Scaling Up Deep Reinforcement Learning for Multi-Domain Dialogue Systems

Heriberto Cuayáhuitl
School of Computer Science
University of Lincoln
Lincoln, United Kingdom
Email: HCuayahuitl@lincoln.ac.uk

Seunghak Yu
Artificial Intelligence Team
Samsung Electronics Co. Ltd.
Seoul, South Korea
Email: seunghak.yu@samsung.com

Ashley Williamson, Jacob Carse
School of Computer Science
University of Lincoln
Lincoln, United Kingdom
Email: {awilliamson,jcarse}@lincoln.ac.uk

***Abstract*—Standard deep reinforcement learning methods such as Deep Q-Networks (DQN) for multiple tasks (domains) face scalability problems due to large search spaces. This paper proposes a three-stage method for multi-domain dialogue policy learning—termed NDQN, and applies it to an information-seeking spoken dialogue system in the domains of restaurants and hotels. In this method, the first stage does multi-policy learning via a network of DQN agents; the second makes use of compact state representations by compressing raw inputs; and the third stage applies a pre-training phase for bootstraping the behaviour of agents in the network. Experimental results comparing DQN (baseline) versus NDQN (proposed) using simulations report that the proposed method exhibits better scalability and is promising for optimising the behaviour of multi-domain dialogue systems. An additional evaluation reports that the NDQN agents outperformed a K-Nearest Neighbour baseline in task success and dialogue length, yielding more efficient and successful dialogues.**

I. Introduction

Neural-based dialogue systems are playing an important role in the research and development of artificially intelligent systems. Agents based on the Reinforcement Learning (RL) paradigm offer the possibility to treat dialogue design as an optimisation problem, and are attractive because they can improve their performance over time with experience. But the application of RL is not trivial due to the complexity of the problem such as large state-action spaces exhibited in human-machine conversations. This is especially true in multi-domain systems, where the number of state variables (features) and dialogue actions increases rapidly as more context and domains are taken into account. On the one hand, unique situations in the interaction can be described by a large number of variables (e.g. words raised in the conversation by the system and user) so that enumerating them would result in very large state spaces. On the other hand, the action space can also be large due to the wide range of unique dialogue actions (e.g. requests, apologies, confirmations in multiple contexts).

While one can aim for optimising the interaction via compression of the search space, it is usually unclear what features to incorporate in the state representation. This is a strong motivation for applying Deep Reinforcement Learning (DRL) to dialogue management so that the agent can simultaneously learn its feature representation and policy [1]. This paper makes use of raw noisy text as features in an attempt to avoid engineered features to represent the dialogue state. By using this representation, dialogue agents bypass spoken language understanding in order to learn dialogue policies directly from raw (noisy) text to actions [2].

We address dialogue policy learning using the divide-and-conquer approach, in which dialogue states can be described at different levels of granularity, and an action can be executed using either a single dialogue action (taking one dialogue turn) or a composite one (equivalent to a subdialogue taking multiple dialogue turns). This approach offers at least two benefits. First, modularity helps to optimise subdialogues that may be easier to optimise than the whole dialogue. Second, subdialogues may include only relevant dialogue knowledge in the states and relevant actions, thus reducing significantly the size of possible solutions: consequently they can be found faster. These properties are crucial for training the behaviour of multi-domain spoken dialogue systems in which there may be a large set of state variables or a large number of actions.

The remainder of this paper describes a novel data-driven method applied to an information-seeking dialogue system in the domains of restaurants and hotels. Experimental results show that the proposed method can train policies faster than previous work [3] and more effectively than standard and baseline algorithms in the literature, showing promise for training conversational neural-based agents in multiple domains.

II. Background

Raw features in human-machine conversations such as words with confidence scores can be given as input to a reinforcement learning agent to induce dialogue policies from interaction with the environment, where situations (words) are mapped to actions (dialogue acts) by maximizing a long-term reward signal [2]. An RL agent is typically characterized by: (i) a finite set of states $S = \{s_1, ..., s_n\}$; (ii) a finite set of actions $A = \{a_1, ..., a_m\}$; (iii) a state transition function $T(s, a, s')$ that specifies the next state s' given the current state s and action a; (iv) a reward function $R(s, a, s')$ that specifies the reward given to the agent for choosing action a when the environment makes a transition from state s to state s'; and (v) a policy $\pi : S \rightarrow A$ that defines a mapping from states to actions. The goal of an RL agent is to find an optimal policy

(π^*) by maximising its cumulative discounted reward defined as

$$Q^*(s,a) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... | s_t = s, a_t = a, \pi],$$

where function Q^* represents the maximum sum of rewards r at time t discounted by factor γ at each time step. An RL agent takes actions with probability $Pr(a|s)$ during training, and the best at test time according to $\pi^*(s) = \arg\max_{a \in A} Q^*(s,a)$.

To induce the Q function above our agent approximates Q^* using a multilayer neural network as in [4]. The Q function is parameterised as $Q(s,a;\theta_i)$, where θ_i are the parameters (weights) of the neural net at iteration i. Training a deep RL agent requires a dataset of experiences $D = \{e_1, ...e_N\}$ (also referred to as 'experience replay memory'), where every learning experience is described as a tuple $e_t = (s_t, a_t, r_t, s_{t+1})$. Inducing the Q function consists in applying Q-learning updates over minibatches of experience $MB = \{(s,a,r,s') \sim U(D)\}$ drawn uniformly at random from the full dataset D. A Q-learning update at iteration i is thus defined according to the loss function

$$L_i(\theta_i) = \mathbb{E}_{MB}\left[(r + \gamma \max_{a'} Q(s',a';\overline{\theta}_i) - Q(s,a;\theta_i))^2\right],$$

where θ_i are the parameters of the neural net at iteration i, and $\overline{\theta}_i$ are the target parameters of the neural net at iteration i. The latter are held fixed between individual updates. This process is implemented in the learning algorithm *Deep Q-Learning with Experience Replay* described in [1].

III. Method

Our proposed method to scale up Deep Reinforcement Learning (DRL) for multi-domain neural-based dialogue agents has three stages. First, multi-policy learning via a network of DRL agents; second, more compact state representations by compressing raw inputs; and third, a pre-training stage using concurrent dialogues to bootstrap the behaviour of dialogue policies. Although these three stages can be applied independently, their combination aims for further scalability than any one of them individually.

A. Network of Deep Q-Networks (NDQN)

We propose to optimise multi-domain neural-based dialogue agents using a network of Deep Reinforcement Learners, for example a network of Deep Q-networks (DQN) — see [1], [4] for an introduction to the standard DQN method. In our method, instead of training a single DQN, we train a set of DQNs (also referred to as NDQN), where every DQN represents a specialised skill to converse in a particular subdialogue — see Figure 1. The network of agents enable DQNs to be executed without a fixed structure in order to support flexible and unstructured dialogues. In contrast to Hierarchical DQNs [5] that follow a strict sequence of agents, an NDQN in our method allows transitions between all DQN agents except for self-transitions. The latter using a stack-based approach as in [6]. While user responses can motivate transitions to another domain in the network, completing a subdialogue within a domain motivates a transition to the previous domain to resume the interaction. Algorithm 1 describes the procedure to train and execute NDQN agents.

An optimal policy in an NDQN performs action selection according to

$$\pi^*_{\theta^{(d)}}(s) = \arg \max_{a \in A^{(d)}} Q^{*(d)}(s,a;\theta^{(d)}), \quad (1)$$

where domain or skill $d \in D$ is selected according to

$$d = \arg \max_{d' \in D} F(d'|d,s'), \quad (2)$$

and evidence s' takes into account all features that describe the environment state of domain d. While this transition function (Eq. 2) is used for high-level transitions in the interaction, Eq. 1 is used for low-level transitions within a node (skill) in the network and subject to reinforcement learning. NDQN assumes that the domain transition function F can be deterministic or probabilistic (the latter due to uncertainty in the interaction), and it is a prior requirement for NDQN-Learning.

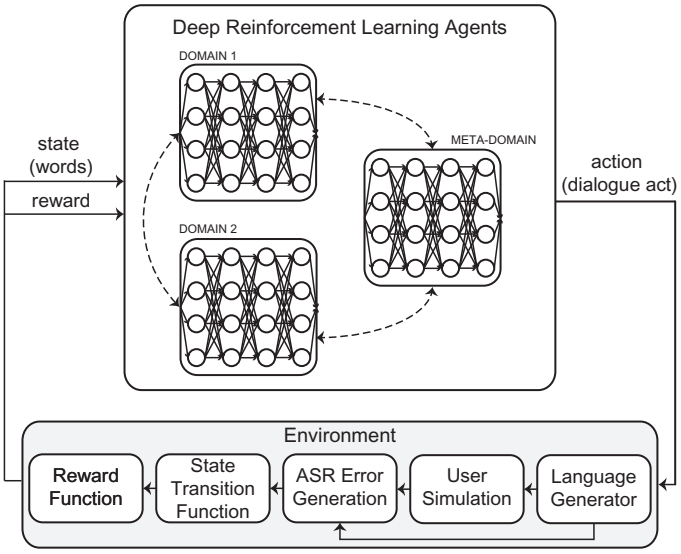


Fig. 1. Illustration of NDQN dialogue agents. The dashed arrows connecting domains denote flexible transitions between domains in order to avoid a rigid structure in the interaction. Although all policies are considered for decision-making, only one domain can be executed at a time implying that a previous domain can continue its execution in order to resume the interaction.

B. NDQN with Compressed Raw Inputs

Previous work on dialogue policy learning using DRL map raw (noisy) text to actions [2], [7]. This is not only computationally intensive, but it becomes infeasible for dialogue systems with large vocabularies. To tackle this problem we propose to use delexicalised sentences (similarly as in [8]) and synonymised sentences. This has the advantage that dialogue policies can be trained from more compact state representations than those using only raw inputs, and have coverage for a larger vocabulary than trained for.

Algorithm 1 Network of Deep Q-Learners (NDQN)

Initialise set of Deep Q-Networks for all domains $d \in \mathcal{D}$ with replay memories $D^{(d)}$, action-value functions $Q^{(d)}$ with random weights $\theta^{(d)}$, and target action-value functions $\hat{Q}^{(d)}$ with weights $\hat{\theta}^{(d)} = \theta^{(d)}$
repeat
  Set initial domain d, predefined or defined by $\arg\max_{d \in D} F_o(d)$
  Set initial environment state $s \in S^{(d)}$
  repeat
    repeat
      Choose action $a \in A^{(d)}$ in s derived from $Q^{(d)}$ (e.g. ϵ-greedy)
      Execute action a and observe reward r and next state s'
      Set next domain d' according to $\arg\max_{d' \in D} F(d'|d, s')$
      Append transition (s, a, r, s') to $D^{(d)}$
      Sample random minibatch of experiences $(s, a, r, s')_j \in D^{(d)}$
      $y_j = \begin{cases} r_j & \text{if } s \text{ is terminal} \\ r_j + \gamma \max_{a \in A^{(d)}} \hat{Q}^{(d)}(s', a'; \hat{\theta}^{(d)}), & \text{otherwise} \end{cases}$
      Gradient descent step on $\left(y_j - Q^{(d)}(s', a'; \theta^{(d)})\right)^2$
      Set $\hat{Q}^{(d)} = Q^{(d)}$
      Set $s = s'$
    until s is a terminal state or $d \neq d'$
    Set $d = d'$
  until s is a goal state
until convergence

1) Delexicalisation: Consider a dialogue system for restaurant search receiving the following user request—with corresponding delexicalised sentence underneath.

I am looking for italian food in the city centre
I am looking for $foodtype food in the $area

The latter representation combining words and slot IDs (denoted with the symbol '$') has several practical advantages. For example, policies can be learnt faster, they contribute to further scalability of systems with large vocabularies, and policies do not have to be retrained if the slot values change over time. In this work we use heuristics to replace slot values by slot IDs, and a trainable component for automatic slot labelling is considered beyond the scope of this paper.

2) Synonymization: Consider the same system above receiving the following user request given the unknown words 'fancy' and 'cuisine'—with corresponding synonyms underneath.

We fancy italian cuisine in the centre of town
We want italian food in the centre of town

We argue that word synonyms can be useful in such situations because the unknown word 'fancy' can trigger the known word feature 'want'. Similarly, the unknown word 'cuisine' can trigger the known word feature 'food'. In this way, the vocabulary of our NDQN incorporates a mapping from filler words and slot values to synonyms in order to cope with unseen wordings. We generated synonyms automatically from word embeddings [9]. Unfortunately, they were not very meaningful and in cases slot values conflicted (e.g. 'north' and 'south'). This work used manually specified synonyms and the automatic generation of meaningful synonyms is left as future work.

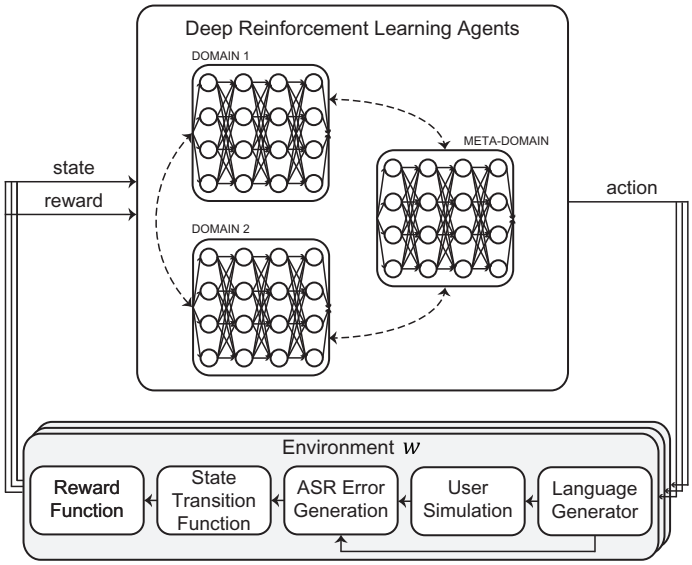


Fig. 2. Illustration of NDQN dialogue agents with policy pre-training. The multiple arrows connecting the agents and environments denote concurrent agent-environment interactions. While all environments w are used concurrently during pre-training (using multi-threaded dialogues), only one environment is used after policy pre-training (using single-threaded dialogues).

C. Policy Pre-Training

Our method above performs policy learning using a single dialogue at a time. This means that during training an agent selects an action, provides a response to the environment, waits for a user response, and so on until the end of the dialogue. Motivated by [10], we extend our method above by incorporating a pre-training phase using multiple concurrent dialogues rather than single-threaded dialogues. To avoid domain mixing in concurrent dialogues and potentially unstable behaviour, we propose to perform the pre-training phase per domain. We consider this pre-training phase as part of initialising the behaviour of policies in multi-domain dialogue systems. Once the initialisation phase is over, the training phase carries on as in the method above. A policy in a concurrent DQN is expected to perform action selection according to

$$\pi^*_{\theta^{(d)}}(w, s) = \arg\max_{a \in A^{(d)}} Q^{*(d)}(w, s, a; \theta^{(d)}), \qquad (3)$$

where $d \in \mathcal{D}$ is a domain or skill and $w \in \mathcal{W}$ is a thread ID. The latter is used to denote multiple copies of the environment, and is also required to track the agent-environment interactions in order to use them accordingly. For example, a decision made by the agent interacting with environment $w = 10$ produces a response that is sent to this particular environment, which observes state and reward of environment $w = 10$. Figure 2 illustrates agent-environment interactions in our extended method. While multi-threaded agent-environment interactions are used during pre-training, single-threaded agent-environment interactions are used after pre-training. Algorithm 2 describes the procedure to train NDQN agents with a pre-training phase.

Algorithm 2 NDQN with Pre-training

1: Initialise set of Deep Q-Networks for all domains $d \in \mathcal{D}$ with replay memories $D^{(d)}$, action-value functions $Q^{(d)}$ with random weights $\theta^{(d)}$, target action-value functions $\hat{Q}^{(d)}$ with weights $\hat{\theta}^{(d)} = \theta^{(d)}$, and number of concurrent dialogues $\mathcal{W}$ during pre-training
2: **for each** domain d in $\mathcal{D}$ **do** ▷ Pre-Training Phase
3: $c \leftarrow$ initialise number of pre-training dialogues
4: **for each** thread w in $\mathcal{W}$ **do**
5: **repeat** concurrently
6: Lines 4-10 and 12-18 of Algorithm 1, and increment c
7: **until** $c > \max$(number of pre-training dialogues)
8: **end for**
9: **end for**
10: **repeat** ▷ Training Phase
11: Lines 4 to 18 of Algorithm 1
12: **until** convergence

IV. Multi-Domain Dialogue System

The proposed framework for training multi-domain neural-based dialogue agents is a substantial extension from the publicly available software tools SimpleDS [2] and ConvnetJS [11]. It can be executed in training or test mode using simulations or speech-based interactions (via a mobile App[1]). Our dialogue system runs under a client-server architecture, where the learning agents—one per domain—act as the *clients* and the dialogue system as the *server*. They communicate by exchanging messages, where the clients communicate to the server the action to execute, and the server communicates to the clients the state and reward observed. The elements for training NDQN-based dialogue systems are as follows.

a) State Spaces: They include word-based features depending on the vocabulary of each learning agent. They include 177 unique words[2] without synonyms, and 150 unique words with synonyms. For example, an agent in the domain of restaurants has relevant features for its domain and it is agnostic of features in other domains. While words derived from system responses are treated as binary variables (i.e. word present or absent), the words derived from noisy user responses can be seen as continuous variables by taking ASR confidence scores into account. Since a single variable per word is used, user features override system ones in case of overlaps.

b) Action Spaces: They include dialogue acts for the targeted domains—currently 69 unique actions in total. Example dialogue act types, dialogue acts without slot-values, are as follows: Salutation(), Request(), AskFor(), Apology(), ExpConfirm(), ImpConfirm(), Retrieve(), Provide(), among others. The set of slots include the following: meta={domain}; restaurants={food_type, area, price}; hotels={city, day, month, nights}. Rather than learning with whole action sets, our framework supports learning from constrained actions by applying learning updates only on the set of valid actions. These actions are derived from the most likely actions, $Pr(a|s) > 0.0001$, from Naive Bayes classifiers (due to scalability purposes) trained from example dialogues. See example demonstration dialogue in Appendix A. In addition to the most probable data-like actions, the constrained actions are extended with legitimate requests, apologies and confirmations. The fact that constrained actions are data-driven and driven by domain-independent heuristics, facilitates its usage across domains.

c) State Transition Functions: They are based on numerical vectors representing the last system and user responses[3]. Taking a wider dialogue context is also possible but not explored in this paper. The system responses are straightforward, 0 if absent and 1 if present (hit-or-miss). The user responses correspond to the confidence level [0..1] of noisy user responses. While system responses are generated from templates, user responses are generated from semi-random user behaviour. The latter is based on sampling user actions from those observed in example interactions, and randomly selecting an observed verbalisation. These elements enable the creation of a vast amount of different conversations for agent training.

d) Domain Transition Function: This function specifies the next domain or task in focus. It is currently defined deterministically, and it is also implemented as an SVM classifier trained from example interactions—see Appendix A. The design of this classifier follows that of a two-deep fully connected neural network with 80 nodes in each hidden layer, with tanh activation, and an SVM output layer, using Hinge Loss. While the input layer accepts domain-independent *words-as-features* vectors representing the unique global vocabulary shared amongst all domains in a hit-or-miss approach, the output layer has 3 classes representing system domains (meta, restaurants and hotels). We refer to meta domain as subdialogues containing domain-general system and user responses. 15K dialogues of data were generated, partitioned as a 60-40 training-testing split, and trained for 180 epochs. Initial results of this classifier shows a 87.5% classification accuracy on user-simulated data.

e) Reward Function: It is defined as $R(s, a, s') = GR + DR - DL$. Briefly, GR is a goal-based score treated as task success [0..1] (i.e. the proportion of positively confirmed slots and information retrieved and presented). DR is a data-like probability of having observed action a in state s. DR scores are derived from Naive Bayes classifiers to allow statistical inference over actions given states ($Pr(a|s)$). Finally, $DL = t * w$ is a dialogue length score used to encourage efficient dialogues with t time steps and weight w (-0.1 in our case).

f) Model Architectures: We use fully-connected multilayer neural nets, trained with stochastic gradient descent, where nodes in the input layers depend on the vocabulary of each agent. The use of convolutional neural nets is work in progress. They include 2 hidden layers with 80 nodes with Rectified Linear Units to normalise their weights [12]. Other hyperparameters include experience replay size=10000, burning steps=1000, discount factor=0.7, minimum epsilon=0.001, batch size=32, learning steps=30000, number of threads in pre-training=10, and number of pre-training dialogues=500.

[1]https://youtu.be/B5fZfZ-xaKM

[2]The unique words in our system's vocabulary excludes words from information presentation due to the vast amount of information about hotels and restaurants. Nonetheless and during testing, our system retrieves live information from http://www.bookatable.co.uk) and www.reservetravel.com.

[3]Representing dialogues using the entire set of system-user responses is left as future work.

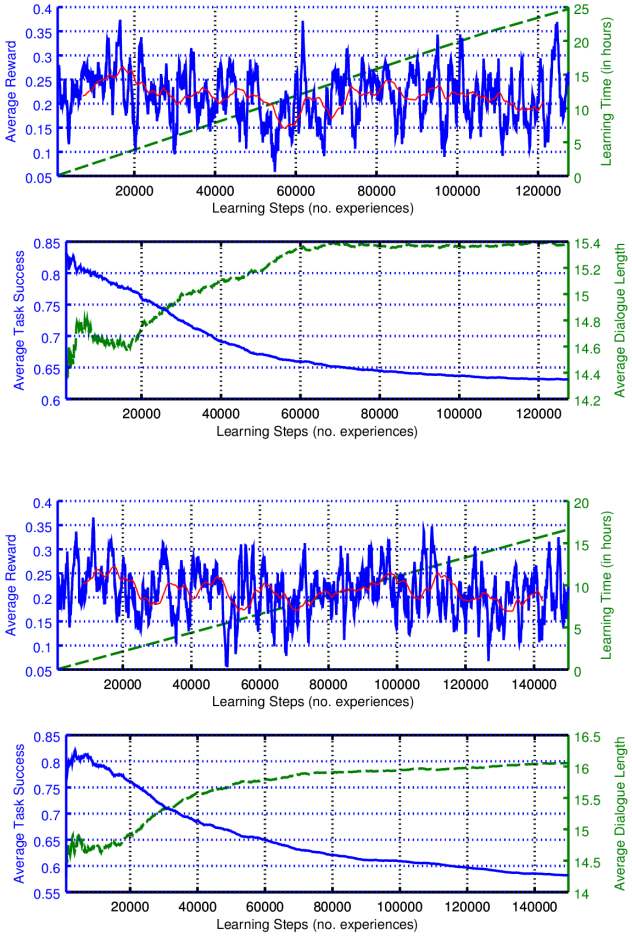
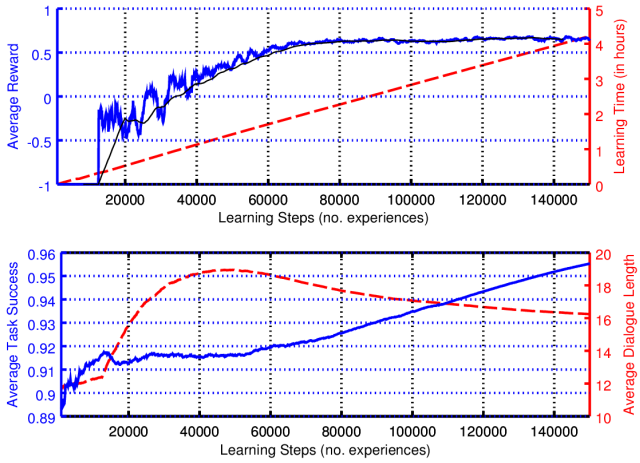


Fig. 3. Learning curves of the baseline DQN-based system: (top 2 plots) without input compression, (bottom 2 plots) with input compression. The higher the better in blue straight lines, and the lower the better in other metrics.

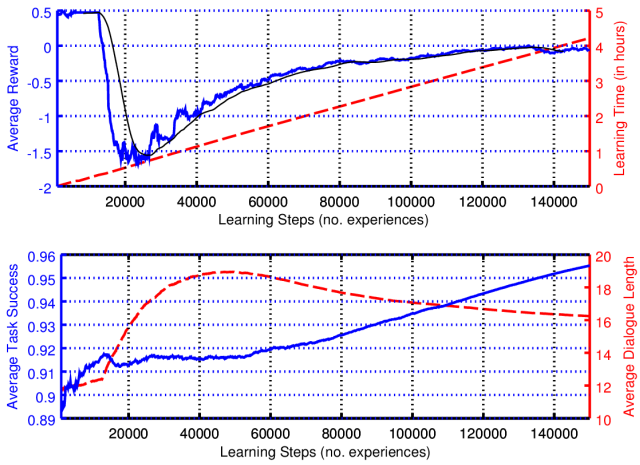
V. Experimental Results

In this section we compare a multi-domain dialogue system using a standard DRL method versus our proposed method described in Sections III and IV. While the former (DQN) uses a single policy for learning (*baseline*), the latter (NDQN) uses multiple policies with and without input compression and policy pre-training (*proposed*). Both multi-domain dialogue systems use the same data, resources and hyperparameters for training. The only difference between both systems is the learning method (DQN or NDQN), state representation (with or without compression), and training approach (with or without policy pre-training).

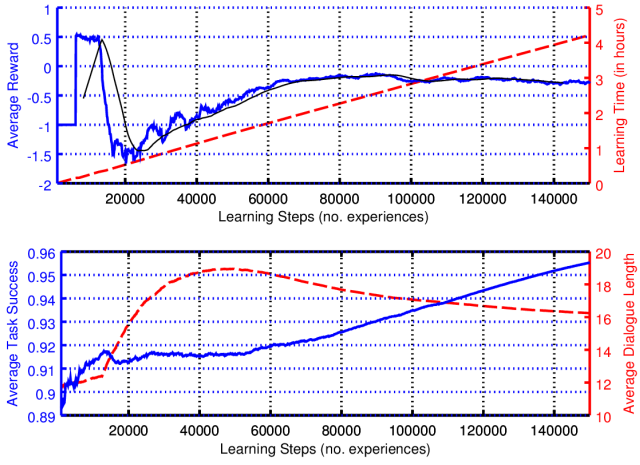
We use four different metrics to measure system performance: avg. reward, learning time (in hours), avg. task success, and avg. dialogue length (i.e. avg. actions per dialogue). The higher the better in the first and third, and the lower the better in the second and fourth. Figure 3 shows learning curves for the baseline DQN-based system, Figure 4 shows learning curves for the proposed NDQN-based system without pre-training, and



(a) Meta Domain with input compression and pre-training

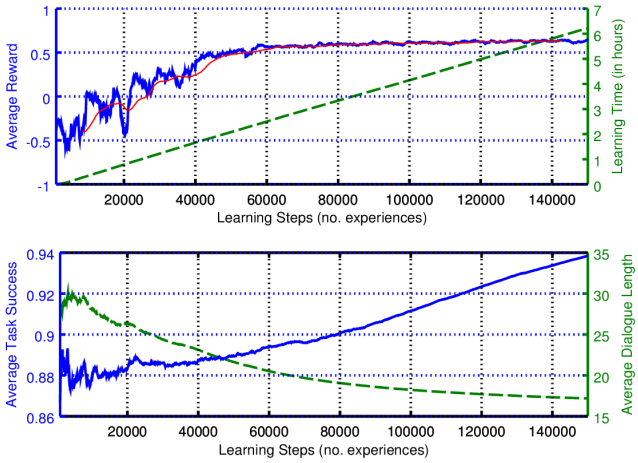


(b) Restaurants Domain with input compression and pre-training

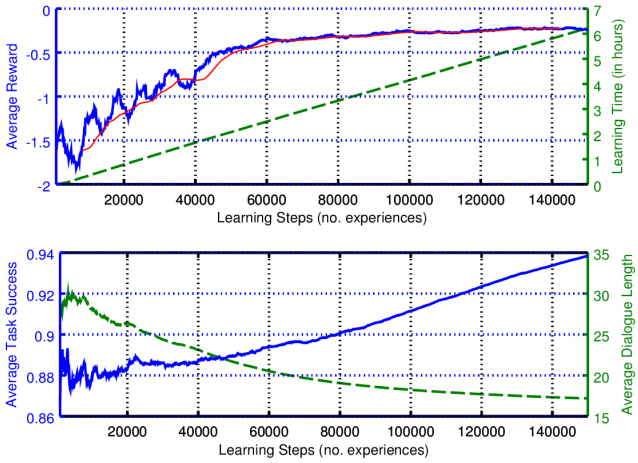


(c) Hotels Domain with input compression and pre-training

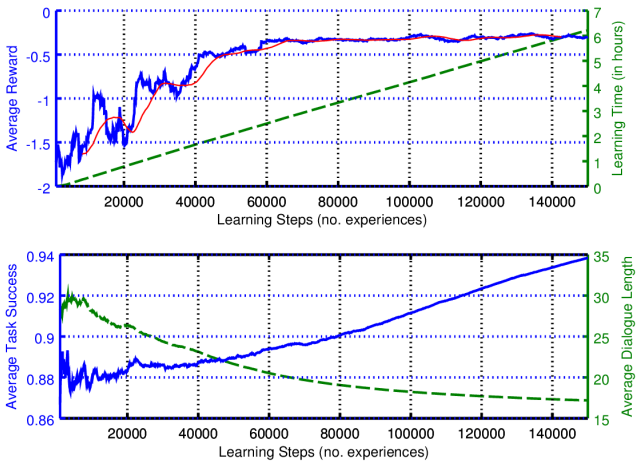
Fig. 5. Learning curves of the NDQN method with policy pre-training



(a) Meta Domain: (left plots) without input compression, (right plots) with input compression



(b) Restaurants Domain: (left plots) without input compression, (right plots) with input compression



(c) Hotels Domain: (left plots) without input compression, (right plots) with input compression

Fig. 4. Learning curves of the proposed NDQN-based system without policy pre-trainig. The higher the better in avg. reward and avg. task success, and the lower the better in other metrics. The plots on the left correspond to our proposed system with word-based features, and the plots on the right correspond to our proposed system with delexicalised inputs as features. The latter plots show no performance degradation despite of using more compact state representations.

Method / Input Type	Without Input Compression	With Input Compression
Baseline (DQN)	28.57 hrs	16.63 hrs
Proposed (NDQN without pre-training)	6.21 hrs	6.05 hrs
Proposed (NDQN with pre-training)	6.04 hrs	**4.21** hrs

TABLE I
LEARNING TIMES OF THE BASELINE AND PROPOSED METHODS

Figure 5 shows learning curves for the proposed NDQN-system with pre-training. Both the baseline and proposed system report results over 150K learning steps (about 8700 dialogues without pretraining and 9200 dialogues with pre-training). Our results report that training multi-domain systems using a single policy is twofold harder than using a multi-policy approach. First, this is evidenced by the fact that the baseline policies do not improve over time[4], and the policies with the proposed method do. This is presumably due to the abstraction exhibited in the multi-policy approach—more focused system actions rather than interleaving them across domains. Second, our proposed system without pre-training learned 4.7 times faster than the baseline, which was accelerated further to 6.8 times faster by using pre-training[5]—see Figure V and Table I. Faster training can be explained by: (a) the use of less parameters (weights) in the NDQN-based system without pre-training than the baseline, and (b) the use of a more guided decision-making in the NDQN-based system with pre-training. Note that the NDQN-based system with and without pre-training use the same amount of parameters, their difference is in bootstraped behaviours in the case of policy pre-training. By applying synonymization we are able to use a smaller vocabulary when training and then a much wider vocabulary at runtime, which adds robustness in the presence of unseen dialogues. These results show indication of better scalability for NDQN to multiple domains.

Although the currently generated dialogues using the trained policies seem reasonable[6], it is natural to ask *How good are the DRL-based policies?* To answer this question we integrated a K-Nearest Neighbour (KNN) baseline [13], which aims to behave as the example demonstration dialogues—see Appendix A. We ran 1000 test dialogues using our fastest learnt DRL policies and compared them against 1000 KNN-based dialogues[7]. The KNN baseline used the same features, actions, and demonstration dialogues as the DRL agents with input compression and pre-training. Our results report that the DRL-based policies achieved an average of 14.3 actions per dialogue and 100% of task success, and the KNN-based behaviours achieved an average of 16.8 actions per dialogue and 95.8% of task success[8]. This is evidence that the DRL-based policies can produce more successful and efficient interactions than other baseline behaviours (KNN-based in our case).

[4]We validated these performance results using different model architectures with 80, 120, and 150 nodes in the hidden layers.

[5]Ran on i5-3210M CPU@2.50GHz x 4; 8GiB DDR4 RAM@2400MHz.

[6]https://youtu.be/B5fZfZ-xaKM

[7]We tried KNN with K=$\{1, 2, 3, 4, 5\}$ and obtained best results with $k = 4$.

[8] Significant at $p < .05$ using a two-tailed Wilcoxon- Signed Rank Test.

VI. RELATED WORK AND DISCUSSION

Multi-domain dialogue agents are receiving an increasing amount of attention. This can be attributed to the increasing maturity of speech technologies. But the question of *How to design conversational agents for human-machine interaction in multiple domains (or tasks)?* is still an open and interesting problem in artificial intelligence. The dialogue system proposed by [14] used a distributed architecture of domain experts modulated by a domain selector. The latter used a decision tree with classification errors over 20% in 5 domains. This indicates that not only individual domains have to exhibit robust interactions against errors, but also that errors increase by incorporating more domains.[15] used rule-based classifiers for predicting user intentions, which are executed using a Hierarchical Task Network incorporating expert knowledge. Trainable multi-domain dialogue systems using traditional reinforcement learning include [16], [17], [18], [19]. These systems use a modest amount of features, and in contrast to neural-based systems, they require manual feature engineering.

Recent work on neural-based task-oriented dialogue agents include the following. [20] uses a Recurrent Neural Network (RNN) for dialogue act prediction in a POMDP-based dialogue system, which focuses on mapping system and user sentences to dialogue acts. [21] applies DRL with a fully-connected neural network for trading negotiations in board games, which focuses on mapping game situations to dialogue actions. [22] trains RNN-based classifiers for predicting dialogue success in multi-domain dialogue systems, which can be applied to unseen domains. [23] also trains RNN-based classifiers but for belief tracking to improve the robustness of recognised user responses across dialogue turns. Other neural-based dialogue agents have been applied to text prediction using the sequence-to-sequence approach [24], [25], and to reasoning with inference for text-based question answering [26].

We observe from these works that supervised learning is playing an important role in neural-based conversational agents. We also observe that recent DRL-based dialogue systems have focused on a single domain [21], [2], [27]. To our knowledge, we report one of the first multi-domain dialogue system using deep reinforcement learning. Future work includes applying neural-based dialogue systems to larger sets of domains, to language generation using a divide-and-conquer approach [28], to multi-task multimodal interaction using different types of devices and machines, and to evaluate neural-based systems in realistic scenarios with genuine users.

VII. CONCLUSION AND FUTURE WORK

The contribution of this paper is a novel method for training multi-domain dialogue agents in a more scalable way than traditional deep reinforcement learning, e.g. using the DQN method. The proposed method uses a Network of DQN (NDQN) agents in order to train specialised agents, compression of input features, and a pre-training phase using concurrent dialogues. Experimental results using simulations report that the proposed method (NDQN) can train policies faster and more effectively than DQN—up to 7 times faster than DQNs.

In addition, our results report that input compression and pre-training contribute faster learning without performance degradation across metrics. Furthermore, our fastest trained policies using NDQN with pre-training showed to be more successful and efficient than a K-nearest neighbour baseline. The proposed NDQN-based dialogue agents are the first applications of deep reinforcement learning to optimising the behaviour of multi-domain human-machine dialogues. A speech-based evaluation is left as future work.

ACKNOWLEDGMENTS

Funding from Samsung Electronics is gratefully acknowledged. We would like to thank Jihie Kim, Gary G. Lee and the Samsung's Software Center AI team for helpful discussions.

APPENDIX

We aim for a data-efficient system, trainable from a small set of demonstrations. Our agents showed to be data-efficient due to the amount of data used to induce the simulated environment and baseline behaviour, which is relevant for initial training from a small set of example demonstrations. The dialogue below is one of the 12 manually-specified seed demonstrations for inducing the system's features (words) and actions (dialogue acts), and for training classifiers to predict the most likely actions given the previous system and user responses. The latter were used by the DRL agents described in Section III for inducing their dialogue behaviour according to the learning environment described in Section IV. Square brackets denote system actions, double quotes denote system or user responses, and words beginning with an underscore encapsulate live web-based information according to the user requests. During testing, our system retrieves live information from http://www.bookatable.co.uk) and www.reservetravel.com.

```
SYS:[Salutation(greeting)]"Welcome to MultiDS!"
USR:null
SYS:[Provide(intro)]"You may ask for information about hotels and restaurants."
USR:null
SYS:[Request(hmihy)]"How can I help you?"
USR:"I am looking for a hotel in edinburgh on the 2nd of january for 2 nights"
SYS:[Apology(h_nights)]"Sorry, how long?"
USR:"2 nights"
SYS:[ExpConfirm(h_day=$h_day,h_month=$h_month,h_nights=$h_nights)]"Did you say
on the 2nd of january for 2 nights?"
USR:"yes i did"
SYS:[ExpConfirm(h_city=$h_city)]"Did you say in edinburgh?"
USR:"i did"
SYS:[Retrieve(h_info)]"Let me see."
USR:null
SYS:[Provide(h_info)]"I found the following hotels. _hotels."
USR:null
SYS:[AskFor(h_more)]"Anything else?"
USR:"I am looking for cheap japanese food in the centre of town"
SYS:[Apology(r_area)]"Sorry, what area?"
USR:"In the centre"
SYS:[ExpConfirm(r_price=$r_price,r_area=$r_area)]"Did you say cheap food in the centre?"
USR:"yes i did"
SYS:[ExpConfirm(r_food=$r_food)]"Did you say japanese food?"
USR:"i did"
SYS:[Retrieve(r_info)]"Let me see."
USR:null
SYS:[Provide(r_info)]"I found the following restaurants. _res."
USR:null
SYS:[AskFor(more)]"Anything else?"
USR:"no"
SYS:[Salutation(closing)]"Okay, talk to you soon. Bye!"
USR:null
```

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*, 2013.

[2] H. Cuayáhuitl, "SimpleDS: A simple deep reinforcement learning dialogue system," *CoRR*, vol. abs/1601.04574, 2016.

[3] H. Cuayáhuitl, S. Yu, A. Williamson, and J. Carse, "Deep reinforcement learning for multi-domain dialogue systems," *CoRR*, vol. abs/1611.08675, 2016.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, 02 2015.

[5] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. B. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," *CoRR*, vol. abs/1604.06057, 2016.

[6] H. Cuayáhuitl, I. Kruijff-Korbayová, and N. Dethlefs, "Nonstrict hierarchical reinforcement learning for interactive systems and robots," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 4, no. 3, 2014.

[7] T. Zhao and M. Eskénazi, "Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning," *CoRR*, vol. abs/1606.02560, 2016.

[8] M. Henderson, B. Thomson, and S. J. Young, "Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation," in *IEEE Spoken Language Technology Workshop, SLT*, 2014.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.

[10] C. W. Anderson, M. Lee, and D. L. Elliott, "Faster reinforcement learning after pretraining deep networks to predict state dynamics," in *IJCNN*, 2015.

[11] A. Karpathy, "ConvNetJS: Javascript library for deep learning," 2015, http://cs.stanford.edu/people/karpathy/convnetjs/.

[12] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.

[13] D. Aha and D. Kibler, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, 1991.

[14] K. Komatani, N. Kanda, M. Nakano, K. Nakadai, H. Tsujino, T. Ogata, and H. G. Okuno, "Multi-domain spoken dialogue system with extensibility and robustness against speech recognition errors," in *SIGdial Workshop on Discourse and Dialogue*, 2006.

[15] H. Jeon, H. R. Oh, I. Hwang, and J. Kim, "An intelligent dialogue agent for the IoT home," in *AAAI Workshop on AI Applied to Assistive Technologies and Smart Environments*, 2016.

[16] H. Cuayáhuitl, S. Renals, O. Lemon, and H. Shimodaira, "Evaluation of a hierarchical reinforcement learning spoken dialogue system," *Computer Speech & Language*, vol. 24, no. 2, 2010.

[17] P. Lison, "Multi-policy dialogue management," in *SIGDIAL*, 2011.

[18] Z. Wang, H. Chen, G. Wang, H. Tian, H. Wu, and H. Wang, "Policy learning for domain selection in an extensible multi-domain spoken dialogue system," in *EMNLP*, 2014.

[19] M. Gasic, N. Mrksic, P. Su, D. Vandyke, T. Wen, and S. J. Young, "Policy committee for adaptation in multi-domain spoken dialogue systems," in *ASRU*, 2015.

[20] W. Ge and B. Xu, "Dialogue management based on multi-domain corpus," in *SIGDIAL*, 2015.

[21] H. Cuayáhuitl, S. Keizer, and O. Lemon, "Strategic dialogue management via deep reinforcement learning," *CoRR*, vol. abs/1511.08099, 2015.

[22] D. Vandyke, P. Su, M. Gasic, N. Mrksic, T. Wen, and S. J. Young, "Multi-domain dialogue success classifiers for policy training," in *ASRU*, 2015.

[23] N. Mrksic, D. Ó. Séaghdha, B. Thomson, M. Gasic, P. Su, D. Vandyke, T. Wen, and S. J. Young, "Multi-domain dialog state tracking using recurrent neural networks," *CoRR*, vol. abs/1506.07190, 2015.

[24] I. V. Serban, A. Sordoni, Y. Bengio, A. C. Courville, and J. Pineau, "Hierarchical neural network generative models for movie dialogues," *CoRR*, vol. abs/1507.04808, 2015.

[25] O. Vinyals and Q. V. Le, "A neural conversational model," *CoRR*, vol. abs/1506.05869, 2015.

[26] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *CoRR*, vol. abs/1410.3916, 2014.

[27] M. Fatemi, L. E. Asri, H. Schulz, J. He, and K. Suleman, "Policy networks with two-stage training for dialogue systems," 2016.

[28] N. Dethlefs and H. Cuayáhuitl, "Hierarchical reinforcement learning for situated natural language generation," *Natural Language Engineering*, vol. 21, no. 3, pp. 391–435, 2015.