COMPARISON OF TRAINING BEHAVIOR AND PERFORMANCE OF REINFORCEMENT LEARNING BASED POLICIES FOR DIALOGUE MANAGEMENT

Stefan Hillmann[1], *Tilo Himmelsbach*[1], *Benjamin Weiss*[2]

[1]*Technische Universität Berlin,* [2]*Hanyang Universität Seoul*
stefan.hillmann@tu-berlin.de

Abstract: We present the results of a laborious comparison of four different reinforcement learning algorithms that are used to train policies for dialogue management. We have trained 32 policies by varying the concept error rate, the number of user dialogue acts, and the number of training dialogues. Data about the training behavior and performance of the trained policy in the evaluation are presented. Actor-critic leads to very good task success rates and notable shorter dialogues among the evaluated algorithms (actor-critic, REINFORCE, Q-Learning, and WoLF-PHC).

1 Introduction

Traditionally, rule-based policies are used for dialogue management. Their development is a manual labour-intensive and inflexible process that can lead to narrow solutions that only fit particular conditions and domains and is difficult to maintain. To overcome these drawbacks, the policy-development should not be regarded as an engineering problem but as a data-driven optimisation problem [1]. By employing methods from the field of reinforcement learning (RL), the policies can be learned by agents and their behavior controlled by a reward-function. Concerning this, Casanueva et al. [2] list further RL-methods that are applied to the training of dialogue managers.

There is a large number of (different kinds) of RL algorithms available today; Sutton and Barto [3] give a representative overview. Approaches on using RL for training of policies for dialogue managers are described and discussed in valuable previous work, e.g. [4, 2]. In general, when applied to the training of dialogue managers, for each dialogue state, the agent tries to maximise its cumulative future reward by picking from a set of predefined actions. Since the number of possible dialogue states can be huge, universal function approximators like artificial neural networks (ANN) have been used as policy agents [5].

When using ANN to learn dialogue policies, a common approach is the n-hot encoding of dialogue states. Such an encoding is rather easy to learn by an ANN but has the drawback that the dimension and structure of the ANN's input layer depend on the features representing the dialogue state.Dialogue state encodings that allow general models for policy learning, however, are beneficial, as they allow a change in the set of dialogue state features without modifying the architecture of the model representing the policy.

Benchmarking of different RL algorithms is presented in [6] for control tasks and for the training of dialogue policies in [2]. Each of the two works presents a framework for benchmarking of RL algorithms, but the tasks of former are not related to dialogue systems, and the latter is strongly anchored in PyDial [7]. Besides the chosen RL algorithm, the representation, i.e. the used features, of the dialogue state have a significant influence on the final performance of a trained policy. For that reason, it is much more efficient for us to do evaluations directly

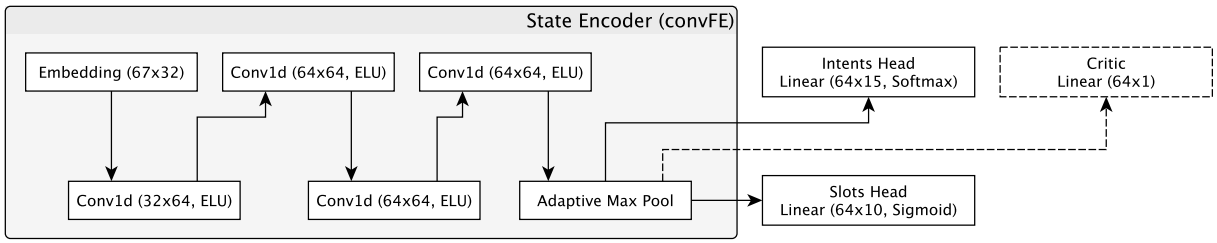


Figure 1 – Scheme of the architecture used for REINFORCE and A2C in combination with convFE for state encoding. The kernel size of all convolutional parts (Conv1d) is 3. Intents Head, Slots Head and Critic are each a single layer. Critic is exclusively used for A2C.

in the framework we have used to implement, which we use for our research. It would be valuable to find a benchmark approach that makes it easier to compare not only RL algorithms but additional parts of the training environment. However, this issue is not covered by our paper.

In this paper, we provide two main contributions. First, we present an elaborated study about the training behavior and the performance of trained dialogue manager policies of four RL algorithms. Two of the algorithms, Q-Learning and WoLF-PHC, are simple in means of conception and computational efforts, which makes them a good baseline. The other two, A2C and REINFORCE, are based on artificial neural networks (ANN) and are therefore more complex in their concepts and with regard to the necessary computations during the training. Our study examines the influence of the number of training dialogues, errors by a natural language understanding component (i.e., concept error rate), and the number of dialogue acts sent by a (simulated) user.

Second, we present an approach that allows a general dialogue state encoding. It combines a string-representation of a dialogue state with a convFE approach [8] as an input network for the ANN-based RL-algorithms. Thus, the same dialogue state representation can be used as input for all four algorithms.

In the following, Section 2 gives an introduction to the implemented dialogue state encoding and the evaluated RL algorithms as well as the training and evaluation setting. The results of the learning behavior during training (i.e., development of achieved reward) and the performance of the learned policy are presented in Section 3 and will be discussed in Section 4.

2 Method

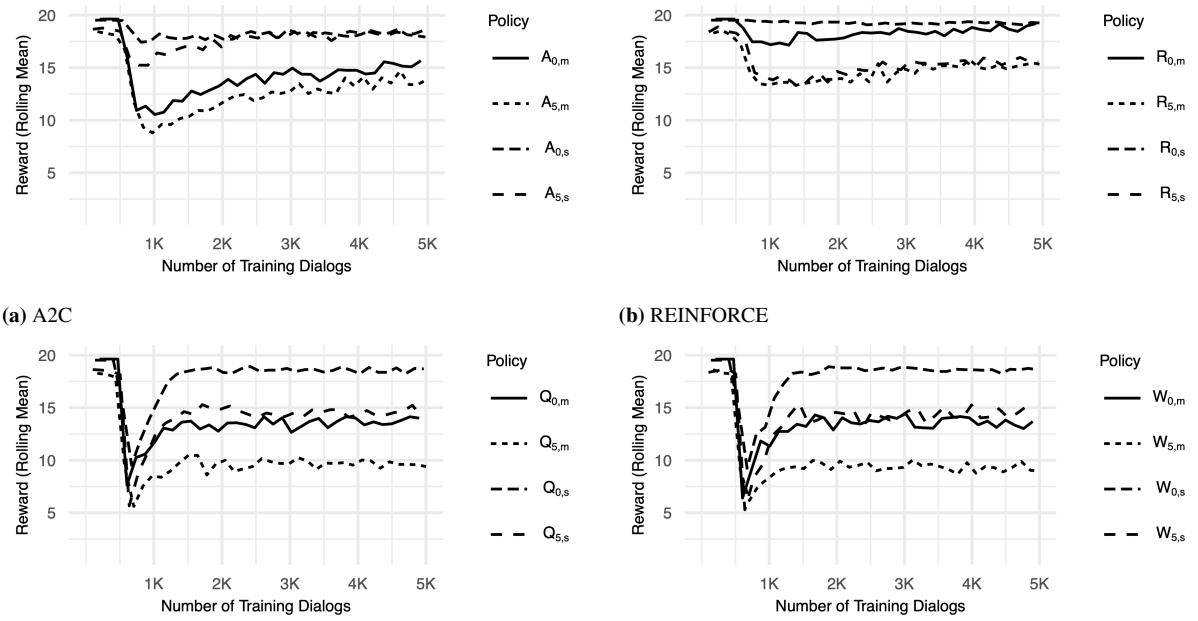
The training and the generation of dialogues for evaluation are carried out with the PLATO Research Dialogue System [9]. The specific domain and the dialogue manager to be trained as well as methods for training and evaluation are described in this section.

2.1 Dialogue Domain

Our application domain is the electronic course module and course catalogue of the TU Berlin, for which a chatbot named Alex has already been implemented [10]. Students can use the chatbot to search for courses that match their requirements regarding the subject, study program, and properties like lectures' times and places. We used a subset of the Alex database with more than 27,205 available course offers.

2.2 Interaction

As we are only interested in the training of the dialogue manager and to avoid the issue of controlling the influence of the natural language understanding and natural language generation components, we restrict the agent and user simulation to interact via dialogue acts solely, not



(a) A2C

(b) REINFORCE

(c) Q-Learning

(d) WoLF-PHC

Figure 2 – Development of reward (R) in the *5K* setting (500 warm-up dialogues, in total 5,000 training dialogues). The diagrams show values for rolling mean (windows size is 100, right aligned). The assignment of labels to the conditions is given in Table 2

natural language. The user simulation used in PLATO is implemented according to the agenda-based user simulation (AS) described by Schatzmann et al. [11, 9]. The AS's agenda is not observable for the system agent. Thus, this setting can be interpreted as a "partially observable Markov decision process" (POMDP) [5][1].

The AS implemented in PLATO provides a concept error simulation, which allows for the swapping of concepts or the value of a concept in the dialogue acts sent by the AS, in order to mimic errors that usually occur in automatic speech recognition and language understanding. Both error rates were varied in our study to probabilities of 0 and 5 % (see Sec. 2.5 for details).

2.3 Dialogue State Encoding

Table 1 shows the features of a dialogue and the most recent turn in a dialogue that are used to define a dialogue state. The interaction between system and user (real and simulated) is modeled as an exchange of dialogue acts. A dialogue act consists of two parts, an intent and a list of attribute-value-pairs (AVP). An AVP represents the assignment of a particular value to a concept or system slot. Dialogue acts used by system and user in a specific turn are part of the dialogue state. However, when encoding dialogue states for the training or action query on/with a policy the value assignments are not encoded. The aim is to train a more general policy which estimates feasible system actions, i.e. system dialogue acts, independently from concrete, e.g. task- or database-specific, value assignments.

In the study presented here, we used a sub-set of the dialogue act types (intents) which are pre-defined in PLATO and again base on the intents defined for the Dialog State Tracking Challenge 2 [12]. The used intents are *welcome*, *bye*, *affirm*, *deny*, *confirm*, *inform*, and *request*. The latter three are used in conjunction with AVPs.

2.4 Training Algorithms

Q-Learning [13] and WoLF-PHC [14] are based on PLATO's implementations; major changes were done to the dialogue-state encoding. **WoLF-PHC** is based on Q-Learning but uses a

Table 1 – Features that represent a dialogue state in dialogue *d*. Here, *t* refers the last previous turn, i.e., the turn that led to the current state (*s*).

Feature	Description	Type
Terminal state	*true* if *d* ends in *s*.	bool
Item in focus	*true* the *DM* has a potential solution (i.e. found an matching item in the database)	bool
Turn number	Number of turns in *d* so far.	int
State value*	Reward estimated by the critic	int
DB ratio	Ratio between number of matching items and items in the DB	int
Filled slots	Names of all filled slots.	string
Confirmed Slots	Names of all slots which were already confirmed by the user.	string
Requested slots	Name of all slots which were already requested by the user.	string
User acts	Names of the user intents and related slots in *t*.	string
System offered	*true* if the system has offered a solution (i.e., an item from the database) in *t*.	bool
User affirmed	*true* if the user has used an *affirm*-act in *t*	bool
User denied	*true* if the user has used a *deny*-act in *t*	bool
System acts	Names of the system intents and related slots in *t*.	string

*only with actor-critic

dynamic learning rate to achieve better convergence of the matrix during the training. For these two algorithms, the encoded state is converted in an ordered way to a text string whose hash serves as a key of the key-value representation of the q-matrix and the π-matrix for WoLF-PHC.

For the two ANN-based approaches described below, the string-representation of the encoded state is tokenized and fed into a four-layered convolutional neural network (see State Encoder in Fig. 1), which serves as a learnable feature extractor (convFE). The architecture is based on a document classifier proposed by [8].

Since we did not manage to generate plausible learning behavior by using the PLATO implementation of the **REINFORCE** algorithm [15], we developed our own implementation based on the PyTorch framework. Hereby we were inspired by PyTorch's example implementation[1].

The policy agent's neural network consists of the convFE architecture extended by two linear network layers that predict intent and slot probabilities and are trained end-to-end together with the convFE network. Fig. 1 shows the network architecture, which is equal for REINFORCE and A2C expcept the Critic.

The prediction layers are called the Actor Head which itself consists of two prediction heads. First, an *Intents Head* which predicts probabilities from which a categorical distribution draws a single intent per forward-pass. Second, a *Slots Head* which predicts logits which serve as a binomial distribution for sampling a varying number of slots per forward-pass from 0 up to the number of available system slots.

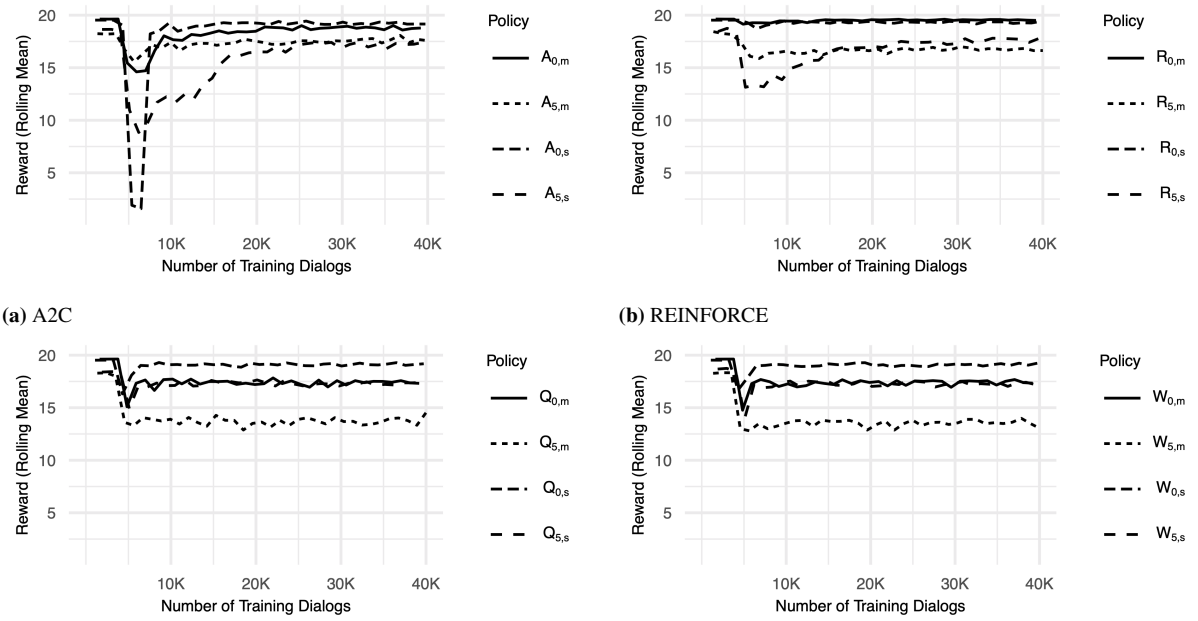
By adding a state-value estimating prediction head (i.e., Critic in Fig. 1), that serves as a critic, to the architecture described above, the **actor-critic** neural architecture is formed. The Critic consists of a single linear layer of output-dimension one, which is trained in a square-error-minimising regression setting. Architecture and training follow the advantage actor-critic (A2C) methodology described by [16].

2.5 Training

All four algorithms are trained with a batch-size of 8 dialogues for ten epochs every 8th simulated dialogue. Furthermore, we use an initial warm-up period in which agents are trained on dialogues generated by a rule-based heuristic.

Depending on the success or failure of the dialogue, a positive or negative reward of +20 or -1 is counted. For each dialogue step (turn), the agent receives a penalty of -0.05. The dialogue

[1] `https://github.com/pytorch/examples/blob/master/reinforcement_learning/reinforce.py`



(a) A2C

(b) REINFORCE

(c) Q-Learning

(d) WoLF-PHC

Figure 3 – Development of reward (R) in the *40K* setting (4,000 warm-up dialogues, in total 40,000 training dialogues). The diagrams show values for rolling mean (windows size is 1,000, right aligned). The assignment of labels to the conditions is given in Table 2

is regarded as successful if the solution offered by the system (in our data a module from the course catalog) meets all requirements set by the user, and all information requested by the user regarding this solution have been correctly answered by the system.

As we are interested in studying the learning behavior of the algorithms as well as the performance of the resulting policies, we have varied two conditions with each algorithm, resulting in the combinations that are shown in Table 2 and explained in the following. First, the number of dialogue acts (*NUDA*) that can be sent by the user simulator in a user turn. That is either a single act (*NUDA* = s in Table 2) vs. the usage of a single or two acts with equal probability in each turn (*NUDA* = m). Second, the error rate (*ER*) used in the AS's concept error simulation. The error rate was 0 % (meaning no concept errors, *ER* = 0 in Table 2) vs. 5 % (*ER* = 5).

Furthermore, each policy was trained, using the setting described before, with 5,000 (*5K*) and 40,000 (*40K*) training dialogues to study the effect on the performance of the trained policy as well as the development of the mean reward during the training. In the *5K* condition the training started with 500 warm-up dialogues and in the *40K* condition with 4,000. The warm-up dialogues are counted within the 5,000 and 40,000 training dialogues.

In the *5K* condition for each setting, a policy was trained and evaluated for five times as we saw a noteworthy variance in the evaluation results. In the *40K* condition, we noticed only a very small variance and decided to train and evaluate a policy only once for each setting, not least because of the high computational efforts for one training.

2.6 Evaluation

For the evaluation of the trained policies, a data set of 1,000 dialogues was generated with each of them and the policies behavior are compared in terms of average reward (R), average dialogue length (L) measured by the number of turns, and task success (S). In the evaluation, each policy was used with the same conditions (*NUDA* and *ER*) as in the respective training.

Table 2 – Evaluation results for 1,000 dialogues being generated in the *5K* and *40K* condition.

				5K						*40K*		
ID	Policy	*ER*	*NUDA*	$\bar{L}$	σ_L	$\bar{R}$	σ_R	$\bar{S}$ *(%)*	σ_S	$\bar{L}$	$\bar{R}$	*S (%)*
$Q_{0,s}$	Q-Policy	0	s	10.65	0.08	18.81	0.26	96.68	1.23	10.53	19.35	99.2
$Q_{0,m}$	Q-Policy	0	m	9.84	0.12	11.22	0.28	60.80	1.31	8.72	17.66	90.8
$Q_{5,s}$	Q-Policy	5	s	13.61	0.20	13.98	0.56	75.40	2.63	12.99	17.40	91.3
$Q_{5,m}$	Q-Policy	5	m	11.51	0.14	5.82	0.44	36.20	2.00	11.17	12.68	68.5
$W_{0,s}$	WoLF-PHC	0	s	10.67	0.10	18.83	0.17	96.78	0.77	10.50	19.46	99.7
$W_{0,m}$	WoLF-PHC	0	m	9.99	0.18	10.75	0.38	58.64	1.80	9.01	16.88	87.2
$W_{5,s}$	WoLF-PHC	5	s	13.55	0.14	14.01	0.49	75.48	2.25	13.10	17.16	90.2
$W_{5,m}$	WoLF-PHC	5	m	11.68	0.07	5.78	0.46	36.10	2.12	11.20	11.83	64.5
$R_{0,s}$	REINFORCE	0	s	11.65	0.36	19.22	0.28	98.80	1.24	11.10	19.31	99.1
$R_{0,m}$	REINFORCE	0	m	8.71	0.28	18.56	0.39	94.97	1.80	8.42	19.54	99.6
$R_{5,s}$	REINFORCE	5	s	13.93	0.41	15.88	0.64	84.38	2.75	12.84	18.00	93.7
$R_{5,m}$	REINFORCE	5	m	11.27	0.34	15.33	0.29	81.10	1.23	10.42	17.03	88.8
$A_{0,s}$	A2C	0	s	8.17	0.80	18.37	1.38	93.96	6.61	7.32	19.28	98.1
$A_{0,m}$	A2C	0	m	7.70	0.67	14.99	1.63	77.80	7.63	6.26	18.60	94.6
$A_{5,s}$	A2C	5	s	8.34	0.21	18.49	0.79	94.58	3.79	8.36	18.45	94.4
$A_{5,m}$	A2C	5	m	9.57	1.77	13.80	1.27	72.60	5.67	6.90	17.96	91.7

3 Results

In the following, we present and describe the results regarding the training behavior of the four different algorithms under the different training conditions (see Section 2.5) as well as the evaluation results of the trained policies in Section 3.2.

3.1 Training

In this paper, our major interest is about how fast a policy learns and how much of the possible reward a policy can gain during the training. Fig. 2 shows the development of reward for the four algorithms under the different settings for *NUDA* and *ER*. The shown reward for a certain condition and algorithm at a certain point in the training is the rolling mean of the rewards of the last 100 dialogues. As we trained five times with each setting in the *5K*-condition, the plots show the average rolling mean of the five trials for the n-th training dialogue. The plot shows that Q-Learning and WoLF-PHC have a substantial drop in the reward after the warm-up and the that both algorithms can achieve less reward than A2C and REINFORCE in the condition with multiple user intents (*NUDA* = s) and *ER* = 5 %. Interestingly, REINFORCE has nearly no decrease after warm-up in the single act and *ER* = 0 % condition. We have double-checked the related code and results for REINFORCE and could not find methodological issues there.

The reward development during training for the *40K* condition is presented in Fig. 3. In each condition, the policies benefit from eight times longer warm-up compared to *5K*. There is a remarkable drop for $A_{0,s}$, which is quickly compensated. We got the general impression that A2C benefits from the active error simulation.

3.2 Evaluation

Table 2 shows the evaluation results for the *5k* and *40k* condition. For the assessment of a trained policy, the values for dialogue length (L) in number of turns and the dialogue success rate (S) are important. Table 2 shows the average mean for L, R, and S, as well as the related standard deviation (σ) over the five training runs of the *5K* condition.

For *5K* and *40K* the results of Q-Learning and WoLF-PHC are quite similar. All policies benefit from the training in the *40K* condition, especially for *NUDA* = m and *ER* = 5 % and the combination of both. The results for A2C are remarkable, as A2C has good success rates and leads to average dialogue lengths which are much smaller than for the other algorithms in the same settings for *NUDA* and *ER*.

4 Discussion

A2C and REINFORCE (and neural network-based policies in general) can handle unknown input data, i.e., an unknown dialogue state in our case. Still, they have the drawback of more complex implementations and higher probabilities for implementation errors as well as higher computational efforts during training.

All algorithms do benefit from the more extensive training in the *40k* condition, but further studies should investigate the influence of the ratio between the number of warm-up dialogues and the overall number of training dialogues.

The actor-critic policy (A2C) does not suffer as much from concept error simulation as other methods do. A possible explanation could be that the error-simulation has an additional exploration effect, which is beneficial for the training. Casanueva et al. [2] find in their benchmark of deep RL techniques that A2C has the lowest performance among the policies evaluated there. One reason for the different ranking results might be that among the RL approaches used in our paper A2C is the most advanced. Q-Learning and WoLF-PHC do even not make use of neural networks. Other reasons could be that different dialogue state features and representations are used in both works as well as different neural network structures for the implementation of the A2C paradigm. Furthermore, we have trained with up to 40,000 dialogues while in [2] less (i.e. up to 10,000) were used.

It is not surprising that the Q-based policies have lower performance in scenarios with an activated error simulation and a user simulation that can send multiple dialogue acts in one turn, as both do increase the possible dialogue state space. Q-based policies cannot generalise and must act only randomly when confronted with new dialogue states. Nevertheless, we believe that it makes sense to use Q-Learning as a baseline, as it is easy to implement and shows comprehensible behavior.

Finally, our results show that our state encoder, which relies on a textual representation of the dialogue state and its interpretation by a convolutional network (convFE), can be used to substitute more artificial n-hot encoding.

5 Conclusions

We presented the results of a laborious comparison of four different reinforcement learning algorithms that are used to train policies for dialogue management. The results show that the A2C-based policy works best among different training conditions like concept errors and the number of user dialogue acts. Additionally, the results show that our approach of a learnable feature extractor (convFE) to interpret text-based dialogue state encodings does work.

We show that more extended training leads to higher performances of the trained policies. However, future work should research on the estimation of feasible training sizes and proper relations between the number of warm-up dialogues and exploration dialogues.

Concerning the presented dialogue encoder, future work should compare this approach directly with an n-hot approach to get insights into potential differences in trained policies' performance.

6 Acknowledgements

We would like to thank the Federal Ministry of Education and Research (BMBF) in Germany, which supports the work on this contribution by funding the project OKS - Optimierung konversationeller Schnittstellen (Funding-ID 16SV8151).

References

[1] YOUNG, S., B. THOMSON, J. D. WILLIAMS, and ET AL.: *POMDP-based statistical spoken dialogue systems: a review*. In *PROC IEEE*. 2013.

[2] CASANUEVA, I., P. BUDZIANOWSKI, P.-H. SU, N. MRKŠIĆ, T.-H. WEN, S. ULTES, L. ROJAS-BARAHONA, S. YOUNG, and M. GAŠIĆ: *A Benchmarking Environment for Reinforcement Learning Based Task Oriented Dialogue Management*. In *Proc. of Deep RL Symposium*. 2017.

[3] SUTTON, R. S. and A. G. BARTO: *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning Series. The MIT Press, Cambridge, Massachusetts, second edn., 2018.

[4] RIESER, V. and O. LEMON: *Reinforcement Learning for Adaptive Dialogue Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-24942-6.

[5] WEISZ, G., P. BUDZIANOWSKI, P.-H. SU, and M. GASIC: *Sample efficient deep reinforcement learning for dialogue systems with large action spaces*. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 26(11), p. 2083–2097, 2018. doi:10.1109/TASLP.2018.2851664.

[6] DUAN, Y., X. CHEN, R. HOUTHOOFT, J. SCHULMAN, and P. ABBEEL: *Benchmarking Deep Reinforcement Learning for Continuous Control*. In *Proc. 33rd Int. Conf. on Machine Learning*, pp. 1329–1338. 2016.

[7] ULTES, S., L. M. ROJAS-BARAHONA, P.-H. SU, D. VANDYKE, D. KIM, I. CASANUEVA, P. BUDZIANOWSKI, N. MRKŠIĆ, T.-H. WEN, M. GAŠIĆ, and S. YOUNG: *PyDial: A Multi-domain Statistical Dialogue System Toolkit*. In *Proc. ACL 2017, Sys. Demo.*, pp. 73–78. ACL, Vancouver, CAN, 2017.

[8] RAO, D. and B. MCMAHAN: *Natural Language Processing with PyTorch: Build Intelligent Language Applications Using Deep Learning*. O'Reilly Media, 2019.

[9] PAPANGELIS, A., Y.-C. WANG, P. MOLINO, and G. TUR: *Collaborative Multi-Agent Dialogue Model Training Via Reinforcement Learning*. In *Proc. SIGdial 2019*, pp. 92–102. Stockholm, Sweden, 2019.

[10] MICHAEL, T., S. HILLMANN, and B. WEISS: *An Artifical Conversational Agent for Students at the TU Berlin*. In *ESSV 2017*, vol. 86 of *Studientexte Zur Sprachkommunikation*, pp. 238–245. Saarbrücken, 2017.

[11] SCHATZMANN, J., B. THOMSON, K. WEILHAMMER, H. YE, and S. YOUNG: *Agenda-based user simulation for bootstrapping a POMDP dialogue system*. In *Conf. Human Language Technologies 2007*, pp. 149–152. ACL, Rochester, New York, 2007.

[12] HENDERSON, M., B. THOMSON, and J. D. WILLIAMS: *The Second Dialog State Tracking Challenge*. In *Proc. of the 15th Annual Meeting of SIGdial*, pp. 263–272. ACL, Philadelphia, PA, U.S.A., 2014.

[13] WATKINS, C. J. C. H.: *Learning from Delayed Rewards*. Ph.D. thesis, King's College, 1989.

[14] BOWLING, M. and M. VELOSO: *Rational and convergent learning in stochastic games*. In *Proc. of the 17th Int. Conf. on AI*, p. 1021–1026. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[15] WILLIAMS, R. J.: *Simple statistical gradient-following algorithms for connectionist reinforcement learning*. *Mach. Learn.*, 8(3–4), p. 229–256, 1992. doi:10.1007/BF00992696.

[16] MNIH, V., A. P. BADIA, M. MIRZA, A. GRAVES, T. LILLICRAP, T. HARLEY, D. SILVER, and K. KAVUKCUOGLU: *Asynchronous methods for deep reinforcement learning*. In *Proc. of the 33rd Int. Conf. on Machine Learning*, vol. 48, pp. 1928–1937. PMLR, New York, US, 2016.