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-Agents) 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.

Fig. 1. An example of task-oriented dialogue.
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-Agents) 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 transfer such 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 \( b \) usually can be decomposed into sub-states \( b_1, \cdots, b_n \) and \( b_g \), i.e. \( b = b_1 \oplus \cdots \oplus b_n \oplus b_g \). \( b_j(1 \leq j \leq n) \) is the belief state corresponding to \( j \)-th informable slot, and \( b_g \) 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 \( A_j(1 \leq j \leq n) \), e.g. request_slot, confirm_slot, select_slot, and one slot-independent action set \( A_g \), e.g. repeat, offer.

MADP consists of \( n \) slot-dependent agents, i.e. S-Agents \( A_j(1 \leq j \leq n) \), each one corresponding to an informable slot, and a slot-independent agent, i.e. G-Agent \( A_g \). Figure 2(a) provides an overview of MADP with DQN as the DRL algorithm.

- The input of \( A_j \) is \( b_j \), and the output is the Q-values \( q_j \) corresponding to actions in \( A_j \), i.e. \( q_j = [Q(b_j, a_{j1}), \cdots, Q(b_j, a_{jm_j})] \), where \( a_{jk}(1 \leq k \leq m_j) \) is the belief state corresponding to \( j \)-th informable slot in \( A_j \).

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).

To obtain the Q-values \( q \) for all actions, the outputs of all agents are concatenated as shown in Figure 2(a), i.e. \( q = q_1 \oplus \cdots \oplus q_n \oplus q_g \). When making decision, the action is chosen according to \( q \).

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 \( A_j \) and \( A_g \) will output some messages. Here, we just use the output of hidden layer as messages, i.e. \( h_j \) for \( A_j \) and \( h_g \) for \( A_g \). At \( (i+1) \)-th hidden layer, the input of \( A_j \) includes the output of its previous layer \( h_j \), the message from other S-Agents \( c_j \), and the message from G-Agent \( g^i \),

\[
c^i_j = \frac{1}{n-1} \sum_{1 \leq l < n, l \neq j} h^i_l,
\]

and the message from G-Agent \( g^i \), \( g^i = h_g \). Based on \( h_j, c_j \) and \( g^i \), the output of \( (i+1) \)-th hidden layer of \( A_j \) is shown at the top of Figure 2(c), i.e.

\[
h^i_{j+1} = \sigma(\mathbf{H}_j^i h^i_j + \mathbf{C}_j^i c^i_j + \mathbf{G}_g^i g^i),
\]

where \( \sigma \) is a non-linear activation function, e.g. RELU. \( \theta_g = \{H_g^i, C_g^i, G_g^i\}_{i=1}^l \) are weight matrices, i.e. parameters shared across all slot-dependent agents.

Similarly, at \( (i+1) \)-th hidden layer of \( A_g \), the input includes the output of its previous layer \( h_g^i \) and the message from S-Agents \( c_g^i \),

\[
c^i_g = \frac{1}{n} \sum_{1 \leq j \leq n} h^i_j.
\]

\(^1\)For simplicity, the bias term is omitted.
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 $A_j$ has its own private parameters $\theta_j \triangleq \{H^i_j, C^i_j, G^i_j\}_{i=1}^L$ in addition to the shared parameters $\theta_s \triangleq \{H^i, C^i, G^i\}_{i=1}^L$ across all slots. For each input $b_j$, the agent first computes the outputs with $\theta_j$ and $\theta_s$ in parallel, then takes the weighted average of two outputs to obtain the final output $q_j$, i.e.

$$q_j = \alpha \text{Net}(b_j; \theta_j) + (1 - \alpha) \text{Net}(b_j; \theta_s),$$

where $\alpha \in [0, 1]$ is the weight. The more complex the domain/task, the larger $\alpha$ 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 $\theta_s$, the private parameters $\theta_j$ for S-Agent $A_j$ and the parameters $\theta_g$ for G-Agent $A_g$. (2) Train the multi-agent policy in original domain. (3) When domain is extended, the private parameters for new S-Agent $\theta_j$ are initialized by $\theta_s$, or when it’s transferred to a new domain, the shared parameters $\theta'_s$ for S-Agents and the $\theta'_g$ 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 $\theta'_j$ are initialized by $\theta_s$ with added noise, i.e. $\theta'_j \leftarrow \theta_s + \mathcal{N}(0, \sigma_{\text{noise}} \cdot \mathbf{1})$. (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 price range slot. DSTC3 is touristic 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.](image)
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

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 faster than dqn_adapt when they are transferred from DSTC2_Simple to DSTC2. The newly added S-Agent for new slot price_range 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


