TOWARDS SCALABLE INFORMATION-SEEKING MULTI-DOMAIN DIALOGUE

Alexandros Papangelis, Margarita Kotti, Yannis Stylianou

Speech Technology Group, Toshiba Cambridge Research Lab

ABSTRACT

Multi-domain dialogue systems face challenges such as scaling algorithms to handle large ontologies, or transferring trained policy models to unseen domains. We attempt to address these challenges by proposing a dialogue management architecture that has an abstracted view of the world but yet is able to focus on relevant parts of the ontology at runtime. Specifically, we train a sub-domain identifier neural network that learns which features are relevant to the current turn and the immediate future, thus filtering out irrelevant information from the ontology and consequently the belief space at each dialogue turn. We then train a policy network that needs: a) to adapt to the sub-domain identifier's output; and b) to learn what information will carry over from previous turns and when it needs to be updated. We evaluate our method on a large information-seeking ontology that contains latent sub-domains. Our results in simulation and a small human trial show that the sub-domain identifier is able to generalise to unseen domains and achieve performance on par with a multi-domain dialogue manager where each sub-domain is carefully defined (golden standard).

***Index Terms*—** Spoken Dialogue Management, Multiple Domains, Dialogue Policy, Policy Network

1. INTRODUCTION AND RELATED WORK

As we move towards Spoken Dialogue Systems (SDS) that are able to converse about multiple topics, we face challenges in scaling these systems to real-world applications. Such challenges include handling large domain ontologies and transferring information across domains. In this work, we attempt to address the first challenge by proposing a dialogue management model that has an abstracted view of the world but yet is able to focus on the topic under discussion at each dialogue turn. Assuming that not all of the knowledge encoded by an ontology is needed in every dialogue, we present a method for filtering out irrelevant information from a domain ontology and consequently the belief space, at each dialogue turn. We achieve this by training a neural network that learns which features are relevant in the current turn, including features that will be useful in the immediate future, effectively defining a sub-domain at each turn. We address the second challenge by using a domain-independent policy model that uses the aforementioned network's output to judge when information obtained from other sub-domains needs to be updated. The main idea behind our architecture is to have a single representation for each ontology attribute, e.g. have a single representation for *price* regardless of any specific items it refers to (a flight, a laptop, a loan, etc.), and let the belief tracker and dialogue policy network decide when the value of each attribute (slot) needs to be confirmed or updated.

Moreover, if the ontology represents general knowledge (e.g. Wikipedia), then this modelling allows the system to operate only on the relevant part of the ontology and to more fluidly move between topics when conversing with human users. We build on prior work on multi-domain SDS that use a single policy model across information-seeking domains [1], by operating in a domain-independent feature space [2]. This allows us to train a policy network that works even if the domain (or sub-domain in this case) changes in real time. Last, we alleviate some of the costly burden of carefully designing ontologies for each domain the system can talk about, and only require a generic ontology; this can be very beneficial in cases where we need to extend the domains, for example. Figure 1 shows the overall architecture of the SDS, where SLU identifies the users' intentions, Belief Tracking outputs some hypotheses about the current dialogue state which is then input to SDI. SDI filters the parts of the ontology that are deemed relevant for the current turn and generates the respective sub-domain, which is passed on to the domain independent feature extractor and policy. NLG then generates the system's output.

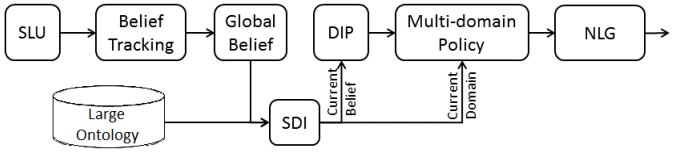

Fig. 1. The architecture of our sub-domain identifier SDS.

While our work may have similarities to topic tracking, we do not use any linguistic features or track the current sub-domain in the traditional topic-tracking sense. What our architecture does is filter out irrelevant parts of the belief state and subsequently the domain ontology (including system actions), implicitly defining features that are relevant to the current turn (which include features that are useful to make decisions about the immediate future). Therefore, it is not nec-

essary that all of the relevant features for a given dialogue will be active on every dialogue turn, as would be the case if we were tracking the topic. This means that our work is not directly comparable to topic tracking.

In related work, Gašić et al. [3] proposed a hierarchical structure to train generic dialogue policies that can be refined when in-domain data become available. A Bayesian Committee Machine (BCM) over multiple domain-specific dialogue policies decides which policy can better handle the input, and delegates control to that policy. In [4], the authors use Deep RL to train a multi-domain Dialogue Manager (DM) represented as a network of Deep-Q policy networks (NDQN), each of which learns a domain-specific dialogue policy. Our work is different in that we train a single Deep-Q network that is able to operate across domains, therefore making it much more scalable. Crook et al. [5] present the Task Completion Platform where the main idea is to decouple task definition from the dialogue policy via a task description language that defines each task which has an associated task-specific policy.

A relevant direction of research is that of decomposing a complex (dialogue) task into a hierarchy of sub-tasks [6, 7, 8, e.g.], that has recently re-emerged in the literature. [9] train a hierarchical policy using deep reinforcement learning that has two parts, the top-level (more generic) policy and the low-level (more domain-specific) policy using semi-MDPs. [10] also train a hierarchical policy in a similar framework but using GP-SARSA as the core learning method. Both methods attempt to address the fact that domains should share information, as what happens in one domain (e.g. flight booking) can affect another domain (e.g. hotel booking); our method attempts to implicitly learn such information transfers across domains, and learns when to keep a slot's value or when to ask for more information. Zhao et al. [11] proposed to chain a belief update network and a policy network, however that work concerns single-domain dialogues.

2. SUB-DOMAIN IDENTIFICATION (SDI)

We here briefly present the problem we are trying to address, within the statistical dialogue management framework. A POMDP DM typically receives an n-best list of language understanding hypotheses, which are used to update the belief state, reflecting an estimate of the user's goals. The system then selects a response that maximises the long-term return of the system. Concretely, a POMDP is defined as a tuple $\{Z, A, T, O, \Omega, R, \gamma\}$, where Z is the state space, A is the action space, $T : Z \times A \rightarrow Z$ is the transition function, $O : Z \times A \rightarrow \Omega$ is the observation function, Ω is a set of observations, $R : Z \times A \rightarrow \Re$ is the reward function and $\gamma \in [0, 1]$ is a discount factor of the expected cumulative rewards $J = E[\sum_t \gamma^t R(z_t, a_t)]$. A policy $\pi : Z \rightarrow A$ dictates which action to take from each state. An optimal policy $\pi^\star$ selects an action that maximises the expected returns of the POMDP, J. Learning consists exactly of finding such optimal policies; however, due to state-action space dimensionality, approximation methods need to be used for practical applications [12, e.g.].

Let now D be a collection of domains, defined as:

$$D = \{d_i | d_i = \{\phi_{i,1}, ..., \phi_{i,N_i}\}, i = 1, ..., K\} \quad (1)$$

where each domain d_i is represented as a set of features $\phi_{k,n} \in \Phi_D$, and Φ_D is a set of features across all domains. A domain feature can be defined as sets of slots and actions, as abilities (i.e. micro-domains with elementary capabilities such as "retrieve price range"), or any other quantity that defines a conversational domain. Each domain d is associated with an optimal dialogue policy $\pi^\star_d(z_t)$. Given an input u_t at time t, the problem we are addressing can be stated as:

$$max_{\psi_D}\{J(\pi^\star_{\psi_D}, u_t)\}, \psi_D \subseteq \Phi_D \quad (2)$$

It should be noted that ψ_D may or may not exactly match an existing domain $d \in D$. $\pi^\star_{\psi_D}$ is the optimal policy w.r.t ψ_D, i.e. $\pi^\star_{\psi_D}(s_t) = a^\star_t$, where $a^\star_t$ is the optimal action for state z_t. The cumulative reward J is then defined as:

$$J(\pi^\star_{\psi_D}, u_t) = E[\sum_t \gamma^t R(u_t, z_t, \pi^\star_{\psi_D}(z_t))] \quad (3)$$

where $R(u_t, z_t, a_t)$ is the reward function conditioned on the cross-domain user goals underlying u_t.

3. DOMAIN-INDEPENDENT DIALOGUE

Domain Independent Parameterisation (DIP) [2] is a method that maps the (belief) state space into a feature space of size N, that is independent of the particular domain: $\Phi_{DIP}(z, s, a) : Z \times S \times A \rightarrow \Re^N, z \in Z, s \in S, a \in A$, where S is the set of slots (including a 'null' slot for actions such as *hello*). Φ_{DIP} therefore extracts features for each slot, given the current belief state, and depends on A in order to allow for different parameterisations for different actions. This allows us to define a fixed-size domain-independent space, and policies learned on this space can be used in various domains, in the context of information-seeking dialogues. As shown in [1, 13], we can take advantage of DIP to design efficient multi-domain DMs, the main benefit being that we learn a single, domain-independent policy model that can be applied to information-seeking dialogues.

For our experiments, we define the action space $\mathcal{A} := \{inform(s_1, v_1, ..., s_N, v_N), request(s), select(s, v_1, v_2), confirm(s, v), reqmore(), repeat(), hello(), bye()\}$. By operating in $\Phi_{DIP} \times \mathcal{A}$ instead of the original belief-action space, we achieve independence in terms of slots and actions as long as the actions of any domain can be represented as functions of $\mathcal{A} \times S$. The DIP policy therefore maximises over both actions and slots:

$$s_{t+1}, \alpha_{t+1} = argmax_{s,a}\{Q[\Phi_{DIP}(b_t, s, a), a]\} \quad (4)$$

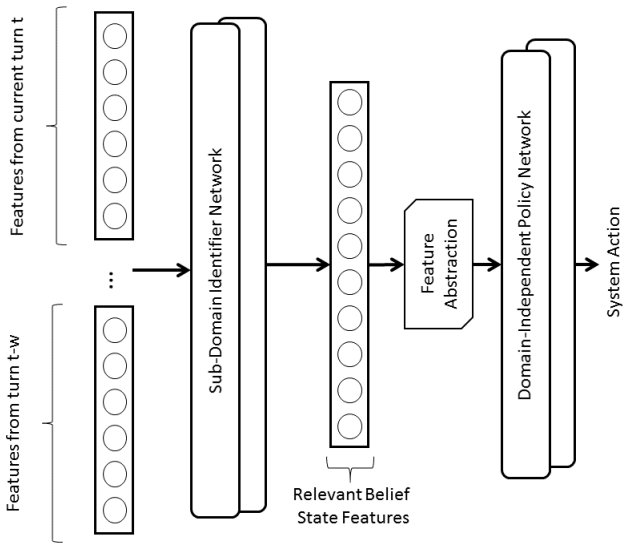

Fig. 2. The architecture of our DNN-based sub-domain DM.

where $\alpha(s) \in \mathcal{A} \times S$ is the selected summary action, b_t is the belief state at time t, and $a \in \mathcal{A}$. To approximate the Q function, we use a 2-layered FFN with 80 and 60 hidden nodes, respectively. The input layer receives the DIP feature vector $\Phi_{DIP}(z, s, a)$ and the output layer is of size $\mathcal{A}$; each output dimension can be interpreted as $Q[\Phi_{DIP}(z, s, a), a]$:

$$\overrightarrow{Q}(\Phi_{DIP}(z_t, s, a)) \approx softmax(W_k^M x_k^{M-1} + c^M) \quad (5)$$

where $\overrightarrow{Q}(\Phi_{DIP}(z_t, s, a))$ is a vector of size $|\mathcal{A}|$, W^m are the weights of the m^{th} layer (out of M layers in total) for nodes k, x_k^m holds the activations of the m^{th} layer, where $x^0 = \Phi_{DIP}(z_t, s, a)$, and c^m are the biases of the m^{th} layer. To generate the summary system action, we simply combine the selected slot and action from equation 4. Figure 2 presents the proposed architecture, where the input B_{in} is passed to the SDI network which outputs the relevant belief features. These are then abstracted through DIP processing and fed into the domain independent policy network that finally outputs the system's next action.

To simulate our use case - a large ontology for information seeking dialogues - we create a unified ontology by selecting the unique slots from the following 12 domains: Cambridge Restaurants (CR), Hotels (CH), Shops (CS), Transportation (CT), and Attractions (CA); San Francisco Hotels (SH) and Restaurants (SR); Toshiba Laptops (with 6 slots and with 11 slots - L6, L11), Televisions (TV), Video Players (VP), and External Hard Drives (HD). This results in an ontology with 32 informable slots, 60 user requestable slots, 31 system requestable slots, and 101 summary actions. Our unified ontology, therefore, is defined through the unique features of Φ_D; thus, we refer to D as the (global) domain and to each $d_i \in D$ as a sub-domain of the unified ontology.

For our experiments, we needed to modify the user simulator [14] to be able to cope with the new setup. Specifically, we restricted the simulated users to sample goals from the original 12 domains, to ensure that database items do exist for the user's sampled goal. We train our policy network using a modified version of the Deep-Q Networks algorithm (DQN) [15]. Specifically, because the reward sparsity problem is even more evident in longer multi-domain dialogues and to mitigate DQN's tendency to overfit, we implement DQN with variance-based exploration and exponential-based sampling of the replay memory [1, 16], with respect to the frequency of each data point's reward. Last, we flush the experience pool after each target network update. These extensions seem to help make the algorithm more robust in our setting.

4. EVALUATION

We evaluate our approach using the simulator described above and compare against a baseline and a golden standard. As our baseline we train a single-domain policy network (SD-DQN) on the unified ontology without SDI, while for the golden standard we train the policy network on a multi-domain setup (MD-DQN), where each sub-domain has been carefully designed and a topic tracker is used to switch between them.

BCM [3], NDQN [4] and similar approaches select a domain d_i for which $Q_{d_i}(z_t, a_t) = max_d\{Q_d(z_t, a_t)\}, d \in D$ and train one policy for each domain. We here relax the assumption to search within D, and search within Φ_D instead, allowing us to define new domains that are created from features of existing domains. Our architecture can thus handle domains not included in the set of pre-defined domains D, and if a sub-domain ψ_D appears often enough, then we can optionally save it ($D' = D \cup \psi_D$) if it meets some criteria.

Input to the sub-domain identifier (SDI) is a window w of past belief states multiplied by a forgetting factor $f \in [0, 1]$:

$$B_{in} = b^{top}(z_t) \oplus b^{top}(z_{t-1}) f \oplus ... \oplus b^{top}(z_{t-w-1}) f^{w-1} \quad (6)$$

where $B_{in} \in [0, 1]^{|Z|w}$ and $b^{top}(z_t)$ is a vector that holds the top beliefs for each slot, plus some domain-independent features (that refer to the dialogue in general). The relevant belief state features B_r therefore are given by:

$$B_r = \{b_t^{(i)}(z_t) | softmax^{(i)}(W_k^M x_k^{M-1} + c^M) > 0.5\} \quad (7)$$

where the superscript (i) represents the i-th element of each vector, W^m are the weights of the m^{th} layer (out of M layers in total) for nodes k, x_k^m holds the activations of the m^{th} layer, where $x^0 = B_{in}$, and c^m are the biases of the m^{th} layer. In our experiments we used a Feed-Forward network to model SDI (2 layers of 155 nodes, w = 3, and f = 0.85).

4.1. Simulation Results

We conduct a series of experiments during our evaluation. First, we test the validity of our method by training on a variety of sub-domains and testing on the same conditions at

Error	0%	5%	10%	15%
	6 train, 6 test (no unseen)			
SDI	81.3 ±3.8	72.5 ±5.5	63.1 ±9.3	60.2 ±4.5
MD	**84.7** ±7.7	**75.8** ±13	**71.2** ±13	**69.6** ±6.6
SD	52.8 ±20	50.1 ±19	49.7 ±18	48.4 ±21
	6 train, 7 test (1 unseen)			
SDI	84.1 ±5.6	**82.0** ±2.5	67.4 ±7.3	**67.5** ±12
MD	**90.9** ±1.8	79.3 ±9.6	**73.1** ±7.4	61.4 ±13
SD	55.1 ±22	48.7 ±20	46.5 ±22	45.9 ±31
	8 train, 9 test (1 unseen)			
SDI	**81.6** ± 4.4	**79.7** ± 5.7	**68.6** ± 7.2	**67.1** ± 9
MD	74.3 ± 10	62.6 ± 21	59.3 ± 31	57.8 ± 12
SD	46.3 ± 28	41.0 ± 19	42.6 ± 23	39.4 ± 24
	8 train, 10 test (2 unseen)			
SDI	**68.0** ± 11	**65.1** ± 15	**64.9** ± 12	**63.1** ± 19
MD	64.3 ± 27	61.1 ± 20	45.0 ± 29	44.7 ± 21
SD	47.6 ± 32	48.4 ± 29	46.6 ± 20	41.4 ± 26

Table 1. Dialogue success rates for the three DMs ± std dev.

various noise levels. At each trial, we sample 6 out of 12 sub-domains (so the belief features relevant to the non-selected sub-domains are effectively noise) and out of these 6 sub-domains the simulated user samples goals for each training and testing dialogue. Next, we assess the generalizability of our setup by evaluating the policies on the trained sub-domains plus extra domains unseen during training. For all our evaluations, we allow 500 training dialogues for the policy network, 100 evaluation dialogues and 5 trials for each noise condition; we randomly select the domains at each trial and present average results. For the SDI-DQN DM, we pre-train the SDI network using 750 simulated training dialogues. Table 1-A shows the results of the matching conditions test and Table 1-B shows the results on mismatch conditions (averaged over the respective sub-domains for each condition).

From the results presented in Table 1-A, it is evident that our architecture performs almost at the same level as the golden standard approach, without the overhead of designing one ontology for each sub-domain. Regarding the results on unseen domains (Table 1-B), one reason why the MD-DQN system does not perform as well as the SDI-DQN is that the semantic error also affects the topic recognition. SDI-DQN is able to better handle such errors, because when it does not have enough (reliable) information, it chooses to let more belief features through the filter, therefore allowing the policy network more flexibility (e.g. to ask further questions). Topic recognition errors do not affect the SD-DQN policy, as it effectively operates on a single domain (the unified ontology domain). However, due to the increased size of the domain, SD-DQN simply needs more data to perform at a level similar to the other two approaches. Overall, as shown in Table 1-B, SDI-DQN can better handle unseen domains, as it outperforms the other two approaches, and handles noise in a better way as its performance degrades more gracefully.

As the number of seen domains increases, SDI performs better on unseen domains because it is more likely that similar domains were seen in training. Of course, the power of SDI-DQN cannot be fully shown here as we cannot have user goals that are between the pre-defined domains, since we cannot have database items that are between domains (e.g. something between a hotel and an attraction). We leave this to be tested in future work using a large and flexible ontology that connects to live semantic knowledge bases [17].

	Task Success	Feedback	Interactions
SDI-DQN	65.0%	**3.67** ±1.1	20
MD-DQN	**68.4%**	3.5 ±1.51	20
SD-DQN	52.3%	3.12 ±1.54	17

Table 2. Results of the human trials (subdom.: 6 train, 8 test). Feedback was given in the range 1(v. bad) to 5(v. good).

4.2. Human Trial Results

To prove that our results transfer to human users, we conducted a small human trial with 6 subjects, as we were not able to use crowd sourcing services due to Intellectual Property restrictions. Table 2 presents the results of the study, where we evaluated all three DMs for a total of 57 interactions. We trained each DM for 500 episodes in simulation, for the following 6 domains: CR, CA, L6, SR, SH, CS. Each participant received a list of tasks, for example *"You are looking for a cheap Chinese restaurant in Cambridge."* from the aforementioned 6 domains plus two extra domains, unseen in training: CH, L11. At the end of each interaction, participants were asked to answer the following question: *"How would you rate the dialogue overall?"*. As summarized in Table 2, the results of our human trial are in accordance with our findings in simulation. Specifically, SDI-DQN performs close to the golden standard (MD-DQN) in terms of task success, and was rated higher in terms of overall dialogue experience. SD-DQN again seems to need more training data in order to perform at the same level as the other DMs.

5. CONCLUSION

We have presented a novel architecture with strong potential to handle large information-seeking ontologies for multi-domain dialogues. We have shown that it performs as well as a carefully designed multi-domain system, and that it can handle unseen domains and semantic noise better. Our approach offers great benefits in reducing the cost of ontology design and expansion, as it only requires generic attributes to be present, and it automatically learns how to combine them into meaningful sub-domains. In current work, we are moving towards complex tasks that require knowledge from various domains in order to be completed [17].

6. REFERENCES

[1] A Papangelis and Y Stylianou, "Single-model multi-domain dialogue management with deep learning," in *International Workshop for Spoken Dialogue Systems*, 2017.

[2] Z Wang, TH Wen, PH Su, and Y Stylianou, "Learning domain-independent dialogue policies via ontology parameterisation," in *16th SIGDial*, 2015, pp. 412–416.

[3] M Gašić, N Mrkšić, PH Su, D Vandyke, TH Wen, and S Young, "Policy committee for adaptation in multi-domain spoken dialogue systems," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015, pp. 806–812.

[4] H Cuayáhuitl, S Yu, A Williamson, and J Carse, "Deep reinforcement learning for multi-domain dialogue systems," *arXiv preprint arXiv:1611.08675*, 2016.

[5] PA Crook, A Marin, V Agarwal, K Aggarwal, T Anastasakos, R Bikkula, D Boies, A Celikyilmaz, S Chandramohan, Z Feizollahi, et al., "Task completion platform: A self-serve multi-domain goal oriented dialogue platform," *NAACL HLT 2016*, p. 47, 2016.

[6] A Papangelis, V Karkaletsis, and F Makedon, "Online complex action learning and user state estimation for adaptive dialogue systems," in *Tools with Artificial Intelligence (ICTAI), IEEE 24th International Conference on*. IEEE, 2012, vol. 1, pp. 642–649.

[7] S Chandramohan and O Pietquin, "User and noise adaptive dialogue management using hybrid system actions," in *Spoken Dialogue Systems for Ambient Environments*, pp. 13–24. Springer, 2010.

[8] 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, pp. 395–429, 2010.

[9] B Peng, X Li, l Li, j Gao, a Celikyilmaz, s Lee, and KF Wong, "Composite task-completion dialogue system via hierarchical deep reinforcement learning," *arXiv preprint arXiv:1704.03084*, 2017.

[10] P Budzianowski, S Ultes, PH Su, N Mrksic, TH Wen, I Casanueva, L Rojas-Barahona, and M Gasic, "Sub-domain modelling for dialogue management with hierarchical reinforcement learning," *18th SIGDial*, 2017.

[11] T Zhao and M Eskenazi, "Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning," *17th SIGDial*, 2016.

[12] S Young, M Gašić, S Keizer, F Mairesse, J Schatzmann, B Thomson, and K Yu, "The hidden information state model: A practical framework for pomdp-based spoken dialogue management," *Computer Speech & Language*, vol. 24, no. 2, pp. 150–174, 2010.

[13] A Papangelis and Y Stylianou, "Multi-domain spoken dialogue systems using domain-independent parameterisation," in *Domain Adaptation for Dialogue Agents*, 2016.

[14] J Schatzmann and S Young, "The hidden agenda user simulation model," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 17, no. 4, pp. 733–747, 2009.

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

[16] T Schaul, J Quan, I Antonoglou, and D Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[17] A Papangelis, P Papadakos, M Kotti, Y Stylianou, Y Tzitzikas, and D Plexousakis, "Ld-sds: Towards an expressive spoken dialogue system based on linked-data," in *Search-Oriented Conversational AI (SCAI)*. ACM, 2017.