E. Hinton, and A. Deoras, "Application of deep belief networks for natural language understanding," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 4, pp. 778–784, 2014.

[3] S. Lee and R. Jha, "Zero-shot adaptive transfer for conversational language understanding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 6642–6649.

[4] M. Henderson, "Machine learning for dialog state tracking: A review," in *Proceedings of The First International Workshop on Machine Learning in Spoken Language Processing*, 2015.

[5] L. Ren, K. Xie, L. Chen, and K. Yu, "Towards universal dialogue state tracking," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 2780–2786.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[7] Z. Lipton, X. Li, J. Gao, L. Li, F. Ahmed, and L. Deng, "Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[8] T.-H. Wen, M. Gašić, N. Mrkšić, P.-H. Su, D. Vandyke, and S. Young, "Semantically conditioned LSTM-based natural language generation for spoken dialogue systems," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015.

[9] S.-Y. Su, K.-L. Lo, Y.-T. Yeh, and Y.-N. Chen, "Natural language generation by hierarchical decoding with linguistic patterns," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Association for Computational Linguistics, Jun. 2018, pp. 61–66.

[10] T. Jaakkola, S. P. Singh, and M. I. Jordan, "Reinforcement learning algorithm for partially observable markov decision problems," *Advances in neural information processing systems*, pp. 345–352, 1995.

[11] G. E. Monahan, "State of the art—a survey of partially observable markov decision processes: theory, models, and algorithms," *Management science*, vol. 28, no. 1, pp. 1–16, 1982.

[12] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," 2011.

[13] B. Peng, X. Li, L. Li, J. Gao, A. Celikyilmaz, S. Lee, and K.-F. Wong, "Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Sep. 2017, pp. 2231–2240.

[14] B. Peng, X. Li, J. Gao, J. Liu, and K.-F. Wong, "Deep Dyna-Q: Integrating planning for task-completion dialogue policy learning," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Jul. 2018, pp. 2182–2192.

[15] L. Chen, R. Yang, C. Chang, Z. Ye, X. Zhou, and K. Yu, "On-line dialogue policy learning with companion teaching," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 2017, pp. 198–204.

[16] P.-H. Su, M. Gasic, N. Mrksic, L. Rojas-Barahona, S. Ultes, D. Vandyke, T.-H. Wen, and S. Young, "On-line active reward learning for policy optimisation in spoken dialogue systems," *arXiv preprint arXiv:1605.07669*, 2016.

[17] 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, pp. 482–489.

[18] A. Papangelis, Y.-C. Wang, P. Molino, and G. Tur, "Collaborative multi-agent dialogue model training via reinforcement learning," in *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*. Stockholm, Sweden: Association for Computational Linguistics, Sep. 2019.

[19] R. Takanobu, R. Liang, and M. Huang, "Multi-agent task-oriented dialog policy learning with role-aware reward decomposition," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020.

[20] S. Ultes, L. M. R. Barahona, P.-H. Su, D. Vandyke, D. Kim, I. Casanueva, P. Budzianowski, N. Mrkšić, T.-H. Wen, M. Gasic *et al.*, "Pydial: A multi-domain statistical dialogue system toolkit," in *Proceedings of ACL 2017, System Demonstrations*, 2017, pp. 73–78.

[21] P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gašić, "MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Oct.-Nov. 2018, pp. 5016–5026.

[22] F. Kreyssig, I. Casanueva, P. Budzianowski, and M. Gašić, "Neural user simulation for corpus-based policy optimisation of spoken dialogue systems," in *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 60–69.

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

[24] M. Kotti, V. Diakoloukas, A. Papangelis, M. Lagoudakis, and Y. Stylianou, "A case study on the importance of belief state representation for dialogue policy management." in *INTERSPEECH*, vol. 986, 2018.

[25] S. Lee, Q. Zhu, R. Takanobu, Z. Zhang, Y. Zhang, X. Li, J. Li, B. Peng, X. Li, M. Huang, and J. Gao, "ConvLab: Multi-domain end-to-end dialog system platform," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Jul. 2019, pp. 64–69.

[26] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young, "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*, 2007, pp. 149–152.