Structured Dialogue Policy with Graph Neural Networks

Lu Chen*, **Bowen Tan***, **Sishan Long and Kai Yu**
Key Lab. of Shanghai Education Commission for Intelligent Interaction and Cognitive Eng.
SpeechLab, Department of Computer Science and Engineering
Brain Science and Technology Research Center
Shanghai Jiao Tong University, Shanghai, China
{chenlusz, tanbowen, longsishan, kai.yu}@sjtu.edu.cn

Abstract

Recently, deep reinforcement learning (DRL) has been used for dialogue policy optimization. However, many DRL-based policies are not sample-efficient. Most recent advances focus on improving DRL optimization algorithms to address this issue. Here, we take an *alternative* route of designing neural network structure that is better suited for DRL-based dialogue management. The proposed *structured deep reinforcement learning* is based on graph neural networks (GNN), which consists of some sub-networks, each one for a node on a directed graph. The graph is defined according to the domain ontology and each node can be considered as a sub-agent. During decision making, these sub-agents have internal message exchange between neighbors on the graph. We also propose an approach to jointly optimize the graph structure as well as the parameters of GNN. Experiments show that structured DRL significantly outperforms previous state-of-the-art approaches in almost all of the 18 tasks of the PyDial benchmark.

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. Dialogue management (DM) is the core of an SDS. It has two missions: one is to maintain the dialogue state, and another is to decide how to respond according to a dialogue policy. In this paper, we focus on the dialogue policy.

A dialogue policy can be viewed simply as a set of hand-crafted mapping rules from dialogue states to dialogue actions. This is referred to as a rule-based policy. However, in real-world scenarios, unpredictable user behavior, inevitable automatic speech recognition, and spoken language understanding errors make it difficult to maintain the true dialogue state and make the decision. Hence, in recent years, there is a research trend towards statistical dialogue management. A well-founded theory for this is the partially observable Markov decision process (POMDP) (Kaelbling et al., 1998).

Under the POMDP-based framework, a distribution of possible states – belief state $\mathbf{b}$ is maintained in every dialogue turn. Then reinforcement learning (RL) methods automatically optimize the policy π, i.e. a mapping function from belief state $\mathbf{b}$ to dialogue action $a = \pi(\mathbf{b})$ (Young et al., 2013). Initially, linear RL-based models are adopted, e.g. least-squares policy iteration (LSPI) and natural actor-critic (NAC). However, these linear models have a poor ability of expression and suffer from slow training. Recently, nonparametric algorithms, e.g. Gaussian process reinforcement learning (GPRL), which can optimize policies from a minimal number of dialogues have been proposed. However, the computation cost of GPRL increases with the increase of the number of data. It is therefore questionable as to whether GPRL can scale to support commercial wide-domain SDS.

More recently, deep reinforcement learning (DRL) methods are adopted for dialogue policies (Cuayáhuitl et al., 2015; Fatemi et al., 2016; Zhao and Eskenazi, 2016; Williams et al., 2017; Chen et al., 2017b; Chang et al., 2017). These policies are often represented by fully connected deep neural networks including deep Q-networks and actor-critic networks. DRL-based models are often more

[*] Both authors contributed equally to this work.

expressive and computational effective. However, these deep models are not sample-efficient and not robust to errors from input modules of SDS. So the focus of these recent advances has been on designing improved RL algorithms to improve the sample-efficiency, e.g. eNAC (Su et al., 2017), BBQ (Lipton et al., 2018).

In this paper, we take an *alternative* but *complementary* approach of focusing primarily on innovating a *structured* neural network architecture that is better suited for dialogue policy. This approach has the benefit that the new network can be easily combined with most existing methods for DRL, which results in *structured deep reinforcement learning*.

The proposed structured DRL is based on graph neural networks (GNN) (Scarselli et al., 2009), which consists of some sub-networks, each one corresponding to a node in the directed graph. The graph has two types of nodes: *slot-dependent* node and *slot-independent* node. The same types of nodes partially share parameters. This can speed up the learning process. In order to model the interaction between sub-networks, they have internal message exchanges between neighbors when doing forward computation.

The main contribution of this paper is three-fold: (1) We proposed a unified *structured* dialogue policy based on GNN. To the best of our knowledge, this is the first work to introduce the graph-based approach for dialogue policy optimization. (2) We introduced a novel method to optimize the graph structure along with the parameters of GNN. (3) Our proposed framework achieves state of the art in PyDial benchmark.

2 Related Work

Recent mainstream studies on statistical dialogue management have been modeling dialogue as a partially observable Markov decision process (POMDP), where reinforcement learning (RL) can be applied to realize automatic dialogue policy optimization (Young et al., 2013).

To achieve efficient policy learning, (Gašić et al., 2010) proposed Gaussian process reinforcement learning (GPRL), where the prior correlations of the objective function given different belief states are defined by the kernel function (Gašić et al., 2010; Gašić and Young, 2014). Recently, deep reinforcement learning (DRL) (Mnih et al., 2015) has been applied in dialogue policy optimization, including value function approximation methods, like deep Q-network (DQN) (Cuayáhuitl et al., 2015; Zhao and Eskenazi, 2016; Fatemi et al., 2016; Chen et al., 2017a; Chang et al., 2017; Peng et al., 2017), and policy gradient methods, e.g. REINFORCE (Williams et al., 2017), advantage actor-critic (A2C) (Fatemi et al., 2016). However, compared with GPRL, most of these models are not sample-efficient. More recently, some methods are proposed to speed up the learning process based on improved DRL algorithms, e.g. eNAC (Su et al., 2017), BBQ (Lipton et al., 2018). In contrast, our proposed method is based on a structured neural architecture, which is complementary to various DRL algorithms.

The most related work is the recently proposed multi-agent dialogue policy (MADP) (Chen et al., 2018). In MADP, the dialogue policy is decomposed into some *slot-dependent* sub-policies and a *slot-independent* sub-policy. While MADP is proposed under the view of multi-agent reinforcement learning, it's can be interpreted as the special instance of our proposed graph-based policy with a fully connected graph structure. That is, our proposed framework is more general. Moreover, the graph structure can be jointly optimized with the parameters of neural networks in our methods.

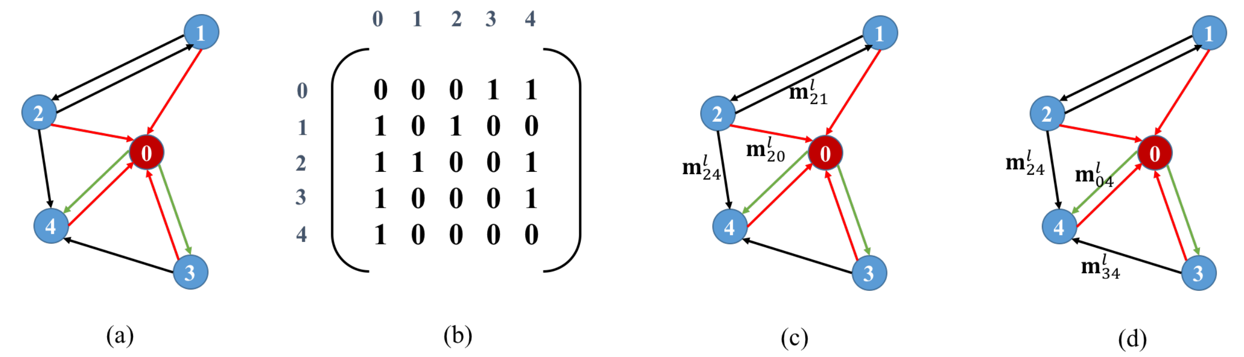
Graph-based approaches are previously adopted for dialogue state tracking, such as the Bayesian update of dialogue state (BUDS) model (Thomson and Young, 2010), structured discriminative model (Lee, 2013). However, they have not been explored for dialogue policy optimization.

3 Proposed Framework

3.1 DRL-based Dialogue Policy

Task-oriented dialogue 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.

At each dialogue turn, the dialogue state tracker maintains a belief state for each informable slot. Generally, the belief state is a distribution of candidate values for the slot. After the update of belief state,



$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 2 & 1 & 1 & 0 & 0 & 1 \\ 3 & 1 & 0 & 0 & 0 & 1 \\ 4 & 1 & 0 & 0 & 0 & 0 \end{array}$$

Figure 1: (a) An example of directed graph G with 5 nodes and 10 edges. There are two types of nodes: nodes 1$\sim$4 (blue) are *slot-dependent* nodes (S-nodes) and node 0 (red) (I-node) is *slot-independent* node. Accordingly, there are 3 type of edges. (b) The adjacency matrix of G. (c) An example of sending messages. S-node 2 sends messages to its out-going neighbors, i.e. S-nodes 1, 4 and I-node 0. (d) An example of aggregating messages. S-node 4 aggregates messages from its in-coming neighbors, i.e. S-nodes 2, 3 and I-node 0.

the values with the largest belief for each informable slot are used as a constraint to search the database. The matched entities in the database as well as the belief state for slots are concatenated as whole belief dialogue state, which is the input of dialogue policy. Therefore, the dialogue state $\mathbf{b}$ usually can be decomposed into some *slot-dependent* states and a *slot-independent* state, i.e. $\mathbf{b} = \mathbf{b}_1 \oplus \cdots \oplus \mathbf{b}_n \oplus \mathbf{b}_0$. $\mathbf{b}_j (1 \leq\ j \leq n)$ is the belief state corresponding to j-th informable slot, and $\mathbf{b}_0$ represents the slot-independent state, e.g. database search results.

The output of dialogue policy is a summary action. Similarly, the summary actions can be divided into $n + 1$ sets including n slot-dependent action sets $\mathbb{A}_j (1 \leq\ j \leq n)$, e.g. *request_slot*$_j$, *confirm_slot*$_j$, *select_slot*$_j$, and one slot-independent action set $\mathbb{A}_0$, e.g. *repeat, inform, reqmore, bye, restart*.

Most of previous DRL-based policies don't take advantage of the structure of belief dialogue state and summary actions. The focus in these recent advances has been on designing improved RL algorithms, e.g. eNAC, BBQ. Here, we take an *alternative* but *complementary* approach of focusing primarily on innovating a *structured* neural network architecture that is better suited for dialogue policy. This approach has the benefit that the new network can be easily combined with existing and future algorithms for RL. That is, this paper advances a new network, but uses already published algorithms. In this paper, we adopt deep-Q-networks (DQN). In the next two sections, we will first give the background of DQN and then introduce the proposed structured policy based on graph neural networks (GNN).

3.2 Deep-Q-Networks (DQN)

A Deep Q-Network (DQN) is a multi-layer neural network which maps a belief state $\mathbf{b}$ to the values of the possible actions a at that state, $Q(\mathbf{b}, a; \theta)$, where θ is the weight vector of the neural network. Neural networks for the approximation of value functions have long been investigated (Lin, 1993). However, these methods were previously quite unstable (Mnih et al., 2013). In DQN, two techniques were proposed to overcome this instability, namely experience replay and the use of a target network (Mnih et al., 2013; Mnih et al., 2015).

At every turn, the transition τ including the previous state $\mathbf{b}$, previous action a, corresponding reward r and current state $\mathbf{b}'$ is put in a finite pool $\mathcal{D}$. Once the pool has reached its maximum size, the oldest transition will be deleted. During training, a mini-batch of transitions is uniformly sampled from the pool, i.e. $\tau \sim \mathrm{U}(D)$. This method removes the instability arising from strong correlation between the subsequent transitions of a dialogue. Additionally, a target network with weight vector $\hat{\theta}$ is used. This target network is similar to the Q-network except that its weights are only copied every K steps from the Q-network, and remain fixed during all the other steps. The loss function for the Q-network at each iteration takes the following form:

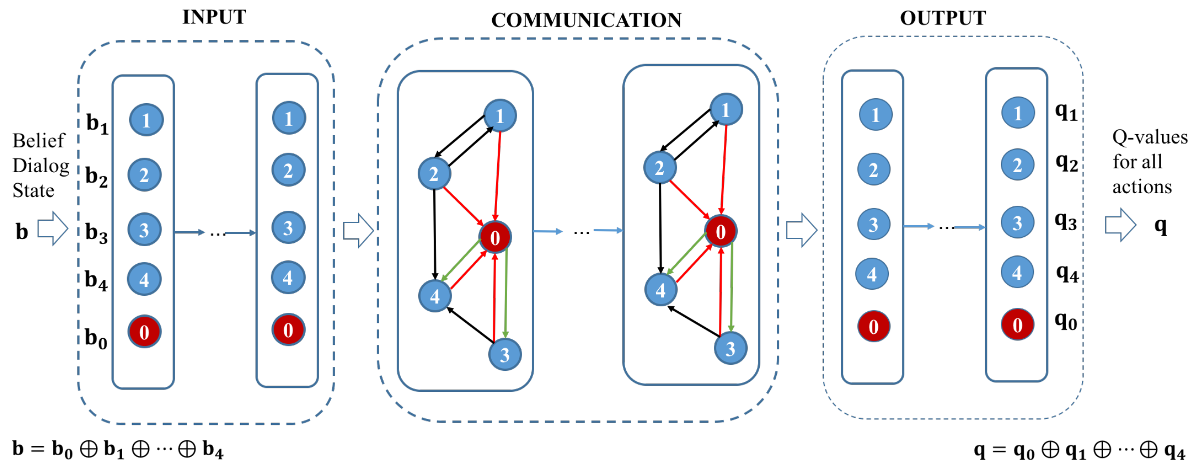


Figure 2: A Q-network with graph neural network (GNN). GNN consists of three modules: input, communication, and output. $\oplus$ denotes concatenation of vectors.

$$\mathcal{L}_{TD}(\theta) = \mathbb{E}_{\tau \sim \mathrm{U}(\mathcal{D})}\left[(y - Q_\theta(\mathbf{b}, a))^2\right], \tag{1}$$

where $y = r + \gamma \max_{a'} Q_{\hat{\theta}}(\mathbf{b}', a')$ and $\gamma \in [0, 1]$ is the discount factor.

Most of the previous work adopts *fully* connected deep neural networks for Q-networks. Here we propose a new network architecture for Q-networks based on *structured* graph neural networks.

3.3 Graph Neural Networks (GNN)

Before delving into details of GNN, we first introduce our notation. We denote the graph structure as $G = (V, E)$ with nodes $v_i (0 \le i \le n) \in V$ and directed edges $e_{ij} \in E$. $\mathcal{N}_{in}(v_i)$ denotes the in-coming neighbors of node v_i, and $\mathcal{N}_{out}(v_i)$ denotes the out-going neighbors of node v_i. $\mathbf{Z}$ is the adjacency matrix of G. The element z_{ij} of $\mathbf{Z}$ is 1 only and only if there is a directed edge from i-th node v_i to j-th node v_j, otherwise z_{ij} is 0. Each node v_i has an associated node type c_i. Similarly, each edge e_{ij} has an edge type u_e, which is determined by node type c_i and node type c_j, i.e. two edges have the same type if and only if their starting node type and their ending node type both are the same.

For dialogue policy, it has two types of nodes: a *slot-independent* node (I-node) and n *slot-dependent* nodes (S-nodes). Accordingly, there are 3 types of edges. Each S-node corresponds to an informable slot in the ontology, while I-node is responsible for slot-independent aspects. Fig.1(a) shows an example of directed graph G with 5 nodes and 10 edges. Nodes 1$\sim$4 (blue) are S-nodes for 4 slots and node 0 (red) is I-node. Fig.1(b) is the corresponding adjacency matrix.

GNN is a deep neural network defined on the graph G. As shown in Fig.3, it consists of three modules: input module, communication module, and output module.

3.3.1 Input Module

At each dialogue turn, each node $v_i (0 \le i \le n)$ will receive the belief dialogue state $\mathbf{b}_i$, which goes through an input module to obtain a state vector $\mathbf{h}_i^0$ as follows:

$$\mathbf{h}_i^0 = f_{c_i}(\mathbf{b}_i), \tag{2}$$

where f_{c_i} is a function for node type c_i, which may be a multilayer perceptron (MLP).

Usually, different slots have different number of candidate values, therefore the dimensions of the inputs of two S-nodes are different. However, in practice, for dialogue policy the probabilities of top K values of each slot play a more important role. Therefore, the whole belief state of each slot is often approximated by the probabilities of *sorted* top K values (Gašić and Young, 2014), which can be implemented by sorted K-max pooling (Kalchbrenner et al., 2014).

3.3.2 Communication Module

The communication module takes $\mathbf{h}_i^0$ as the initial state for node v_i, then update state from one step to the next with following operations.

Sending Messages At l-th step, each node v_i will send messages $\mathbf{m}_{ij}^l$ to node v_j if there is a directed edge from v_i to v_j,

$$\mathbf{m}_{ij}^l = m_{u_e}^l(\mathbf{h}_i^{l-1}), \tag{3}$$

where $m_{u_e}^l$ is a function for edge type u_e at l-th step. For simplicity, we only consider $m_{u_e}^l$ in the form of a linear embedding: $m_{u_e}^l(\mathbf{h}_i^{l-1}) = \mathbf{W}_{u_e}^l \mathbf{h}_i^{l-1}$, where $\mathbf{W}_{u_e}^l$ is a weight matrix to be learned. Fig.1(c) is an example of sending messages for S-node 2. It sends messages to its out-going neighbors, i.e. S-nodes 1, 4 and I-node 0.

Aggregating Messages After sending messages, each node v_j will aggregate messages from its in-coming neighbors,

$$\mathbf{a}_j^l = \sum_{v_i \in \mathcal{N}_{in}(v_j)} \hat{k}_{u_e}^l(\mathbf{h}_i^{l-1}, \mathbf{h}_j^{l-1})\mathbf{m}_{ij}^l = \sum_{1 \le i \le n} z_{ij}\hat{k}_{u_e}^l(\mathbf{h}_i^{l-1}, \mathbf{h}_j^{l-1})\mathbf{m}_{ij}^l, \tag{4}$$

where the function $\hat{k}_{u_e}^l$ computes a normalized scalar representing relationship between node v_i and node v_j. There are various versions of $\hat{k}_{u_e}^l$. Here we introduce two instantiations:

- **Mean-Comm**: All messages from in-coming neighbors are treated equally, i.e. $\hat{k}_{u_e}^l(\mathbf{h}_i^{l-1}, \mathbf{h}_j^{l-1}) = \frac{1}{|\mathcal{N}_{in}(v_j)|}$.
- **Attention-Comm**: In practice, some messages are more important than others. Inspired by *self-attention* model for machine translation (Vaswani et al., 2017), here we first compute the similarity of two states in an unified space, i.e.

$$k_{u_e}^l(\mathbf{h}_i^{l-1}, \mathbf{h}_j^{l-1}) = (\mathbf{W}_{1u_e}^l \mathbf{h}_i^{l-1})^\mathsf{T}(\mathbf{W}_{2u_e}^l \mathbf{h}_j^{l-1}), \tag{5}$$

then normalize it with softmax:

$$\hat{k}_{u_e}^l(\mathbf{h}_i^{l-1}, \mathbf{h}_j^{l-1}) = \frac{e^{(\mathbf{W}_{1u_e}^l \mathbf{h}_i^{l-1})^\mathsf{T}(\mathbf{W}_{2u_e}^l \mathbf{h}_j^{l-1})}}{\sum_i e^{(\mathbf{W}_{1u_e}^l \mathbf{h}_i^{l-1})^\mathsf{T}(\mathbf{W}_{2u_e}^l \mathbf{h}_j^{l-1})}}. \tag{6}$$

Fig.1(d) is an example of aggregating messages for S-node 4. It aggregates messages from its in-coming neighbors, i.e. S-nodes 2, 3 and I-node 0.

Updating State After aggregating messages from neighbors, every node v_i will update its state from $\mathbf{h}_i^{l-1}$ to $\mathbf{h}_i^l$,

$$\mathbf{h}_i^l = g_{c_i}^l(\mathbf{h}_i^{l-1}, \mathbf{a}_i^l), \tag{7}$$

where $g_{c_i}^l$ is the update function for node type c_i at l-th step, which may be a non-linear layer in practice, i.e.

$$\mathbf{h}_i^l = \sigma(\mathbf{W}_{c_i}^l \mathbf{h}_i^{l-1} + \mathbf{a}_i^l), \tag{8}$$

where σ is an activation function, e.g. RELU, and $\mathbf{W}_{c_i}^l$ is the transition matrix.

3.3.3 Output Module

After updating state L steps, based on the last state $\mathbf{h}_i^L$ each node v_i will compute the Q-values $\mathbf{q}_i$ for the actions corresponding to v_i:

$$\mathbf{q}_i = o_i(\mathbf{h}_i^L), \tag{9}$$

where o_i is a function for node i, which may be a MLP in practice. Note that, in input module and communication module, the same types of nodes share parameters, which may speed up the learning process. However, in output module, in oder to capture the *specific characteristics* of each node, they don't share parameters.

When making decision, the outputs of all nodes are first concatenated, i.e. $\mathbf{q} = \mathbf{q}_0 \oplus \mathbf{q}_1 \oplus \cdots \oplus \mathbf{q}_n$, then the action is chosen according to $\mathbf{q}$ as done in vanilla DQN.

3.4 Inference of Graph Structure

In the previous section, we assume that the structure of graph G, i.e. the adjacency matrix $\mathbf{Z}$, is known. However, usually the graph is not known in practice, and the hypothetical structure is not guaranteed to be optimal. Therefore, it's better to optimize the structure along with the parameters of GNN. We assume that $\mathbf{Z}$ is a *latent* variable, and follows a factorized Bernoulli distribution, i.e. each element $z_{ij} \sim \mathrm{Bern}(\phi_{ij})$. The exact posterior distribution of $\mathbf{Z}$ is hard to inference and we can obtain the approximate posterior $q_\phi(\mathbf{Z})$ via variational inference. The loss function in equation (1) will be reformulated as follows:

$$\begin{aligned}\mathcal{L}(\theta,\phi) =& \mathbb{E}_{\tau\sim \mathrm{U}(\mathcal{D}),\mathbf{Z}\sim q_\phi(\mathbf{Z})}\left[(y - Q_\theta(\mathbf{b},a;\mathbf{Z}))^2\right] + \lambda \mathrm{KL}(q_\phi(\mathbf{Z})||p(\mathbf{Z})) \\ =& \mathcal{L}_E(\theta,\phi) + \lambda\mathcal{L}_C(\phi),\end{aligned} \tag{10}$$

where $p(\mathbf{Z})$ is the prior, and also follows a factorized Bernoulli distribution.

As shown in the equation, the loss function $\mathcal{L}(\theta,\phi)$ consists of two terms $\mathcal{L}_E(\theta,\phi)$ and $\mathcal{L}_C(\phi)$. $\mathcal{L}_E(\theta,\phi)$ corresponds to the error loss that measures how well the model is fitting the current dataset, whereas $\mathcal{L}_C(\phi)$ refers to the complexity loss that measures the flexibility of the model. For a uniform Bernoulli prior distribution, $\mathcal{L}_C(\phi)$ can be written following:

$$\mathcal{L}_C(\phi) = \sum_{i,j}\mathrm{KL}(q_{\phi_{ij}}(z_{ij})||p(z_{ij}))) = \sum_{i,j}\left[-\mathrm{log}0.5 + \phi_{ij}\mathrm{log}\phi_{ij} + (1-\phi_{ij})\mathrm{log}(1-\phi_{ij})\right]. \tag{11}$$

Minimising the KL divergence between $q_{\phi_{ij}}(z_{ij})$ and the prior is equivalent to maximising the entropy of the Bernoulli random variable with probability ϕ_{ij}. This pushes the probability towards 0.5.

While $\mathcal{L}_C(\phi)$ is straightforward to minimize, $\mathcal{L}_E(\theta,\phi)$ does not allow for efficient gradient based optimization due to the discrete nature of $\mathbf{Z}$. To obtain the Q-values $Q_\theta(\mathbf{b},a;\mathbf{Z})$, we need first to sample the discrete graph structure $\mathbf{Z}$ according to the factorized Bernoulli distribution with parameters ϕ. Therefore, the loss function $\mathcal{L}_E(\theta,\phi)$ is non-differential w.r.t. ϕ. While in principle a score function estimator such as the REINFORCE (Williams, 1992) could be employed, it suffers from high variance in practice. An alternative is to replace the discrete Bernoulli distribution with its continuous relaxation, i.e. the Concrete distribution as done at Concrete dropout (Gal et al., 2017). Instead of sampling the random variable from the discrete Bernoulli distribution we sample realisations from the following Concrete distribution with some temperature t:

$$\tilde{z}_{ij} = \mathrm{sigmoid}\left(\frac{1}{t}\left(\mathrm{log}\phi_{ij} - \mathrm{log}(1-\phi_{ij}) + \mathrm{log}\epsilon_{ij} - \mathrm{log}(1-\epsilon_{ij})\right)\right), \tag{12}$$

where ϵ_{ij} is the noise sampled from a uniform distribution, i.e. $\epsilon_{ij} \sim \mathrm{U}(0,1)$. The Concrete distribution concentrates most mass on the boundaries of the interval 0 and 1. With the continue relaxation $\tilde{z}_{ij}$ of the Bernoulli random variable z_{ij}, the loss function will be reparameterized as follows:

$$\mathcal{L}(\theta,\phi) = \mathbb{E}_{\tau\sim \mathrm{U}(\mathcal{D}),\epsilon\sim \mathrm{U}(0,1)}\left[\left(y - Q_\theta(\mathbf{b},a;\tilde{\mathbf{Z}})\right)^2\right] + \lambda KL(q_\phi(\mathbf{Z})||p(\mathbf{Z})), \tag{13}$$

where $\epsilon \sim \mathrm{U}(0,1)$ denotes that every ϵ_{ij} is sampled from the uniform distribution independently. Every element z_{ij} of $\tilde{\mathbf{Z}}$ is calculated with equation (12). Note that whereas the sampling still exists, the sampled noise ϵ is independent of the parameters ϕ. Therefore the loss function $\mathcal{L}(\theta,\phi)$ is differential w.r.t. ϕ and it can be straightforward to minimize.

Note that the procedure of sampling according to equation (12) is approximately equivalent to sampling the neighbors of every node v_i according to the probability ϕ_{ij}. It is a variant of dropout and adopted in self-attention models (Tan et al., 2018) and graph convolutional networks (Kipf and Welling, 2017).

task	Env.1	Env.2	Env.3	Env.4	Env.5	Env.6
SER	0%	0%	15%	15%	15%	30%
Masks	On	Off	On	Off	On	On
User	Standard	Standard	Standard	Standard	Unfriendly	Standard

Table 1: The set of benchmarketing tasks

4 Experiments

4.1 PyDial Benchmark

Dialogue management research is typically evaluated on a small set of environments. Fortunately, a set of extensive simulated dialogue management environments is published in (Casanueva et al., 2017), which can test the capability of models in different environments. These environments are implemented in an open domain toolkit: PyDial. With providing domain-independent implementations of all the dialogue system modules, simulated users and simulated error models, PyDial has the potential to create a set of benchmark environments to compare different models in the same conditions. In this paper, we evaluate our proposed approaches on these environments, which will be briefly introduced next and are summarized in Table 1.

Firstly, there are three different **domains**: information seeking tasks for restaurants in Cambridge (CR) and San Francisco (SFR) and a generic shopping task for laptops (LAP). They are slot-based, which means the dialogue state is factorized into slots.

The second different dimension of variability is the **semantic error rate (SER)**, which simulates different noise levels in the speech understanding input channel.

In addition, env.5's **user model** is defined to be an *Unfriendly* distribution, where the users barely provide any extra information to the system, while others' are all *Standard*.

Finally, in order to test the learning capability of the models, the action **masking mechanism** is disabled in two of the tasks: env2 and env4.

The metrics in the next section has presented the average success rate and average reward for each applied model. The success rate is defined as the percentage of dialogues which are completed successfully.The reward is defined as $20 \times \mathbb{1}(\mathcal{D}) - T$, here $\mathbb{1}(\mathcal{D})$ is the success indicator and T is the dialogue length in turns.

4.2 Results

We evaluate four variants of our proposed structured DRL-based dialogue policy:

- **GNN-M**: GNN-based dialogue policy with *fully* connected graph. The communication method between nodes is *Mean-Comm* described in section 3.3.2.

- **GNN-M-C**: It is similar to GNN-M except that the graph structure is jointly optimized with the parameters of GNN as described in section 3.4. The hyperparameter λ in equation (13) is 4×10^{-4}.

- **GNN-A**: GNN-based dialogue policy with *fully* connected graph. The communication method between nodes is *Attention-Comm* described in section 3.3.2.

- **GNN-A-C**: It is similar to GNN-A except that the graph structure is jointly optimized with the parameters of GNN as described in section 3.4. The hyperparameter λ in equation (13) is 4×10^{-4}.

These models are compared with three baselines[1] presented in (Casanueva et al., 2017): GP-Sarsa, DQN, and eNAC. The results in the 18 tasks after 1000/4000 training dialogues are shown in Table 2. For each task, the result is the mean over 10 different random seeds.

[1]Due to the limitation of space, the results of A2C is omitted. The performance of A2C is worst among these baselines.

		Baselines						Structured DRL							
		GP-Sarsa		DQN		eNAC		GNN-M		GNN-A		GNN-M-C		GNN-A-C	
	Task	Suc.	Rew.	Suc.	Rew.	Suc.	Rew.	Suc.	Rew.	Suc.	Rew.	Suc.	Rew.	Suc.	Rew.
		after 1000 training dialogues													
Env.1	CR	98.0	13.0	88.6	11.6	93.0	12.2	95.2	13.0	97.7	13.5	96.3	13.0	***98.8***	**13.7**
	SFR	91.9	10.0	48.0	2.7	85.8	9.9	70.1	8.6	66.6	6.3	**89.0**	**10.2**	85.8	9.6
	LAP	78.9	6.7	61.9	5.5	84.2	8.8	71.0	7.2	79.0	8.7	**92.3**	**10.8**	87.0	9.8
Env.2	CR	91.1	10.8	67.6	6.4	78.8	6.6	94.0	12.4	94.7	12.3	**95.8**	**12.6**	94.5	12.0
	SFR	82.1	7.4	64.2	5.4	67.5	2.7	**85.4**	**10.9**	76.2	7.5	83.8	8.4	77.8	6.7
	LAP	68.4	3.1	70.8	5.8	57.8	-0.5	78.2	8.0	80.7	**8.5**	83.4	6.6	**83.4**	7.1
Env.3	CR	91.9	10.4	79.5	9.2	85.7	10.0	88.4	10.8	**97.1**	**12.5**	95.9	12.2	95.8	12.2
	SFR	76.6	5.5	42.4	1.0	73.6	6.2	78.6	7.5	80.4	**7.8**	80.9	7.1	**81.5**	7.0
	LAP	65.0	2.8	51.9	3.1	71.0	5.5	68.2	5.5	71.1	6.1	**80.0**	**7.1**	79.2	6.2
Env.4	CR	88.2	9.3	73.5	6.9	73.6	4.4	**91.3**	**10.8**	88.9	10.2	91.2	10.4	86.6	9.3
	SFR	73.6	4.9	65.9	4.5	60.4	0.8	**81.3**	**8.2**	79.0	7.4	71.5	3.9	80.9	7.1
	LAP	61.3	0.3	53.2	2.7	46.9	-2.9	66.1	**4.5**	65.8	4.1	65.8	2.1	**67.6**	1.7
Env.5	CR	90.2	9.0	60.1	4.1	81.2	8.1	94.8	**11.2**	95.1	11.1	92.9	10.3	**95.2**	11.0
	SFR	65.3	1.3	32.5	-2.0	54.0	0.9	53.8	1.3	74.3	**5.0**	74.5	3.6	**77.6**	4.3
	LAP	44.9	-2.8	31.4	-1.8	61.3	1.7	51.4	1.3	67.1	2.8	70.7	2.7	**76.6**	**3.1**
Env.6	CR	84.9	8.3	72.3	6.9	73.6	6.7	87.7	9.9	**88.8**	**10.1**	86.9	9.0	87.3	9.2
	SFR	59.7	0.7	35.6	-1.2	55.2	1.4	51.2	1.4	65.6	**3.6**	67.2	2.8	**71.1**	3.1
	LAP	52.0	-1.5	47.5	1.4	56.3	1.9	57.0	2.7	60.1	2.8	**71.4**	**3.3**	65.4	1.6
Mean	CR	90.7	10.1	73.6	7.5	81.0	8.0	91.9	11.4	**93.7**	**11.6**	93.2	11.3	93.0	11.2
	SFR	74.9	5.0	48.1	1.7	66.1	3.6	70.1	6.3	73.7	6.3	77.8	6.0	**79.1**	**6.3**
	LAP	61.7	1.4	52.8	2.8	62.9	2.4	65.3	4.9	70.7	**5.5**	**77.3**	5.4	76.5	4.9
	ALL	75.8	5.5	58.2	4.0	70.0	4.7	75.8	7.5	79.4	**7.8**	82.8	7.6	**82.9**	7.5
		after 4000 training dialogues													
Env.1	CR	**99.4**	13.5	93.9	12.7	94.8	12.4	96.1	13.3	99.2	**14.0**	98.6	13.7	99.1	13.9
	SFR	**96.1**	11.4	65.0	5.9	94.0	**11.7**	86.8	10.2	88.7	10.7	93.5	10.8	94.5	11.4
	LAP	89.1	9.4	70.1	6.9	91.4	10.5	85.4	9.9	86.6	9.8	92.2	10.6	**93.7**	**10.8**
Env.2	CR	96.8	12.2	91.9	12.0	83.6	9.0	**98.5**	**13.6**	96.4	12.7	96.6	12.8	90.5	11.3
	SFR	91.9	9.6	84.3	9.2	65.6	3.7	**93.6**	**11.6**	87.4	9.5	91.8	10.3	89.2	10.3
	LAP	82.3	7.3	74.5	6.6	55.1	1.5	83.3	8.8	**92.2**	**11.3**	89.1	9.0	83.9	7.5
Env.3	CR	95.1	11.0	93.4	11.9	90.8	11.2	**96.4**	**12.6**	95.3	12.1	95.6	12.1	96.3	12.3
	SFR	81.6	6.9	60.9	4.0	84.6	8.6	84.1	8.3	84.2	**8.5**	**88.2**	8.2	83.0	7.1
	LAP	68.3	4.5	61.1	4.3	76.6	6.7	77.7	6.9	78.5	**7.0**	**85.6**	6.8	79.6	5.6
Env.4	CR	91.5	9.9	90.0	10.7	85.3	9.0	93.5	**11.5**	**93.9**	11.4	93.4	11.0	92.0	10.9
	SFR	81.6	7.2	77.8	7.7	61.7	2.0	77.7	7.2	79.8	7.5	80.9	7.2	**85.8**	**8.4**
	LAP	72.7	5.3	68.7	5.5	52.8	-0.8	**85.3**	**8.8**	77.5	7.7	72.9	3.5	75.6	4.0
Env.5	CR	93.8	9.8	90.7	10.3	91.6	10.5	**95.1**	**11.2**	94.7	10.9	95.0	10.8	93.4	10.2
	SFR	74.7	3.6	62.8	2.9	74.4	4.5	**88.8**	**7.9**	81.6	6.3	83.9	5.4	79.9	4.3
	LAP	39.5	-1.6	45.5	0.0	75.8	4.1	73.0	**4.2**	64.7	1.7	75.2	1.5	**76.9**	2.6
Env.6	CR	89.6	8.8	87.8	10.0	79.6	8.0	89.0	**10.0**	89.3	9.8	**90.4**	9.9	89.4	9.8
	SFR	64.2	2.7	47.2	0.4	66.7	3.9	72.8	**4.9**	69.3	4.0	67.3	1.7	**74.4**	3.7
	LAP	44.9	-0.2	46.1	1.0	64.6	3.6	64.1	3.6	69.4	**4.1**	**72.0**	1.8	69.8	2.0
Mean	CR	94.4	10.9	91.3	11.3	87.6	10.0	94.8	**12.0**	94.8	11.8	**94.9**	11.7	93.5	11.4
	SFR	81.7	6.9	66.3	5.0	74.5	5.7	84.0	**8.4**	81.8	7.8	84.3	7.3	**84.5**	7.5
	LAP	66.1	4.1	61.0	4.1	69.4	4.3	78.1	**7.0**	78.2	6.9	**81.2**	5.5	79.9	5.4
	ALL	80.7	7.3	72.9	6.8	77.2	6.7	85.6	**9.1**	84.9	8.8	**86.8**	8.2	86.0	8.1

Table 2: Reward and success rates after 1000/4000 training dialogues. The results in bold blue are the best success rates, and the results in bold black are the best rewards.

We can find that our models can achieve the best performance in almost all of the tasks. In the two larger domains (SFR, LAP), the performance boost is more clear. For our four models, there is no significant difference in their performance, which indicates our proposed framework is not sensitive to these variants. However, through further analysis of these results, some interesting phenomena are observed.

After 1000 training dialogues, on average, two models (GNN-A and GNN-A-C) with *Attention-Comm* achieve the best reward and the best success rate respectively. It shows that *Attention-Comm* may benefit the sample-efficiency in the early learning stage. The potential reason is that models with *Attention-Comm* can quickly adjust their parameters to pay more attention on most important messages.

It is interesting to observe that two models with optimization of graph structure obtain better success rate than the other two (GNN-M-C vs. GNN-M, GNN-A-C vs. GNN-A). As described in section 3.4, our proposed approach for structure optimization is a variant of dropout, i.e dropout neighbors. Dropout introduces some noise in the neural networks and can improve exploration in DRL (Gal and Ghahramani, 2016; Gal et al., 2017). Here, GNN-A-C and GNN-M-C also do more exploration, resulting in long dialogues which have more potential for success.

4.3 Analysis and Discussion

Turn0: Sys: hello()

Turn1: User: hello(type=‘restaurant’)

Sys: request(area) [action from node 1]

Turn2: User: inform(area=‘dontcare’)

Sys: request(pricerange) [action from node 3]

Turn3: User: inform(pricerange="expensive", food="gastropub")

Sys: inform(food="gastropub", pricerange="expensive", name="royal standard", area="east") [action from node 0]

Turn4: User: request(addr)

Sys: inform(name="royal standard",addr="290 mill road city centre") [action from node 0]

Turn5: User: bye()

Node0: I-node
Node1: S-node area
Node2: S-node food
Node3: S-node pricerange

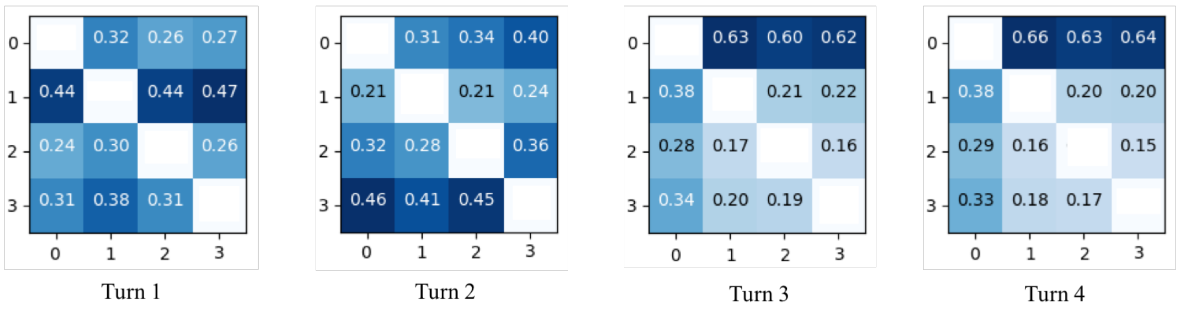


Figure 3: An example of dialogue on semantic level with a policy of GNN-A in CR domain (Env.1).

Fig.3 is an example of dialogue with a policy of GNN-A in CR domain (Env.1). The confusion matrices show the attention weights in the first communication step computed by equation (6). The elements in the i-th row represent the weights node v_i sending messages to other nodes. And each column is normalized by *softmax*. We can find that if the final action comes from the action sets $\mathbb{A}_i$, which is corresponding to node v_i, then the weights in the i-th row will be larger, as if to say:“ *Hey, it’s my duty to making decision at this turn, and you should make the Q-values for your action sets smaller!*”.

The confusion matrix in Fig.4 in Appendix A is the optimized graph structure in LAP domain (Env.3), i.e. the parameter ϕ_{ij} represents the probability of the existence of edge from node v_i to node v_j. We can find that all probabilities are in the range [0.7,0.9]. However, the probabilities associated with I-node is much larger than others, which means the communication between I-node and S-node is more important than the communication between S-node and S-node.

5 Conclusion

In this paper, we propose a structured dialogue policy, which is represented by a graph neural network (GNN). It consists of some sub-networks, each one for a node in a directed graph. The graph has two types of nodes: a *slot-independent* (I-node) node and n *slot-dependent* nodes (S-nodes). Each S-node corresponds to a slot. The same types of nodes partially share parameters. In order to model the interaction between sub-networks, they have internal message exchanges between neighbors in the graph when doing forward computation. We evaluate our framework on PyDial benchmark. Our models achieve state of the art in almost all of 18 tasks.

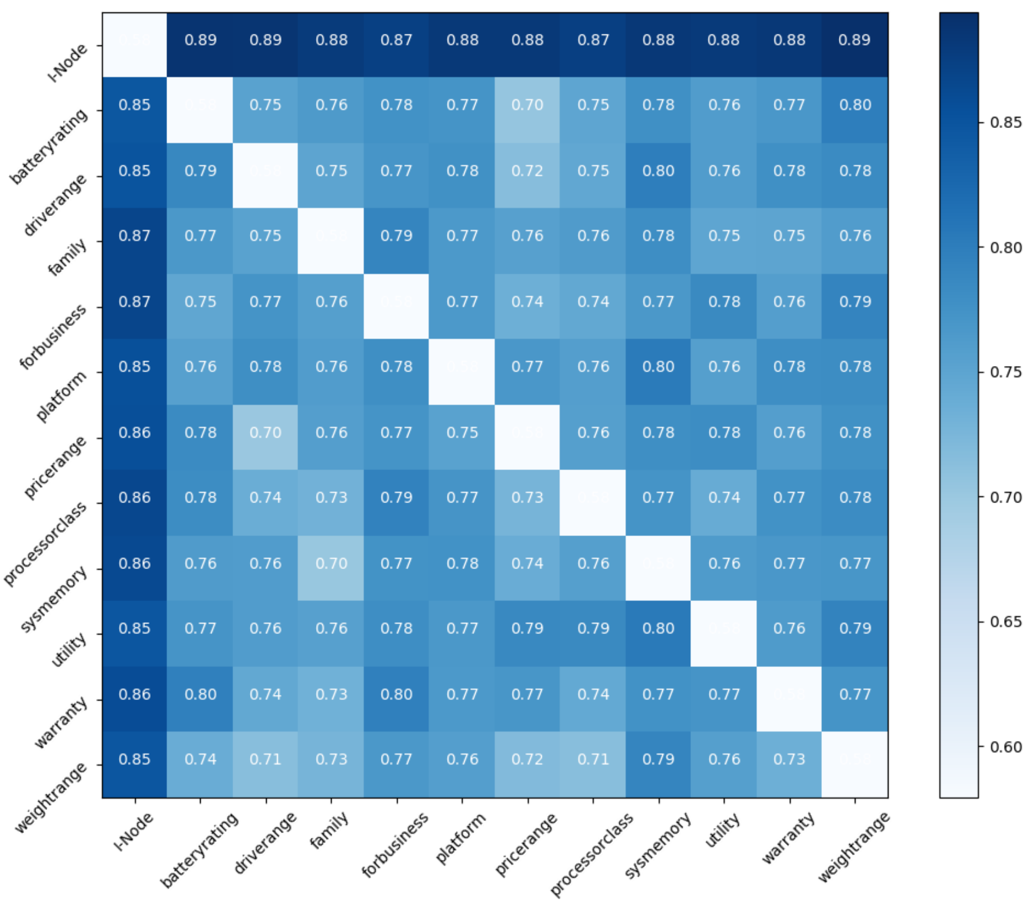


Figure 4: The confusion matrix of graph structure in LAP domain (Env.3). Each element denotes the probability that there is a edge from node v_i to v_j.

Acknowledgements

The corresponding author is Kai Yu. This work has been supported by the National Key Research and Development Program of China under Grant No.2017YFB1002102, Shanghai International Science and Technology Cooperation Fund (No. 16550720300) and the Major Program of Science and Technology Commission of Shanghai Municipality (STCSM) (No. 17JC1404104). Experiments have been carried out on the PI supercomputer at Shanghai Jiao Tong University.

References

Iñigo Casanueva, Paweł Budzianowski, Pei-Hao Su, Nikola Mrkšić, Tsung-Hsien Wen, Stefan Ultes, Lina Rojas-Barahona, Steve Young, and Milica Gašić. 2017. A benchmarking environment for reinforcement learning based task oriented dialogue management. *arXiv preprint arXiv:1711.11023*.

Cheng Chang, Runzhe Yang, Lu Chen, Xiang Zhou, and Kai Yu. 2017. Affordable on-line dialogue policy learning. In *Proceedings of EMNLP*, pages 2190–2199.

Lu Chen, Runzhe Yang, Cheng Chang, Zihao Ye, Xiang Zhou, and Kai Yu. 2017a. On-line dialogue policy learning with companion teaching. *Proceedings of EACL*, pages 198–204.

Lu Chen, Xiang Zhou, Cheng Chang, Runzhe Yang, and Kai Yu. 2017b. Agent-aware dropout dqn for safe and efficient on-line dialogue policy learning. In *Proceedings of EMNLP*, pages 2444–2454.

Lu Chen, Cheng Chang, Zhi Chen, Bowen Tan, Milica Gašić, and Kai Yu. 2018. Policy adaptation for deep reinforcement learning-based dialogue management. In *Proceedings of ICASSP*, pages 6074–6078.

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

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

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of ICML*, pages 1050–1059.

Yarin Gal, Jiri Hron, and Alex Kendall. 2017. Concrete dropout. In *Proceedings of NIPS*, pages 3584–3593.

Milica Gašić, Filip Jurčíček, Simon Keizer, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. 2010. Gaussian processes for fast policy optimisation of POMDP-based dialogue managers. In *Proceedings of SIGDIAL*, pages 201–204.

Milica Gašić and Steve Young. 2014. Gaussian processes for pomdp-based dialogue manager optimization. *IEEE/ACM TASLP*, 22(1):28–40.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*, pages 655–665.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Sungjin Lee. 2013. Structured discriminative model for dialog state tracking. In *Proceedings of SIGDIAL*, pages 442–451.

Long-Ji Lin. 1993. *Reinforcement learning for robots using neural networks*. Ph.D. thesis, Fujitsu Laboratories Ltd.

Zachary C. Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. 2018. Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking. In *Proceedings of AAAI*, pages 5237–5244.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

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. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of EMNLP*, pages 2231–2240.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Pei-Hao Su, Pawel Budzianowski, Stefan Ultes, Milica Gasic, and Steve Young. 2017. Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. In *Proceedings of SIGDIAL*, pages 147–157.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proceedings of AAAI*, pages 4929–4936.

Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*, pages 6000–6010.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of ACL*, pages 665–677.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *Proceedings of SIGDIAL*, pages 1–10.