Graph Neural Network Policies and Imitation Learning for Multi-Domain Task-Oriented Dialogues

Thibault Cordier[*,1,2], **Tanguy Urvoy**[2], **Fabrice Lefèvre**[1], **Lina M. Rojas-Barahona**[2]
[1]LIA - University of Avignon, Avignon, France
[2]Orange Labs, Lannion, France
thibault.cordier@alumni.univ-avignon.fr
fabrice.lefevre@univ-avignon.fr
{thibault.cordier, linamaria.rojasbarahona, tanguy.urvoy}@orange.com

Abstract

Task-oriented dialogue systems are designed to achieve specific goals while conversing with humans. In practice, they may have to handle simultaneously several domains and tasks. The dialogue manager must therefore be able to take into account domain changes and plan over different domains/tasks in order to deal with multi-domain dialogues. However, learning with reinforcement in such context becomes difficult because the state-action dimension is larger while the reward signal remains scarce. Our experimental results suggest that structured policies based on graph neural networks combined with different degrees of imitation learning can effectively handle multi-domain dialogues. The reported experiments underline the benefit of structured policies over standard policies.

Introduction

Task-oriented dialogue systems are designed to achieve specific goals while conversing with humans. They can help with various tasks in different domains, such as seeking and booking a restaurant or a hotel (Zhu et al., 2020). The conversation's goal is usually modelled as a slot-filling problem. The *dialogue manager* (DM) is the core component of these systems that chooses the dialogue actions according to the context. *Reinforcement learning* (RL) can be used to model the DM, in which case the policy is trained to maximize the probability of satisfying the goal (Gao et al., 2018).

We focus here on the multi-domain multi-task dialogue problem. In practice, real applications like personal assistants or chatbots must deal with multiple tasks: the user may first want to **find** a hotel (first task), then **book** it (second task). Moreover, the tasks may cover several domains: the user may want to find a hotel (first task, first domain), book it (second task, first domain), and then find a restaurant nearby (first task, second domain).

One way of handling this complexity is to rely on a *domain hierarchy* which decomposes the decision-making process; another way is to switch easily from one domain to another by scaling up the policy. Although *structured dialogue policies* can adapt quickly from a domain to another (Chen et al., 2020b), covering multiple domains remains a hard task because it increases the dimensions of the state and action spaces while the reward signal remains sparse. A common technique to circumvent this reward scarcity is to guide the learning by injecting some knowledge through a teacher policy[1].

Our main contribution is to study how structured policies like *graph neural networks* (GNN) combined with some degree of *imitation learning* (IL) can be effective to handle multi-domain dialogues. We provide large scale experiments in a dedicated framework (Zhu et al., 2020) in which we analyze the performance of different types of policies, from multi-domain policy to generic policy, with different levels of imitation learning.

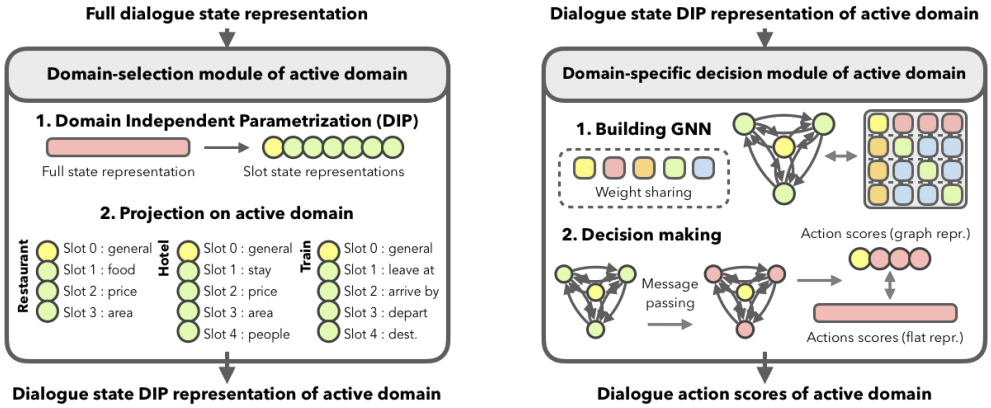
The remainder of this paper is structured as follows. We present the related work in Section 1. Section 2 presents our structured policies combined with imitation learning. The experiments and evaluation are described in Sections 3 and 4 respectively. Finally, we conclude in Section 5.

1 Related Work

Fundamental hierarchical reinforcement learning (Dayan and Hinton, 1993; Parr and Russell, 1998; Sutton et al., 1999; Dietterich, 2000) has inspired a previous string of works on dialogue management (Budzianowski et al., 2017; Casanueva et al., 2018a,b; Chen et al., 2020b). Recently, the use of structured hierarchy with GNN (Zhou et al., 2020; Wu et al., 2020) rather than a set of classical *feed-forward networks* (FNN) enables the learning of non-independent sub-policies (Chen et al., 2018,

[1] For deployment the teacher is expected to be a human expert, however, for experimentation purposes we used the handcrafted policy as a proxy (Casanueva et al., 2017).

Proceedings of the SIGdial 2022 Conference, pages 91–100
Heriot-Watt University, Edinburgh, UK. 07-09, September, 2022



(a) Domain-selection module.

(b) Domain-specific decision module.

Figure 1: GNN policy for multi-domain dialogues with hierarchical decision making and weight sharing.

2020a). These works adopted the *Domain Independent Parametrisation* (DIP) that standardizes the slots representation into a common feature space to eliminate the domain dependence. It allows policies to deal with different slots in the same way. It is therefore possible to build policies that handle a variable number of slots and that transfer to different domains on similar tasks (Wang et al., 2015).

Our contribution differs from Chen et al. (2020b) on three points: first we perform our experiments on ConvLab (Zhu et al., 2020) which is a dedicated multi-domain framework; second, the *dialogue state tracker* (DST) output is not discarded when activating the domain; third, we adapt the GNN structure to each domain by keeping the relevant nodes while sharing the edge's weights.

The reward sparsity can be bypassed by guiding the learning through the injection of some knowledge via a teacher policy. This approach, called *imitation learning* (IL) (Hussein et al., 2017), can be declined from pure *behaviour cloning* (BC) where the agent only learns to mimic its teacher to pure *reinforcement learning* (RL) where no hint is provided (Shah et al., 2016; Hester et al., 2018; Gordon-Hall et al., 2020; Cordier et al., 2020).

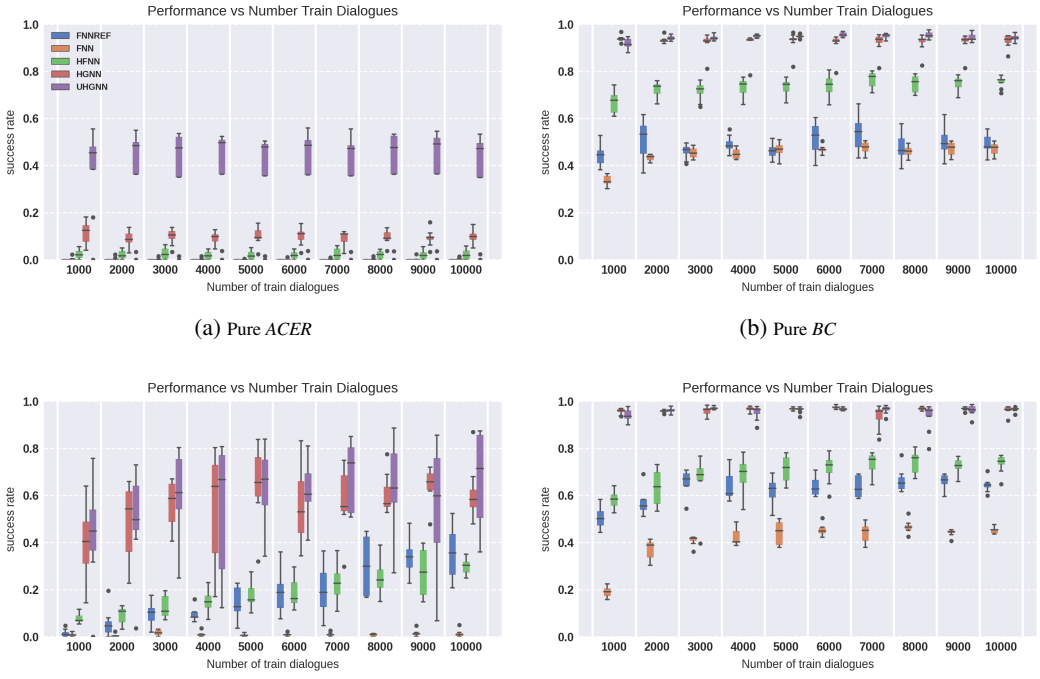
2 Extended GNN Policies with Imitation

We adopt the multi-task setting as presented in ConvLab, in which a single dialogue can have the following tasks: (i) **find**, in which the system requests information in order to query a database and make an offer; (ii) **book**, in which the system requests information in order to book the item. A single dialogue can also contain multiple domains such as *hotel*, *restaurant*, *attraction*, *train*, etc.

Our method, illustrated in Figure 1, is designed to adapt: (i) at the domain-level (*i.e.* be scalable to changes in the number of slots), and (ii) at the multi-domain-level (*i.e.* be scalable to changes of domain). For each dialogue turn, it works as follow: first, the DST module chooses which domain to activate. Then, the multi-domain belief state (and action space) is projected into the active domain (*i.e* only the DIP nodes corresponding to the active domain are kept) as shown in Figure 1a. Afterwards, we apply the GNN message passing as Chen et al. (2020b) but only among the domain specific DIP nodes in the decision making module (Figure 1b).

GNN Policies The GNN structure we consider is a fully connected graph in which the nodes are extracted from the DIP. We distinguish two types of nodes: the slot nodes representing the parametrisation of each slot (denoted as S-node) and the general node representing the parametrisation of the domain (as I-node for slot-Independent node). This yields three types of edges: I2S (for I-node to S-node), S2I and S2S. This abstract structure is a way of modelling the relations between slots as well as exploiting symmetries based on weight sharing (Figure 1b).

Imitation Learning In addition to the structured architecture, we use some level of IL to guide the agent's exploration. In our experiments, we used ConvLab's handcrafted policy as a *teacher* (or *oracle*)[1], but other policies could be used as well. *Behaviour cloning* (BC) is a pure supervised learning method that tries to mimic the teacher policy. Its loss function is the cross-entropy loss as in a classi-



(a) Pure *ACER*

(b) Pure *BC*

(c) *ACER* with *ILfOD*.

(d) *ACER* with *ILfOS*.

Figure 2: Distribution via boxplot of the performance of the proposed approaches on CONVLAB, with 10 different initializations and without pre-training. The coloured area represents the interquartile Q1-Q3 of the distribution, the middle line represents its median (Q2) and the points are outliers.

fication problem. *Imitation Learning From Oracle Demonstrations* (ILFOD) is a RL method which allows the agent to play oracle actions as demonstrations and to inject them in its *replay buffer*. In our experiments, we kept half of the agent's own actions in the buffer along with those generated by the oracle. *Imitation Learning From Oracle Supervision* (ILFOS) is the combination of supervised and reinforcement learning when the agent learns with a supervised loss, namely the margin loss (Hester et al., 2018).

3 Experiments

We performed an ablation study: (i) by progressively extending the baseline to our proposed GNNs and (ii) by guiding the exploration with IL. All the experiments were restarted 10 times with random initialisations and the results evaluated on 500 dialogues were averaged. Each learning trajectory was kept up to 10,000 dialogues with a step of 1,000 dialogues in order to analyse the variability and stability of the methods.

Models The baseline is **ACER** which is a sophisticated actor-critic method (Wang et al., 2016). After an ablation study, we progressively added some notion of hierarchy to FNNs to approximate the structure of GNNs. **FNN** is a feed-forward neural network with DIP parametrisation. Thus, the agent actions are single-actions. **FNN-REF** is a FNN with the native parametrisation (no DIP) with multiple-actions of CONVLAB[2]. **HFNN** is a hierarchical policy with domain-selection module and based on FNNs for each domain. **HGNN** is a hierarchical policy with domain-selection module and based on GNNs. **UHGNN** is a HGNN with a unique GNN for all domains.

Metrics We evaluate the performance of the policies for all tasks. For the find task, we use the precision, the recall and the F-score metrics: the **inform rates**. For the book task, we use the accuracy metric namely the **book rate**. The dialogue is marked as **successful** if and only if both inform's recall and book rate are 1. The dialogue is considered **completed** if it is successful from the user's point of view (*i.e* a dialogue can be completed without being successful if the information provided is not the one objectively expected by the simulator).

[2]The native parametrisation manually groups multi-actions based on MULTIWOZ (Budzianowski et al., 2018).

Configuration			Avg Turn	Inform (%)	Book	Complete		Success	
NLU	Policy	NLG	(succ/all)	Prec. / Rec. / F1	Rate (%)	Rate (%)		Rate (%)	
-	HDC	-	10.6/10.6	87.2 / 98.6 / 90.9	98.6	97.9	-	97.3	-
-	ACGOS (ours)	-	13.1/13.2	94.8 / 99.0 / 96.1	98.7	98.2	(+0.3)	97.0	(-0.3)
BERT	HDC	T	11.4/12.0	82.8 / 94.1 / 86.2	91.5	92.7	-	83.8	-
BERT	HDC†	T	11.6/12.3	79.7 / 92.6 / 83.5	91.1	90.5	(-2.2)	81.3	(-2.5)
BERT	MLE†	T	12.1/24.1	62.8 / 69.8 / 62.9	17.6	42.7	(-50.0)	35.9	(-47.9)
BERT	PG†	T	11.0/25.3	57.4 / 63.7 / 56.9	17.4	37.4	(-55.3)	31.7	(-52.1)
BERT	GDPL†	T	11.5/21.3	64.5 / 73.8 / 65.6	20.1	49.4	(-43.3)	38.4	(-45.4)
BERT	PPO†	T	13.1/17.8	69.4 / 85.8 / 74.1	86.6	75.5	(-17.2)	71.7	(-12.1)
BERT	ACGOS (ours)	T	14.0/14.8	88.8 / 92.6 / 89.5	86.6	89.1	(-3.6)	81.7	(-2.1)

Table 1: Dialogue system evaluation with simulated users. T means template-based NLG. Configurations without NLU and NLG modules pass directly the dialogue act. Configurations with ACGOS and HDC policies are evaluated on a single run with 1,000 dialogues. Configurations with † are taken from the GitHub of ConvLab. PPO in ConvLab used behaviour cloning as the pre-trained weights (see for more details).

4 Evaluation

We evaluate the dialogue manager and the dialogue system both with simulated users.

Dialogue Manager We performed an ablation study based on ACER as reported in Figure 2. First, all RL variants of ACER (Figure 2a) have difficulties to learn without supervision in contrast to BC variants (Figure 2b). In particular, we see that hierarchical decision making networks (HFNN in green), graph neural network (HGNN in red) and generic policy (UHGNN in purple) drastically improve the performance compared to FNNs. Similarly, using IL like ILFOD (Figure 2c) and ILFOS (Figure 2d) notably improves the performance. Therefore, learning generic GNNs allows collaborative gradient update and efficient learning on multi-domain dialogues. Conversely, we observe that hierarchical decision making with HFNNs does not systematically guarantee any improvement. These results suggest that GNNs are useful for learning dialogue policies on multi-domain which can be transferred during learning across domains on-the-fly to improve performance. Finally, regarding ILFOD variants (Figure 2c), we can observe that all architectures are affected by a large variability. This shows that multi-domain dialogue management is difficult despite the use of demonstrations and that learning with reward is not sufficient to robustly succeed.

Dialogue System We evaluate the policy learning algorithms in the entire dialogue pipeline, in particular our best DM policy ACER-ILFOS-UHGNN under a shorter name **ACGOS**. The results of our experimentation are presented in Table 1. We observe that the performance of our approach is closed to the handcrafted policy (the teacher) when directly passing the dialogue acts (97.3 *vs.* 97.0). It is also closed to the handcrafted policy when using BERT NLU (Devlin et al., 2018) and template-based NLG (83.8 *vs.* 81.7). It is much better compared to the baselines with a significant difference (*e.g.* with 81.7 for ACGOS *vs.* 71.7 for pre-trained PPO). These results highlight the benefit of structured policies against standard policies.

5 Conclusion

We studied structured policies like GNN combined with some imitation learning that effectively handle multi-domain dialogues. The results of our large-scale experiments on ConvLab confirm that an actor-critic based policy with a GNN structure can solve multi-domain multi-task dialogue problems. Finally, we evaluated our best policy (ACGOS) in a complete dialogue system with simulated users. It overcomes the baselines and it is comparable to the handcrafted policy.

A limitation of current policies in ConvLab, including ours, is that the robustness to noisy inputs is not specifically addressed as it had been done in PyDial (Ultes et al., 2017). It could be also interesting to study the impact of incorporating real human feed-backs and demonstrations instead of a handcrafted teacher.

The GNN structured policies combined with imitation learning avoid sparsity, while being data efficient, stable and adaptable. They are relevant for covering multi-domain task dialogue problems.

References

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Paweł Budzianowski, Stefan Ultes, Pei-Hao Su, Nikola Mrkšić, Tsung-Hsien Wen, Inigo Casanueva, Lina Rojas-Barahona, and Milica Gašić. 2017. Sub-domain modelling for dialogue management with hierarchical reinforcement learning. *arXiv preprint arXiv:1706.06210*.

Pawel Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *EMNLP*.

Iñigo Casanueva, Paweł Budzianowski, Pei-Hao Su, Stefan Ultes, Lina M Rojas Barahona, Bo-Hsiang Tseng, and Milica Gasic. 2018a. Feudal reinforcement learning for dialogue management in large domains. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 714–719.

Iñigo Casanueva, Paweł Budzianowski, Stefan Ultes, Florian Kreyssig, Bo-Hsiang Tseng, Yen-Chen Wu, and Milica Gasic. 2018b. Feudal dialogue management with jointly learned feature extractors. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 332–337.

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:1711.11023 [cs, stat]*. ArXiv: 1711.11023.

Lu Chen, Bowen Tan, Sishan Long, and Kai Yu. 2018. Structured dialogue policy with graph neural networks. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1257–1268.

Zhi Chen, Lu Chen, Xiaoyuan Liu, and Kai Yu. 2020a. Distributed structured actor-critic reinforcement learning for universal dialogue management. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2400–2411.

Zhi Chen, Xiaoyuan Liu, Lu Chen, and Kai Yu. 2020b. Structured hierarchical dialogue policy with graph neural networks. *arXiv preprint arXiv:2009.10355*.

Thibault Cordier, Tanguy Urvoy, Lina M Rojas-Barahona, and Fabrice Lefèvre. 2020. Diluted near-optimal expert demonstrations for guiding dialogue stochastic policy optimisation. In *Human in the loop dialogue systems Workshop at 34th Conference on Neural Information Processing Systems*.

Peter Dayan and Geoffrey E Hinton. 1993. Feudal reinforcement learning. In *Advances in neural information processing systems*, pages 271—-278.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Thomas G Dietterich. 2000. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 13:227–303.

Jianfeng Gao, Michel Galley, and Lihong Li. 2018. Neural approaches to conversational ai. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1371–1374.

Gabriel Gordon-Hall, Philip Gorinski, and Shay B Cohen. 2020. Learning dialog policies from weak demonstrations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1394–1405.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. 2018. Deep q-learning from demonstrations. In *Thirty-second AAAI conference on artificial intelligence*.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. 2017. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35.

Ronald Parr and Stuart Russell. 1998. Reinforcement learning with hierarchies of machines. *Advances in neural information processing systems*, pages 1043–1049.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pararth Shah, Dilek Hakkani-Tur, and Larry Heck. 2016. Interactive reinforcement learning for task-oriented dialogue management.

Richard S Sutton, Doina Precup, and Satinder Singh. 1999. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211.

Stefan Ultes, Lina M Rojas Barahona, Pei-Hao Su, David Vandyke, Dongho Kim, Iñigo Casanueva, Paweł Budzianowski, Nikola Mrkšić, Tsung-Hsien

Wen, and Milica Gasic. 2017. Pydial: A multi-domain statistical dialogue system toolkit. In *Proceedings of ACL 2017, System Demonstrations*, pages 73–78.

Zhuoran Wang, Tsung-Hsien Wen, Pei-Hao Su, and Yannis Stylianou. 2015. Learning domain-independent dialogue policies via ontology parameterisation. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 412–416.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. 2016. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81.

Qi Zhu, Zheng Zhang, Yan Fang, Xiang Li, Ryuichi Takanobu, Jinchao Li, Baolin Peng, Jianfeng Gao, Xiaoyan Zhu, and Minlie Huang. 2020. Convlab-2: An open-source toolkit for building, evaluating, and diagnosing dialogue systems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 142–149.

A Appendix

A.1 Domains

Domain	# constraint slots	# request slots
CONVLAB	find/book	search
Restaurant	4/3	5
Attraction	3/-	7
Hotel	7/3	5
Taxi	4/-	2
Train	5/1	5
Hospital	1/-	3
Police	-/-	3

Table 2: Domains Description of CONVLAB framework

Belief State The belief state representation is deterministic. As shown in Figure 3, there is no uncertainty (all values are either 0's or 1's).

State Space The input to the dialogue manager is the belief state which is a dictionary of all tractable information (slot-value pairs, history, dialogue actions of system and user, etc.). This is called the *master state space*. And, due to its large size, the representation is projected into the *summary state space* by a process called *value abstraction* (Wang et al., 2015). Finally, it must be vectorised in order to be interpretable by neural networks.

Action Space The dialogue manager's output is a probabilistic distribution over all possible actions. To reduce the complexity of the learning problem, *master actions*, which are valued dialogue acts such as INFORM(date = '2022-01-15'), are abstracted into *summary actions* like INFORM(date), the *value abstraction* module being in charge of restoring the relevant values in the context. On CONVLAB the policy may activate several actions simultaneously (called *multiple-actions*).

Domain Independent Parametrisation (or DIP) (Wang et al., 2015) standardises the slots representation into a common feature space to eliminate the domain dependence. In particular, the DIP state and action representations are not reduced to a flat vector but to a set of sub-vectors: one corresponding to the domain parametrisation (called *slot-independent representation*), the others to the slots parametrisation (called *slot-dependent representations*).

Component / Description
Beliefs
constraint slot beliefs: $\{b_{d,s}^{inf} \in \mathcal{V}_s, \forall s \in \mathcal{S}_d^{inf}, \forall d \in \mathcal{D}\}$ The goal constraints belief for each informable slot. This is either an assignment of a value from the ontology which the user has specified as a constraint, or has a special value — either *dontcare* which means the user has no preference, or *none* which means the user is yet to specify a valid goal for this slot. To be exact, for each domain, the constraint slot dictionary separates slots with respect to the task i.e we distinguish the *find* slot dictionary and the *book* slot dictionary.
request slot beliefs: $\{b_{d,s}^{req} \in \mathbb{B}, \forall s \in \mathcal{S}_d^{req}, \forall d \in \mathcal{D}\}$: A set of requested slots, i.e. those slots whose values have been requested by the user, and should be informed by the system.
Features
terminated: $f_1 \in \mathbb{B}$: A boolean showing that the user wants to end the call.
booked: $f_2 \in \mathcal{V}_{DB(d)}$: The name of the last venue offered by the system to the user with respect to the constraint slots with additional information like reference. To be exact, this feature is located in the *book* slot dictionary.
degree pointer: $f_3 \in \mathbb{B}^6$: The vector counting the number of entities *count* matching with constraint slots in acceptance list: [count==0, count==1, count==2, count==3, count==4, count>=5].
System Acts
system acts: $a^{sys} \in list(\mathcal{A}^{sys})$: The list of the last system actions.
User Acts
user acts: $a^{user} \in list(\mathcal{A}^{user})$: The list of the last user actions.

Table 3: Belief State Template in ConvLab framework

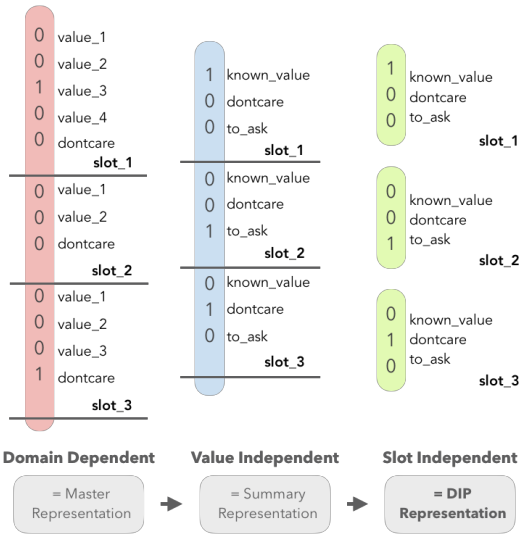


Figure 3: Transformation from initial state to DIP state representation (it works similarly for actions).

A.2 State and Action Representations

We propose to formally present the state representations used in our experiments. For details about our notations, see Table 3.

Flat state representation in ConvLab

$$\phi(x) = \left(\bigoplus_{s \in \mathcal{S}^{inf}} b_s^{inf} \right) \oplus a^{user} \oplus a^{sys} \oplus [f_1] \oplus f_2 \oplus f_3$$

where x is the initial state, $\phi(x)$ is the full state parametrisation, $\mathcal{S}^{inf}$ is the set of informable slots, b_s^{inf} is the one-encoding vector of the informable slot s, a^{user} and a^{sys} are the one-encoding vectors of previous user and system actions, f_1 is the boolean "terminated dialogue", f_2 is the boolean "booked offer" with respect to each domain, f_3 is the one-encoding vector of the matching entities count with respect to each domain and $\oplus$ is the vector concatenation operator.

DIP state representation

Slot independent parametrisation:

$$\phi_d(x) = a^{user}|_g \oplus a^{sys}|_g \oplus [f_1,\ f_2|_d,\ f_3|_d]$$

where x is the initial state, $\phi_d(x)$ is the active domain state parametrisation, $a^{user}|_g$ and $a^{sys}|_g$ are the one-encoding vectors of previous general user and system actions, f_1 is the boolean "terminated dialogue", $f_2|_d$ is the boolean "booked offer" with respect to the active domain, $f_3|_d$ is the one-encoding vector of the matching entities count with respect to the active domain and $\oplus$ is the vector concatenation operator.

Slot dependent parametrisation:

$$\forall s_i \in \mathcal{S}_d,\ \phi_{s_i}(x) = a^{user}|_{s_i} \oplus a^{sys}|_{s_i}$$
$$\oplus \left[\mathbb{1}(\exists\, v \in \mathcal{V}_{s_i}/\{\text{none}\},\ b^*_{s_i}[v] = 1)\right] \quad (2a)$$
$$\oplus \left[\mathbb{1}(s_i \in \mathcal{S}_d^{inf})\right] \quad (2b)$$
$$\oplus \left[\mathbb{1}(s_i \in \mathcal{S}_d^{req})\right] \quad (2c)$$

where x is the initial state, $\phi_{s_i}(x)$ is the slot parametrisation of the i^{th} slot, $\mathcal{S}_d$ is the set of slots of the active domain, $a^{user}|_{s_i}$ and $a^{sys}|_{s_i}$ are the one-encoding vectors of previous user and system actions of the i^{th} slot, (2a) is the indicator of known value, (2b) is the indicator of informable slot and (2c) is the indicator of requestable slot and $\oplus$ is the vector concatenation operator.

A.3 Implementation Details

Imitation learning The used oracle is the hand-crafted agent proposed by each framework. When we use ILFOD or ILFOS methods, 50% of the time the oracle trajectories is used. When we use ILFOS, we call also in 100% of the time the oracle which gives us the best expert action as supervision and a margin penalty $\mu = \log(2)$ (Hester et al., 2018).

Reinforcement learning Our policy algorithm is an off-policy learning that uses experience replay (all data are stored in buffers) without priority *i.e* without importance sampling. The exploitation-exploration procedure is achieved by Boltzmann sampling with a fixed temperature $\tau = 1$.

Metrics and Rewards **Inform recall** evaluates whether all the requested information has been informed when **inform precision** evaluates whether only the requested information has been informed. **Book rate** assesses whether the offered entity meets all the constraints specified in the user goal. The system is guided by the rewards as follows. If all domains are solved (a domain is solved if all related tasks are solved), it gains 40 points. If the current active domain is solved, it gains 5 points. Otherwise, it is penalised by 1 point.

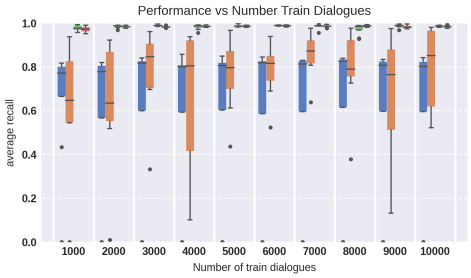
Model setup for neural network architectures Our FNN models have two hidden layers, both with 128 neurons. Our GNN models have one first hidden layer with 32 neurons for each node (two in all: S-NODE and I-NODE). Then the second hidden layer is composed of 32 neurons for each relation (three in all: S2S, S2I and I2S). The size of the tested networks are of the order of magnitude of $10\,000$ to more than $100\,000$ parameters.

For learning stage, we use a learning rate $\mathrm{lr} = 10^{-3}$, a dropout rate $\mathrm{dr} = 0.1$ and a batch size $\mathrm{bs} = 64$. Each loss function has a weight of $\lambda_Q = 0.5$, $\lambda_\pi = 1.$, $\lambda_{IL} = 1.$ and $\lambda_{ent} = 0.01$ respectively. The learning frequency is one iteration after each episode (finished dialogue) with only one gradient iteration.

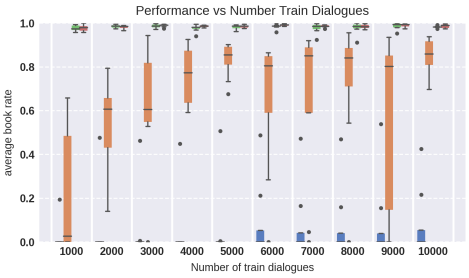
Used packages for the experiment We used the dialogue system frameworks named CONVLAB (Zhu et al., 2020). For the implementation of neural networks, we used PYTORCH (Paszke et al., 2019) in our dialogue systems. We also used another toolkit for reinforcement learning research named OPENAI GYM (Brockman et al., 2016).

A.4 Supplementary Results

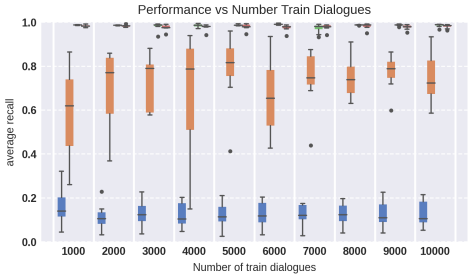
We propose to present supplementary results of our ablation study. We show the distribution (via boxplot) of different measures with 10 different initialisations and without pre-training. In particular, Figure 4 presents the distribution of inform recall, Figure 5 the distribution of book rate, Figure 6 the distribution of success rate and Figure 7 the distribution of cumulative rewards. We precise that the coloured area represents the interquartile Q1-Q3 of the distribution, the middle line represents its median (Q2) and the points are outliers.



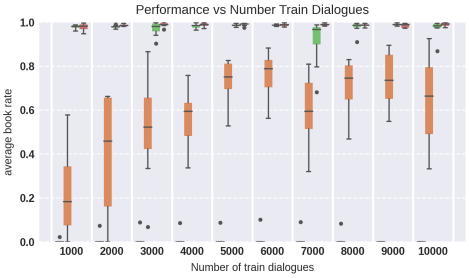
(a) Recall Average - UHGNN models



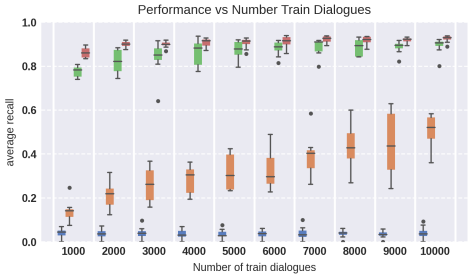
(a) Book Rate - UHGNN models



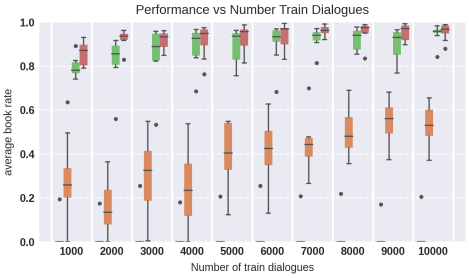
(b) Recall Average - HGNN models



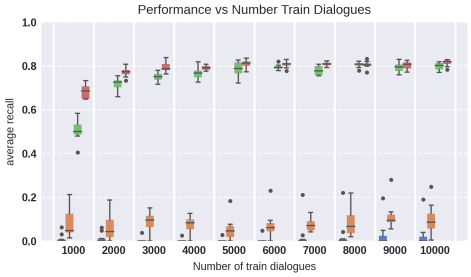
(b) Book Rate - HGNN models



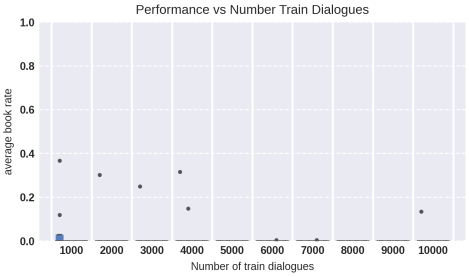
(c) Recall Average - HFNN models



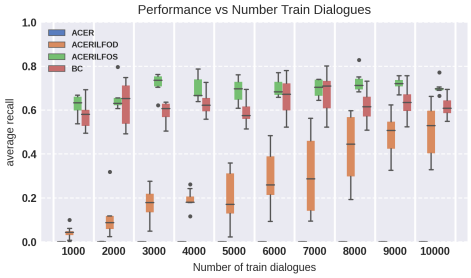
(c) Book Rate - HFNN models



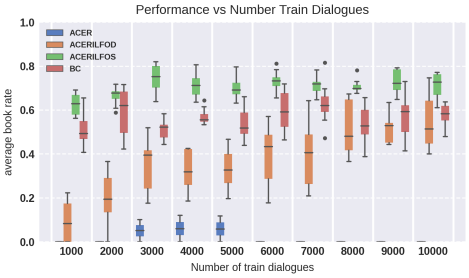
(d) Recall Average - FNN models with DIP parametrization



(d) Book Rate - FNN models with DIP parametrization



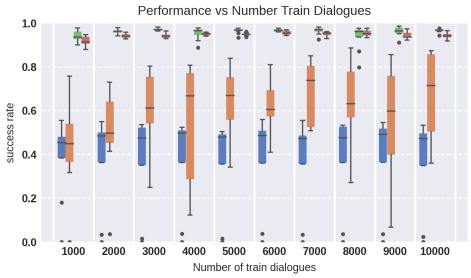
(e) Recall Average - FNN models with native parametrization



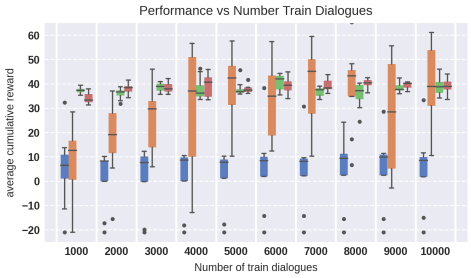
(e) Book Rate - FNN models with native parametrization

Figure 4: Summary of performance - Task *find*

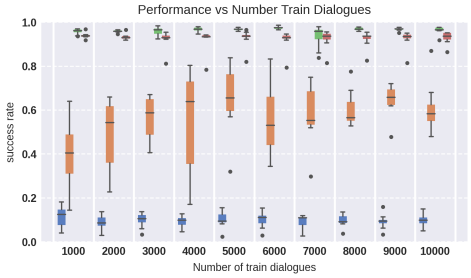
Figure 5: Summary of performance - Task *book*



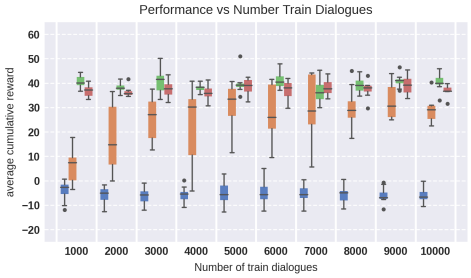
(a) Success Rate - UHGNN models



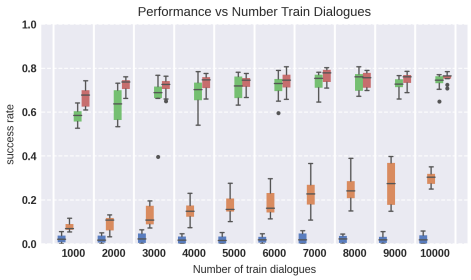
(a) Cumulative rewards - UHGNN models



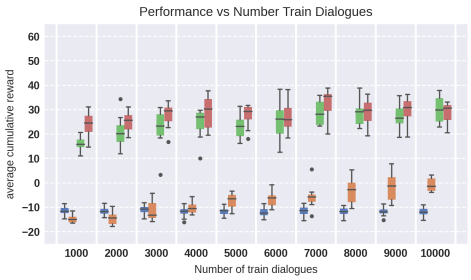
(b) Success Rate - HGNN models



(b) Cumulative rewards - HGNN models

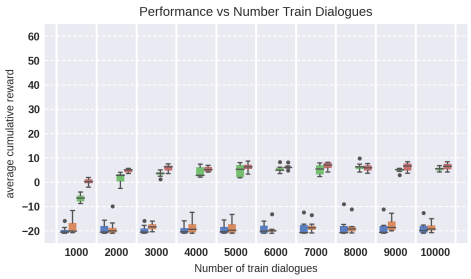


(c) Success Rate - HFNN models

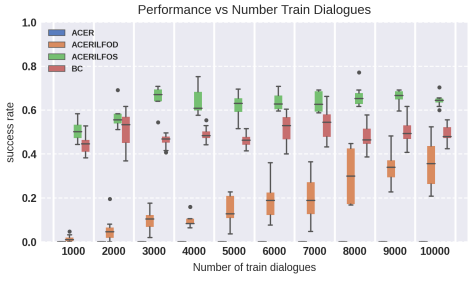


(c) Cumulative rewards - HFNN models

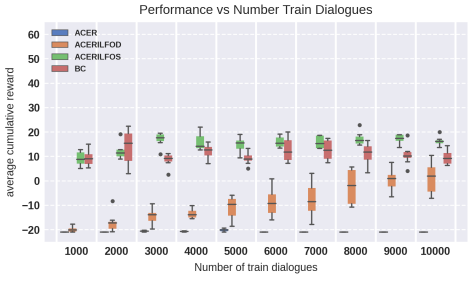
(d) Success Rate - FNN models with DIP parametrization



(d) Cumulative rewards - FNN models with DIP parametrization



(e) Success Rate - FNN models with native parametrization



(e) Cumulative rewards - FNN models with native parametrization

Figure 6: Summary of performance - Global task (Task *find* and/or Task *book*)

Figure 7: Summary of performance - Cumulative rewards