Dense Rewards and Continual Reinforcement Learning for Task-oriented Dialogue Policies

Inaugural-Dissertation

zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultät
der Heinrich-Heine-Universität Düsseldorf

vorgelegt von

Christian Geishauser
aus Deutschland

Düsseldorf, October 2023

aus dem Institut für Informatik
der Heinrich-Heine-Universität Düsseldorf

Gedruckt mit der Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Heinrich-Heine-Universität Düsseldorf

Berichterstatter:

1. Prof. Dr. Milica Gašić

2. Prof. Dr. Dilek Hakkani-Tür

Tag der mündlichen Prüfung: 23. April 2024

Declaration of Authorship

I, *Christian Geishauser*, declare under oath that I have produced my thesis independently and without any undue assistance by third parties under consideration of the Principles for the Safeguarding of Good Scientific Practice at Heinrich Heine University Düsseldorf.

Düsseldorf,

Location, Date	Christian Geishauser

To my mum Angelika

Acknowledgements

First of all, I would like to thank the person who made all of this possible: my supervisor Prof. Dr. Milica Gašić. Thank you for putting the trust and believe into me that I am able to learn reinforcement learning and make a contribution to the field of dialogue policies. Thank you for giving me the time to learn and grow, and the support throughout the years. Thank you for teaching me how to become a researcher. I am very grateful for this unique part of my life that you made possible.

Secondly, I would like to thank all my colleagues. It was a great pleasure to work with you and spending so much time with you the last years. I am very grateful for all your support and your help with discussions, paper writing, or anything that came up. Moreover, a special thanks goes to the committee of my PhD defence and the reviewers of my PhD thesis who made the completion of this work possible.

I would like to thank all my friends that I got to know in Düsseldorf. Düsseldorf became my second home these years and I loved it to live here. I loved it because of you and I will really miss you.

Lastly, I thank my mother Angelika for her support from day 1. Your endless support, excitement and happiness about my life journey always makes me realize how grateful I can be.

Abstract

Humans continue learning throughout their lifetime in order to adapt to the changing world and advance as the number of challenges to face grows. When faced with a particular task to solve, they ask questions in the right moments to gather information about it and resolve uncertainty about any misunderstandings before actually solving the task. Task-oriented dialogue systems center around fulfilling the goal or task of a user during a conversation, where the goal is restricted to a certain, fixed, scope of possibilities such as travel planning and schedule organization. Unlike humans, these systems lack the ability for continual learning and neglect the important skill of information gathering or uncertainty reduction in their training. Nevertheless, the potential range of tasks a dialogue system can assist with is vast due to the expansive nature of human communication. Consequently, the scope of operation inevitably expands and the circumstances evolve, which necessitates the human-like abilities of continual learning. In addition, the dialogue policy, the decision-making component of the system that selects the response to the user, is trained in a trial-and-error process to maximize goal fulfillment. This training procedure is slow, since the only feedback given by goal fulfillment is obtained at the end of a conversation.

The main contributions of this thesis are as follows. Firstly, we propose an additional feedback signal for learning called *information gain* that is provided in every turn of the conversation and thus increases sample efficiency. Information gain encourages the dialogue policy to gather information about the user goal and reduce uncertainty in its understanding, which is an essential step preceding goal fulfillment. Our experiments with different tasks and noise settings show that the additional usage of information gain leads to faster learning and a better final policy.

Secondly, we are the first to introduce continual reinforcement learning for dialogue policies and propose a novel, dynamic architecture called *dynamic dialogue policy transformer* (DDPT). DDPT is based on the Transformer and a pre-trained language model and further advanced with a domain gate and hard-attention to allow dynamic input and output, forward transfer and dealing with many possible tasks. In the continual learning setup where tasks are introduced sequentially over time, our proposal DDPT achieves significant forward transfer and robustness against forgetting, while requiring no growth in neural network parameter size. Thirdly, we propose *realistic environments for continual reinforcement learning of dialogue policies* (RECORD), a more general and controllable continual learning setup with the goal of modeling the most important challenges in continual dialogue policy learning. Furthermore, we propose the usage of lifetime return and meta-gradient reinforcement learning for dialogue policy optimization for more robust learning and better adaptation during continual learning. We test multiple configurations of RECORD with different user behaviors and algorithms, where our proposals of lifetime return and meta-gradient reinforcement learning lead to consistent improvements.

Our results on information gain warrant a more widespread application to any task where information acquisition and uncertainty reduction plays a significant role, such as general information retrieval through dialogue. Furthermore, our contributions on continual learning pave the way for future advancements that move the field of dialogue policies from static to dynamic learning. In that realm, the versatility of RECORD lays the foundation for training and testing continual learning abilities of any task-oriented dialogue system.

Contents

List of Figures

Summary of Notation

This chapter summarizes the notations used throughout this thesis.

General Mathematical Notation

a, b, c	Scalars
$\mathbf{a}, \mathbf{b}, \mathbf{c}$	Vectors
a_i	i-th element of vector $\mathbf{a}$
$\mathbf{A}, \mathbf{B}, \mathbf{C}$	Matrices
a_{ij}	element of matrix $\mathbf{A}$ in row i and column j
$\boldsymbol{a_i}$	i-th row of matrix $\mathbf{A}$
$\mathbf{A}^T$	Transposition of matrix $\mathbf{A}$
$\mathbf{X} = (\mathbf{x_1}, ..., \mathbf{x_N})$	Sequence of N vectors
$\mathbf{a} = [a_i]_{i=1}^N$	vector written as a list of elements a_i
$\mathbf{A} = [\mathbf{a_i}]_{i=1}^N$	Matrix written as a list of rows $\mathbf{a_i}$
$\mathcal{X}, \mathcal{Y}, \mathcal{Z}$	Sets
$\{\}$	The empty set
$\mathcal{X} \setminus \mathcal{Y}$	Set consisting of all elements in $\mathcal{X}$ that are not in $\mathcal{Y}$
$\mathcal{X} \cup \mathcal{Y}$	Set consisting of all elements in $\mathcal{X}$ and $\mathcal{Y}$
$x \in \mathcal{X}$	x is an element of set $\mathcal{X}$
$f(x)$	A function of x
$f(\cdot)$	A function f with one argument
$\mathbb{N}$	Natural numbers
$\mathbb{Z}$	Integers
$\mathbb{R}$	Real numbers
$\mathbb{R}^d$	Set of d-dimensional vectors with elements in $\mathbb{R}$
$\mathbb{R}^{d_1 \times d_2}$	Real-valued matrices with d_1 rows and d_2 columns
$f : \mathcal{X} \to \mathcal{Y}$	Function f from set $\mathcal{X}$ to set $\mathcal{Y}$
$x \mapsto f(x)$	Value x maps to $f(x)$
$f \circ g$	Composition of functions f and g, i.e. $f(g(\cdot))$
$\mathbf{a} \odot \mathbf{b}$	element-wise multiplication of vectors $\mathbf{a}$ and $\mathbf{b}$
$\boldsymbol{a} \oplus \boldsymbol{b}$	Concatenation of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$
$\mathbf{AB}$	Matrix-matrix multiplication of matrices $\mathbf{A}$ and $\mathbf{B}$
e^x	Exponential function, where $e \approx 2.71828$
$\ln(x)$	Natural logarithm of x
Σ	Summation
∇f	Gradient of a function $f : \mathbb{R}^d \to \mathbb{R}$
$\nabla f\|_x$	Gradient evaluated at element x
$\approx$	approximately equal
$\max_x f(x)$	Maximum value of function f
$\arg\max_x f(x)$	Value x where $f(x)$ is maximal

X, Y, Z	Random variables
$P(X = x)$	Probability that random variable X takes values x
$P(X = x \vert Y = y)$	Probability of $X = x$ conditioned on $Y = y$
$\mathbb{E}_{X \sim P}[X]$	Expectation of a random variable X with distribution P
$\mathbb{E}_{x \sim P}[x]$	Expectation of random variable X with realizations x and distribution P
$\mathbb{1}_{x=y}$	Indicator function that is 1 if $x = y$ and 0 else
$O(n)$	Big-O notation, indicating the complexity of a function

Deep Learning Notation

$f(x; \boldsymbol{\theta})$	function f with input x, parameterized with parameters $\boldsymbol{\theta}$
f_l	l-th layer of a neural network
$\mathcal{D} = \{(x_i^{\text{in}}, y_i^{\text{out}})\}_{i=1}^{N}$	Dataset of size N with input-output pairs x_i^{in} and y_i^{out}
$\mathcal{L}(\mathcal{D}; \boldsymbol{\theta})$	Loss function dependent on parameters $\boldsymbol{\theta}$ and dataset $\mathcal{D}$
$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}; \boldsymbol{\theta})$	Gradient of $\mathcal{L}$ with respect to $\boldsymbol{\theta}$
η	Learning rate
ϕ	Activation function
$\mathbf{h}^{(i)}$	Output of the i-th layer in a neural network
$\hat{\boldsymbol{y}}$	Output vector of a neural network $f(x; \boldsymbol{\theta})$ for an input x
$\mathcal{D}_{\text{batch}}$	Mini-batch of data
$x \sim \mathcal{D}$	Element x uniformly sampled from dataset $\mathcal{D}$
$\mathbb{E}_{\mathcal{D}}[f(x)]$	Expectation where x is uniformly sampled from dataset $\mathcal{D}$
$a \leftarrow b$	Assigning value b to a

Reinforcement Learning Notation

$\mathcal{M}$	Markov decision process
$\mathcal{S}, \mathcal{A}, \mathcal{O}$	state space, action space and observation space
s, s'	State and next state
a	Action
r	Reward
$p(s' \vert s, a)$	Probability of s' given s, a
$p_0(s)$	Probability of starting in state s
t	discrete time step
s_t, a_t, r_t	state, action, reward at time step t
g_t	Return from time step t onwards
S, A, R, G	Random variables for state, action, reward and return
S_t, A_t, R_t, G_t	Random variables for state, action, reward and return at time step t
π, μ	Policies
$\pi(a \vert s)$	Probability of action a given state s
$V^{\pi}(s)$	State-value function of π
$V^{*}(s)$	Optimal state-value function
$Q^{\pi}(s, a)$	Action-value function of π
$Q^{*}(s, a)$	Optimal action-value function
$A^{\pi}(s, a)$	Advantage function of π
$V(s)$	Estimate of $V^{\pi}(s)$ or $V^{*}(s)$
$Q(s, a)$	Estimate of $Q^{\pi}(s, a)$ or $Q^{*}(s, a)$

$\pi_{\boldsymbol{\theta}}$	Policy π parameterized by $\boldsymbol{\theta}$
$V_{\boldsymbol{\theta}}$	State-value function approximator parameterized by $\boldsymbol{\theta}$
$Q_{\boldsymbol{\theta}}$	Action-value function approximator parameterized by $\boldsymbol{\theta}$
d^{π}	Occupancy measure
$d^{\pi}_{s_0}$	Occupancy measure conditioned on state s_0
τ	Trajectory of states, actions and rewards
$p_{\pi}(S_t = s)$	Probability that s occurs at time step t when using π
$p_{\pi}(\tau)$	Probability of trajectory τ when using π
$\mathbb{E}_{\pi}[X]$	Expectation of X when generating experience using π
o	Observation in a POMDP
$p_{\text{obs}}(o\|s)$	Probability of observation o given state s
$\mathcal{B}$	Belief state

Chapter 1

Introduction

The world is ever-changing. To survive in this complex environment, humankind as intelligent species has been equipped with various skills that enable continual learning. The capability for continual learning is a remarkable trait inherent in humans, enabling them to consistently acquire new knowledge and skills by drawing upon past experiences while being robust to forgetting. From infancy, individuals begin by mastering basic motor skills, which serve as building blocks for crawling, walking, and eventually running. In the realm of language, they progress from learning individual words to constructing intricate sentence structures and even acquiring multiple languages. Similarly, in mathematics, they grasp simple operations like addition and multiplication, laying the groundwork for complex calculations in the future. This ongoing ability to learn is an essential component of human development, allowing individuals to navigate the ever-evolving world that surrounds them (Bremner et al., 2012; Power and Schlaggar, 2016).

The success of humanity can be partially attributed to the collaborative efforts of individuals assisting one another in various tasks. Whenever someone encountered a particular challenge, they sought out an expert who could provide guidance and support in accomplishing that task. Effective communication through dialogue played a crucial role in identifying the underlying problem and working together to find a solution. Furthermore, the expert's information seeking behavior is essential in comprehending the problem at hand.

Similar to human experts, task-oriented dialogue systems strive to help humans in achieving their task through dialogue interaction. The tremendous amount of potential tasks necessitates the human-like abilities of continual learning as well as the information seeking behavior for these systems. In this thesis, we study how the information seeking behavior can be reinforced in dialogue systems to guide their learning as well as how dialogue systems are able to continue learning throughout their lifetime.

1.1 Task-oriented Dialogue Systems

A task-oriented dialogue system (TOD system) helps the user to fulfill a certain task. This can be for instance booking a flight, finding popular local attractions in a new city or querying the weather. The system here is not restricted to one task per dialogue but should for instance be able to book a hotel, find a restaurant and additionally book a taxi to commute between the two locations (Budzianowski et al., 2018). In order to find adequate entities such as restaurants or hotels, the system is equipped with an external database for entity retrieval. In contrast to chatbots which try to maximize user engagement, TOD systems aim to fulfill the task as quickly as possible. TOD systems operate within certain boundaries that are defined by an underlying ontology, which is usually hand-crafted by the designers. The ontology is comprised of domains, e.g. restaurants or hotels, domain-specific slots, e.g. the area or price,

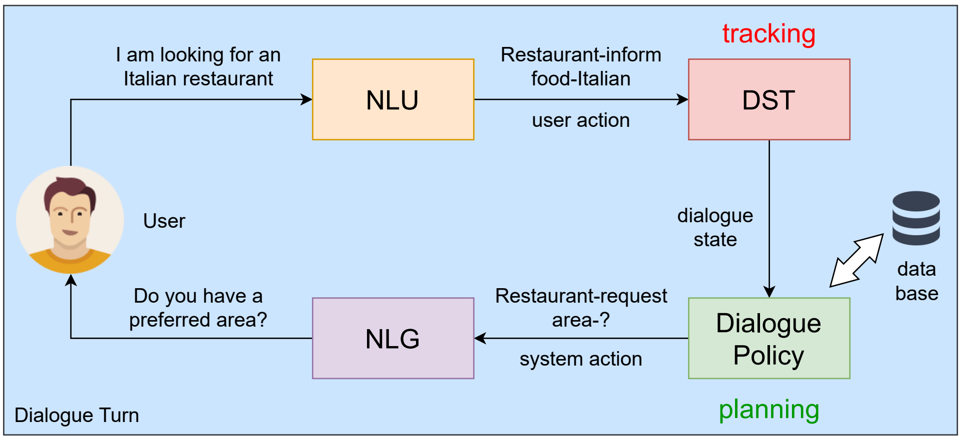


FIGURE 1.1: Overview of one dialogue turn during a conversation. The user text will be converted by the natural language understanding (NLU) component into a semantic representation, i.e. the user action. Subsequently, the dialogue state tracker (DST) updates its dialogue state and passes it further to the dialogue policy. The dialogue policy selects an action that is driving the dialogue towards task success. The policy can query the database for obtaining entities such as restaurants. Finally, the natural language generation (NLG) component converts the semantic system action into natural language.

and values that a slot can take, e.g west, east, north, south, and center for the area slot. These concepts together define the scope of information that can be understood by the system, such as the restaurant area or the number of hotel stars. Moreover, the ontology defines the actions of the system for each such domain, such as booking a hotel room or providing the phone number of the requested restaurant.

In a conversation it is crucial to understand what the user just said and remember what has been uttered before. Based on that information, the system has to come up with an appropriate response. In modular TOD systems these capabilities are modeled by dedicated components as can be seen in Figure 1.1. The natural language understanding (NLU) module is the first component in a dialogue system and is responsible for mapping a given user utterance into a semantic representation that decodes the meaning (Namazifar et al., 2021; Tur et al., 2012). It is tasked with inferring the user intent as well as slot-value pairs. As an example, the utterance "I am looking for an Italian restaurant" can be mapped to the semantic representation `restaurant-inform-food-Italian`, also called a dialogue act. While the NLU component deals with the current utterance, the subsequent dialogue state tracking (DST) component is responsible for tracking the dialogue history and in every step updates its dialogue state (Heck et al., 2020; Lee et al., 2019; Ramadan et al., 2018). The main objective of the DST is to track the user goal throughout the conversation. In order to take uncertainty about the dialogue history into account, it is also possible to produce a belief state, which defines a probability distribution over possible dialogue states (van Niekerk et al., 2020; Young et al., 2007). This helps in propagating uncertainty through the pipeline and allows subsequent components to produce a more robust behavior (van Niekerk et al., 2021; Young et al., 2007). Based on the dialogue or belief state, the dialogue policy component decides on a semantic action to take such as requesting the area of the restaurant or informing the phone number of the hotel (Levin and Pieraccini, 1997; Williams and Young, 2007). The dialogue policy is the planning component of the system and steers the conversation towards task success. When deciding on an action to take, the policy can additionally query an accompanying data base for retrieving matching entities. Finally the semantic policy action is transformed into natural language by the natural language generation (NLG) module (Peng et al., 2020; Wen et al., 2015). Spoken dialogue systems are additionally equipped with an automatic speech recognizer (Lu et al., 2015) as well as a text-to-speech synthesizer (Oord et al., 2016).

One turn in a dialogue is given by a user utterance and the corresponding system utterance as depicted in Figure 1.1. A dialogue is comprised of a sequence of turns where the policy tries to find the best possible response in every turn of the dialogue in order to accomplish the user goal. Consequently, the dialogue policy has to solve a sequential decision making problem during interaction, which motivates the optimization through reinforcement learning (RL) (Levin and Pieraccini, 1997).

1.2 Dialogue Policy Optimization through Reinforcement Learning

This work focuses on dialogue policy optimization through reinforcement learning (RL). Reinforcement learning is a learning paradigm where a RL agent learns through a trial-error process by interacting with its environment (Sutton et al., 1999). In every step, the RL agent's policy (this is where the name dialogue policy comes from) maps states into actions so as to maximize the sum of a numerical reward signal, where this reward signal can be delayed. The action taken can influence the next state and hence the future of all states. This also holds for dialogue where what is said now can have an effect on future responses. In terms of dialogue, the dialogue policy interacts with a user (the environment in this case), collects experience through that interaction, obtains rewards (usually defined by whether the user goal was achieved and how efficient the dialogue was) and learns from it to improve itself. Since interactions with real humans are costly and time-consuming, researchers usually employ user simulators for their studies (Gur et al., 2018; Lin et al., 2021; Schatzmann et al., 2007).

1.3 Challenges in Dialogue Policy Learning

1.3.1 Sparse Rewards

Task-oriented systems focus on achieving the user's objective, which can only be assessed once the conversation is complete. The dialogue policy obtains a feedback signal, the reward, at the end that measures the success or failure of the dialogue and an uninformative small negative reward in every step to encourage fulfillment of the goal in the shortest amount of time (see also Figure 1.2). The reward is hence *sparse* in the sense that only few turns give the policy informative feedback. Consequently, the policy faces the challenge of credit assignment, meaning it has to find the responses of the dialogue that are responsible for task success or failure and reinforce or suppress them for future behavior. This analogous problem also occurs in chess for instance, where one has to find the sequence of moves that eventually lead to a winning or losing of the game. Since reinforcement learning relies on trial-and-error, coupled with sparse rewards, numerous interactions are necessary to learn optimized behavior. This is especially problematic if dialogue systems are supposed to learn in interaction with real users and not only in simulation.

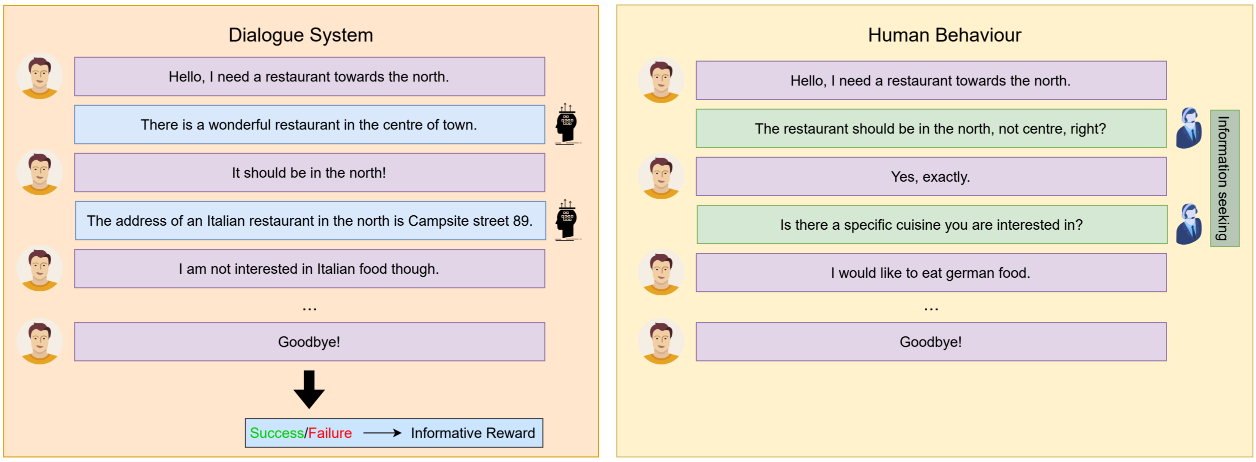


FIGURE 1.2: Left: as a task-oriented dialogue can only be evaluated for success or failure once it is finished, the corresponding informative reward for training the dialogue system can be only obtained at the end, leading to sample inefficient learning since it is difficult to judge which outputs should be reinforced in the future and which should be suppressed. Right: humans naturally resolve uncertainty and seek information about the task during a conversation, potentially rewarding themselves without external feedback.

The integral first step for solving a user goal is to understand the goal in the first place. Humans ask question to get to know each other or resolve uncertainty naturally during the conversation. An example where a human operator naturally resolves uncertainty and asks for information, while the dialogue system fails to do so, can be seen in Figure 1.2. The question is whether this information seeking skill can be reinforced in the dialogue policy through a reward in order to guide learning and provide more dense reward signals.

1.3.2 Continual Dialogue Policy Learning

Humans continuously face challenges during their lifetime, seamlessly comprehend new knowledge and master new skills (Grossberg, 2013). We often take the ability to continually learn for granted and ignore how remarkable it actually is. The brain requires plasticity in order to adapt to changes rapidly. At the same time, it has to stabilize the existing knowledge to prevent forgetting. The brain excels at solving these two contradicting tasks, what is known in neuroscience as plasticity-stability dilemma (Grossberg, 1982). When we learn something new, we rarely start from scratch but leverage already existing knowledge to speed up our learning (Barnett and Ceci, 2002; Bremner et al., 2012) - learning one language becomes easier once other similar languages are already known. When we are faced with a specific task, we put our attention on the important information and ignore irrelevant one. Moreover, we master all of that with limited memory and neural capacity.

Similarly to humans, intelligent task-oriented dialogue systems that are meant to be assisting humans throughout their lifetime will face ongoing challenges and tasks as well. The system might start off with helping users to find restaurants in their favorite city. Soon, users might want to also book hotels or flights for their journey. Another prominent example was recently given by the Corona pandemic, where a dialogue system could have helped users to book vaccination or test appointments. The tremendous amount of tasks that a task-oriented dialogue system can potentially help with makes it absolutely inevitable to study the circumstances of continual learning.

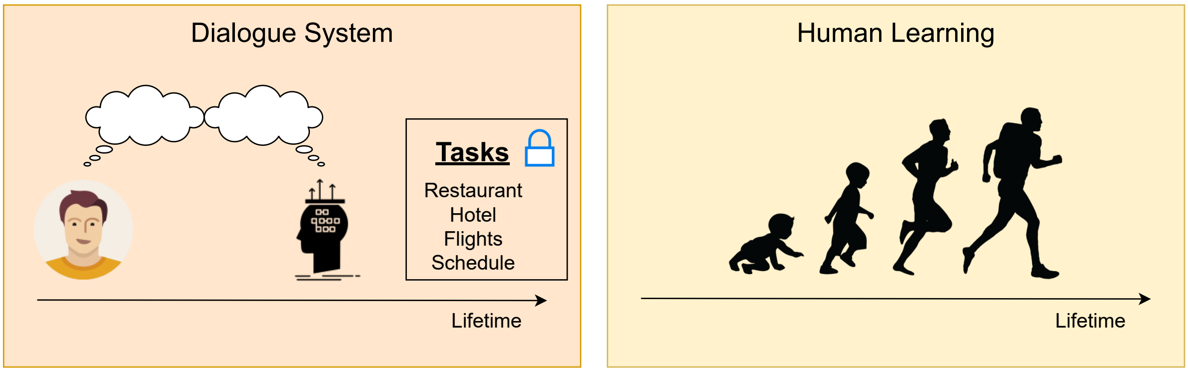


FIGURE 1.3: Left: dialogue policy research has so far focused mainly on fixed environments with a fixed number of tasks (e.g. restaurant, hotel, flights, and schedule) that are given by a pre-defined ontology. Right: humans continuously learn throughout their lifetime and master new skills over time, building on already acquired knowledge. How can we equip dialogue policies with continual learning abilities?

Research on the dialogue policy has so far focused mainly on how to excel in a fixed environment, including a fixed user and ontology (Su et al., 2016; Takanobu et al., 2019; Weisz et al., 2018; Wesselmann et al., 2019). As depicted in Figure 1.3, the policy is challenged on a single or multiple tasks such as providing restaurant or hotel information and the goal is to maximize performance in this static environment. Many advancements have been made to reach as high performance as possible while being sample efficient (Lipton et al., 2018; Weisz et al., 2018).

In a parallel line of work, researchers looked into how policies can be built in order to transfer their knowledge from one domain to another (Chen et al., 2018; Chen et al., 2020; Lin et al., 2021; Wang et al., 2015; Xu et al., 2020). These works assumed that the possible information being passed to the dialogue policy as well as the actions have a certain structure that allowed them to build models that can be used for multiple tasks without architectural changes.

One of the key challenges that so far remained unaddressed in dialogue policy optimization is the question of how the policy can continue learning over time. This starts with fundamental challenges such as how new information or actions can be incorporated into the policy, which arises through a growing ontology. It requires measurements for forgetting and forward transfer to new tasks. Furthermore, we need to answer the question of what constitutes a reasonable continual learning setup for dialogue policies.

1.4 Contributions

1. **Information gain as intrinsic reward to reinforce information seeking behavior:** based on the observation that understanding the user goal is the first challenge before any task can be fulfilled, we propose information gain for enhancing dialogue policy learning. Information gain is an intrinsic reward that can be calculated in every turn of the conversation, thus providing informative feedback in every step compared to the sparse reward measuring task success. Information gain reinforces actions if they gather new information of the user or resolve uncertainty, such as requests or confirmations. We used a hierarchical dialogue policy with a dedicated information

seeking policy and optimized it for maximizing information gain only. Experiments in various domains and noise setups show that information gain leads to enhanced learning as well as better final performance. Interactions with humans moreover showed that the policy using information gain appropriately asks for information if necessary.

2. **A dynamic dialogue policy architecture for continual reinforcement learning:** we are the first to explore continual reinforcement learning for dialogue policies. We provide a continual RL algorithm, baseline models and evaluation metrics to enable dialogue policy research to go from static to dynamic learning. Moreover, we propose the dynamic dialogue policy transformer (DDPT), that can incorporate new information and actions seamlessly through descriptions and enables reasoning over a large amount of domains by leveraging a hard attention mask on active domains. Additionally, it leverages a domain gate for enabling zero-shot transfer to new domains. Experiments with multiple continual learning setups validate that DDPT obtains strong zero-shot capabilities as well as being robust to forgetting. Our architecture achieves that without any growth in neural network parameter size

3. **Learning with an open horizon in ever-changing dialogue circumstances:** we propose a novel continual learning framework called RECORD that models the most important challenges for lifelong dialogue policy learning. In contrast to previous works, it models 1) different users, 2) multi-domain dialogues, 3) reoccuring domains, and 4) changing user demands. The framework allows flexible and controllable continual learning setups and evaluates the performance of dialogue policies throughout their lifetime. In addition, we propose the optimization of lifetime return and meta-gradient reinforcement learning for taking long-term future into account and dynamically adapting hyperparameters during learning. We equip two RL algorithms with lifetime return optimization and meta-gradient RL and show that they outperform baselines in a challenging approximate real world.

1.5 Thesis Structure

This thesis consists of eight chapters:

Chapter 1 presents a general introduction, setting the context and outlining the thesis objectives.

Chapter 2 offers an overview of deep learning, explaining fundamental concepts and architectures.

Chapter 3 covers the necessary background on reinforcement learning, which will be used for dialogue policy optimization.

Chapter 4 discusses dialogue policy modeling and challenges in more detail.

Chapter 5 introduces information gain as intrinsic reward for dialogue policy optimization and showcases its benefit for accelerating the learning process.

Chapter 6 introduces the first work on continual reinforcement learning for the dialogue policy, including baseline algorithms and a novel architecture based on the transformer.

Chapter 7 introduces a more general and flexible continual learning setup for dialogue policies and proposes algorithmic advancements based on lifetime return optimization and meta-gradient RL.

Chapter 8 summarizes the thesis's key findings, examines the implications of the presented research, and suggests directions for future investigation.

Chapter 2

Deep Learning

2.1 Neural Networks as Function Approximators

Imagine the task of classifying an image into one of several categories such as cats, dogs, horses, etc. We can think of the image-category pairs as generated by an unknown function $f_{\text{unk}} : \mathcal{X} \to \mathcal{Y}$ that takes images x as input and outputs the correct category y. We do not have access to f_{unk} but only to a data set $\mathcal{D}$ containing pairs $(x^{\text{in}}, y^{\text{out}})$ produced by f_{unk}. The goal is to produce an approximation f of f_{unk}, leveraging data set $\mathcal{D}$, that obtains correct predictions for images in $\mathcal{D}$ and generalizes well to novel images.

In order to produce such an approximation, we are required to first specify a function class. The function class of artificial neural networks (ANN), or short neural networks, has obtained a surge in popularity due to its powerful capabilities of feature extraction and approximation. The empirical success of ANN is verified by the Universal Approximation Theorem (Hornik et al., 1989) that states that any continuous function $f_{\text{unk}} : \mathcal{K} \to \mathbb{R}^m$, where $\mathcal{K} \subset \mathbb{R}^n$ is compact[1], can be approximated with a feed-forward neural network with two layers. In the following sections we discuss popular neural network architectures and how to optimize them.

2.2 Feed-forward Neural Networks

A feed-forward neural network (FFN) defines a mapping $\hat{\boldsymbol{y}} = f(\boldsymbol{x}; \boldsymbol{\theta})$ with learnable neural network parameters $\boldsymbol{\theta}$. The FFN is organized in a sequence of layers $f_1, ..., f_L$, where the first layer f_1 is called the input layer, the last layer f_L is called the output layer, and intermediate layers are called hidden layers. These models are called feed-forward because information flows through the layers without any feedback connections in which outputs of the model are fed back into itself (Goodfellow et al., 2016). Given an input together with its input feature vector $\boldsymbol{x} \in \mathbb{R}^{d_0}, d_0 \in \mathbb{N}$, (such as a flattened image), the computation of a feed-forward neural network f with L layers is depicted in Figure 2.1 and given as follows:

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = f_L \circ ... \circ f_l \circ ... \circ f_1(\boldsymbol{x}), \quad \text{where } f_l(\boldsymbol{h}) = \phi_l(\boldsymbol{W_l h} + \boldsymbol{b_l}), \tag{2.1}$$

where $\boldsymbol{W_l} \in \mathbb{R}^{d_l \times d_{l-1}}$ and $\boldsymbol{b_l} \in \mathbb{R}^{d_l}$ are learnable weight matrices and bias vectors, respectively, and the operator $\circ$ denotes the composition of functions. The neural network parameters $\boldsymbol{\theta}$ are given by the set of all weight matrices and bias vectors. The number of input and output neurons in layer l is d_{l-1} and d_l, respectively. The number of layers define

[1] $\mathcal{K} \subset \mathbb{R}^n$ is compact if it is closed and bounded.

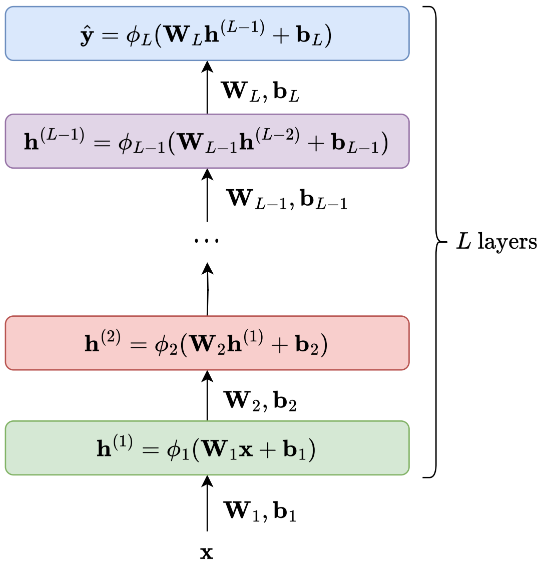


FIGURE 2.1: The architecture of a feed forward network (FFN). The input vector $\boldsymbol{x}$ is fed through L layers, each layer being comprised of a learnable weight matrix $\boldsymbol{W}_l$, bias vector $\boldsymbol{b}_l$, and an activation function ϕ_l.

the depth of the neural network, whereas the number of neurons within a layer determines its width.

The different ϕ_l are called activation functions and are commonly applied element-wise if given through a mapping from $\mathbb{R}$ to $\mathbb{R}$. These functions are typically chosen to be non-linear as the neural network can otherwise only model linear functions. If ϕ_l is the identity function, the layer f_l is called a linear layer. Popular activation functions are given by

$$\phi_{\text{ReLU}}: \quad \mathbb{R} \to \mathbb{R}, \quad x \mapsto \max(0, x) \tag{2.2}$$

$$\phi_{\text{Sigmoid}}: \quad \mathbb{R} \to \mathbb{R}, \quad x \mapsto \frac{1}{1 + e^{-x}} \tag{2.3}$$

$$\phi_{\text{Tanh}}: \quad \mathbb{R} \to \mathbb{R}, \quad x \mapsto \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.4}$$

The softmax activation function is typically used for mapping a general vector into a vector of probabilities and given by

$$\phi_{\text{softmax}}: \quad \mathbb{R}^d \to \mathbb{R}^d, \quad \boldsymbol{x} = [x_1, ..., x_d] \mapsto \left[\frac{e^{x_i}}{\sum_j e^{x_j}}\right]_{i=1}^{d} \tag{2.5}$$

If the task of the neural network is regression, i.e. predicting a real-valued number, the weight matrix $\boldsymbol{W_L}$ is of size $1 \times d_{L-1}$ and $f(\boldsymbol{x}; \boldsymbol{\theta}) \in \mathbb{R}$. On the other hand, if the task is classifying an input into one of N_{cat} categories $\{y_1, ..., y_{N_{\text{cat}}}\}$, the weight matrix $\boldsymbol{W_L}$ is of size $N_{\text{cat}} \times d_{L-1}$ and consequently $f(\boldsymbol{x}; \boldsymbol{\theta}) \in \mathbb{R}^{N_{\text{cat}}}$. Moreover, it is common to set $\phi_L = \phi_{\text{softmax}}$ such that the output is a probability distribution

$$\hat{\mathbf{y}} = f(\boldsymbol{x}; \boldsymbol{\theta}) = [p(y_1 \mid \boldsymbol{x}; \boldsymbol{\theta}), ..., p(y_{N_{\text{cat}}} \mid \boldsymbol{x}; \boldsymbol{\theta})]. \tag{2.6}$$

An actual category $\hat{y}$ is produced by sampling from $\hat{\boldsymbol{y}} = f(\boldsymbol{x};\boldsymbol{\theta})$ or taking the category with the highest probability.

The feed-forward neural network has the disadvantage that it can only process one input feature vector, but not a sequence of input vectors that are required for processing a sequence of words or a time series.

2.3 Recurrent Neural Networks

Recurrent neural networks (RNNs) are built for the purpose of processing a sequence of input vectors $\boldsymbol{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_i, ..., \boldsymbol{x}_M)$ that naturally occur for time series prediction or if a sequence of words are required to be processed (such as in text classification or machine translation). In every step i, a recurrent network with $L > 1$ layers $f_1, ..., f_L$ obtains as input not only the input feature vector $\boldsymbol{x}_i \in \mathbb{R}^{d_0}$, but also additional hidden vectors $\boldsymbol{h}_{i-1}^{(l)} \in \mathbb{R}^{d_l}$ for every $l = 1, ..., L-1$. The purpose of $\boldsymbol{h}_{i-1}^{(l)}$ is to capture important information from all previous steps $\boldsymbol{x}_1, ..., \boldsymbol{x}_{i-1}$, which constitutes it as a form of memory. For a step i, the RNN calculations are performed as follows (and shown in Figure 2.2), where $\boldsymbol{h}_0^{(l)}$ is typically initialized as a vector of zeros if no other context information is available:

$$f_1(\boldsymbol{x}_i, \boldsymbol{h}_{i-1}^{(1)}) = \boldsymbol{h}_i^{(1)} = \phi_1(\boldsymbol{W_1}\boldsymbol{x}_i + \boldsymbol{U_1}\boldsymbol{h}_{i-1}^{(1)} + \boldsymbol{b_1}), \tag{2.7}$$

$$f_l(\boldsymbol{h}_i^{(l-1)}, \boldsymbol{h}_{i-1}^{(l)}) = \boldsymbol{h}_i^{(l)} = \phi_l(\boldsymbol{W_l}\boldsymbol{h}_i^{(l-1)} + \boldsymbol{U_l}\boldsymbol{h}_{i-1}^{(l)} + \boldsymbol{b_l}), \quad l = 2, ..., L-1, \tag{2.8}$$

$$f_L(\boldsymbol{h}_i^{(L-1)}) = \phi_L(\boldsymbol{W_L}\boldsymbol{h}_i^{(L-1)} + \boldsymbol{b_L}), \tag{2.9}$$

where $\boldsymbol{W_l} \in \mathbb{R}^{d_l \times d_{l-1}}$, $\boldsymbol{U_l} \in \mathbb{R}^{d_l \times d_l}$ and $\boldsymbol{b_l} \in \mathbb{R}^{d_l}$ are learnable weight matrices and bias vectors, respectively. The different ϕ_l are activation functions as before.

The equations lead to the following observations. Firstly, calculating the vectors for step i requires the calculation for $i-1$, such that no computational parallelism across steps is possible. Secondly, the number of neural network parameters is increased compared to the feed-forward network as we have additional weight matrices $\boldsymbol{U_l}$. Moreover, the same weight matrices and bias vectors are used for every step. Thirdly, if $\boldsymbol{h}_0^{(l)}$ is initialized as a vector of zeros and the sequence of input vectors is of length 1, the RNN calculation coincides with the calculation of the feed-forward network.

The vanilla RNN architecture as explained above has been advanced in subsequent works, which led to long short-term memory networks (Hochreiter and Schmidhuber, 1997) and gated recurrent units (Cho et al., 2014a) with improved memory and bidirectional RNNs (Schuster and Paliwal, 1997) that leverage both past and future steps for calculating the hidden vectors.

2.3.1 Encoder-Decoder RNNs

Consider the task of machine translation, where a sequence of words $x_1, ..., x_i, ..., x_M$ in a source language has to be translated into a sequence of words $y_1, ..., y_j, ..., y_N$ in a target language. In order to solve this task, one approach is to first extract feature vectors from the sequence $x_1, ..., x_i, ..., x_M$, i.e. *encode* it, and subsequently use the features to generate the sequence $y_1, ..., y_j, ..., y_N$, i.e. *decode* it. This idea resulted in the Encoder-Decoder RNN (Cho

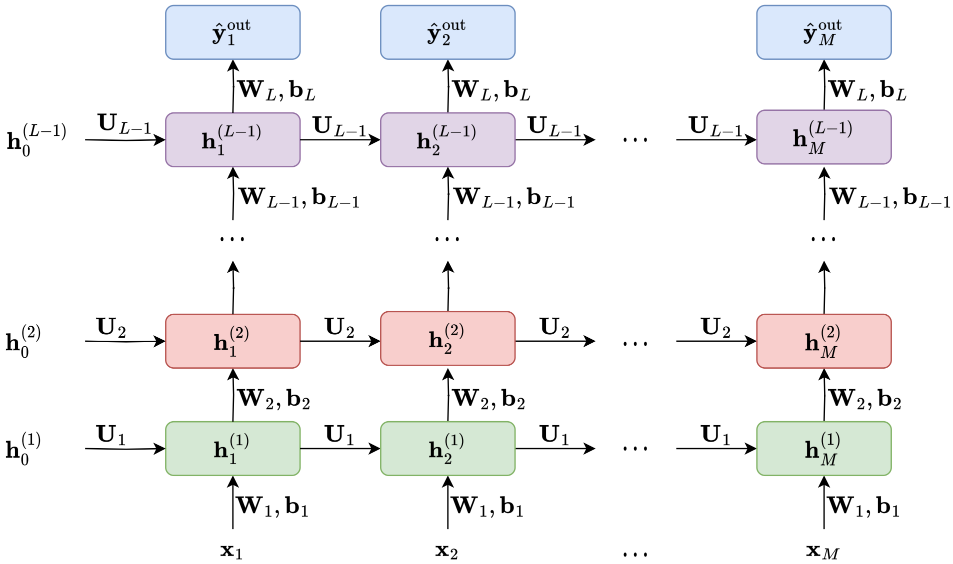


FIGURE 2.2: The architecture of a recurrent neural network (RNN). RNNs are built for processing a sequence of vectors $(\boldsymbol{x}_1, ..., \boldsymbol{x}_i, ..., \boldsymbol{x}_M)$. In every step i, each layer of the RNN obtains the output of the previous layer as input. The hidden layers additionally obtain a hidden vector $\boldsymbol{h}$ that encodes information of previous steps.

et al., 2014b), which consists of two RNNs for encoding and decoding, respectively. The encoder-decoder architecture is depicted in Figure 2.3 and explained below.

Encoder

The task of the encoder is to extract useful features from the input sequence. In a first step, feature vectors $(\boldsymbol{x}_1, ..., \boldsymbol{x}_M)$ are extracted from the input sequence $x_1, ..., x_M$. In machine translation, this is commonly achieved via using a learnable vector for every word x_i. The sequence $(\boldsymbol{x}_1, ..., \boldsymbol{x}_M)$ is then fed into the encoder RNN to produce hidden vectors $\boldsymbol{h}_{M,\text{enc}}^{(l)}$ for all $l = 1, ..., L-1$, where $\boldsymbol{h}_0^{(l)}$ is initialized as a vector of zeros.

Decoder

The task of the decoder is to produce a sequence $\hat{y}_1, ..., \hat{y}_N$ that matches the output $y_1, ..., y_N$. In order to pass the information extracted from the encoder to the decoder, we initialize the hidden vectors $\boldsymbol{h}_{0,dec}^{(l)}$ of the decoder as $\boldsymbol{h}_{0,\text{dec}}^{(l)} = \boldsymbol{h}_{M,\text{enc}}^{(l)}$ for all $l = 1, ..., L-1$.

In every decoding step $j = 1, ..., N$, the input is a feature vector $\hat{\boldsymbol{y}}_{j-1}$ of the previous prediction $\hat{y}_{j-1}$, where $\hat{\boldsymbol{y}}_0$ is initialized as a vector of zeros or a dedicated vector indicating the start of the decoding process for instance. The decoder then produces an output vector $\hat{\boldsymbol{y}}_j^{\text{out}}$, where the activation function of the last layer of the decoder is given by $\phi_{L,\text{dec}} = \phi_{\text{softmax}}$. Afterwards, we sample an outcome $\hat{y}_j$ from the distribution $\hat{\boldsymbol{y}}_j^{\text{out}}$ and the process repeats with decoding step $j+1$.

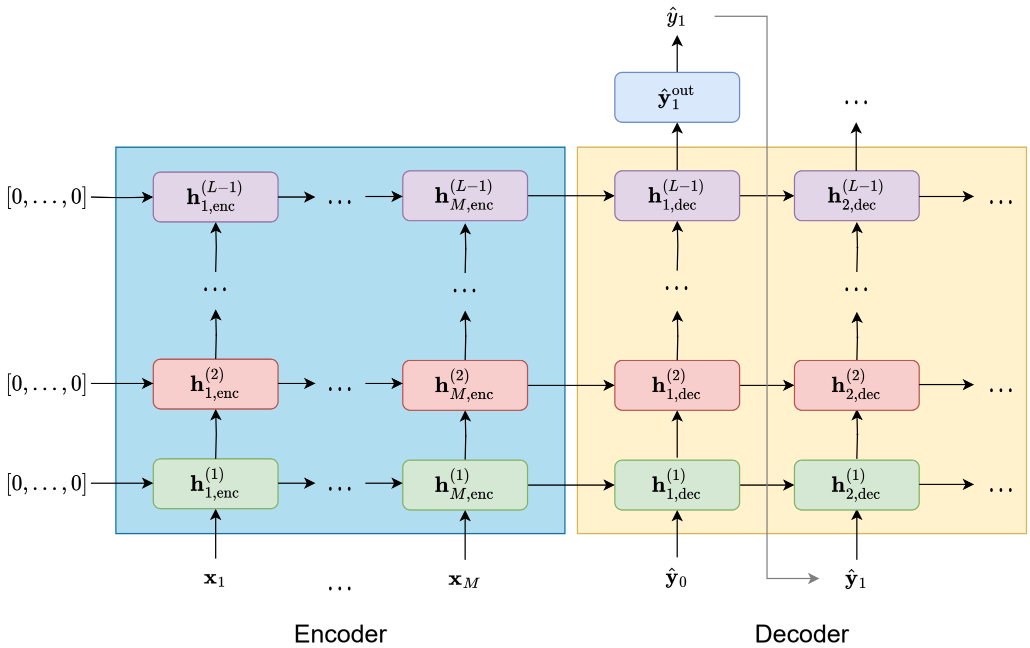


FIGURE 2.3: Encoder-Decoder RNN architecture. The architecture is composed of two RNNs for the tasks of encoding and decoding. The encoder obtains a sequence of vectors $(\boldsymbol{x}_1, ..., \boldsymbol{x}_i, ..., \boldsymbol{x}_M)$ as input and produces hidden vectors $\boldsymbol{h}_{M,\text{enc}}^{(l)}$ for all $l = 1, ..., L-1$. In order to pass the information from the encoder to the decoder, the hidden vectors $\boldsymbol{h}_{0,\text{dec}}^{(l)}$ of the decoder are initialized as $\boldsymbol{h}_{0,\text{dec}}^{(l)} = \boldsymbol{h}_{M,\text{enc}}^{(l)}$ for all $l = 1, ..., L-1$. The decoder then produces a sequence of outputs $\hat{y}_1, ..., \hat{y}_N$. In every decoding step j, the feature vector $\hat{\mathbf{y}}_{j-1}$ of the previous prediction $\hat{y}_{j-1}$ becomes the input to the decoder.

Attention

The hidden vectors $\boldsymbol{h}_{M,\text{enc}}^{(l)}$ create a bottleneck in the encoder-decoder architecture as they carry the burden of containing the information of the entire input sequence for producing the entire output sequence. This is especially problematic for long input sequences since we only have $L-1$ hidden vectors of fixed dimensionality to store the entire input. This is exacerbated due to the fact that different output predictions might require the input features to different extents. For instance, producing the translation of the word dog in the output will require knowledge of the presence of the word dog in the input, whereas this knowledge might not be relevant for the prediction of another word. In order to circumvent the issue, the attention mechanism has been proposed (Bahdanau et al., 2015; Luong et al., 2015), which provides a context vector $\boldsymbol{c}_j$ that encapsulates the encoded features in every decoding step j to different extents.

In every decoding step j, the attention mechanism is applied before producing the output vector $\hat{\boldsymbol{y}}_j^{\text{out}}$ as follows (see also Figure 2.4). For a context vector $\boldsymbol{c}_j \in \mathbb{R}^{d_{L-1}}$, where d_{L-1} is the dimension of the hidden vectors $\boldsymbol{h}_{i,\text{enc}}^{(L-1)}$, we calculate the output $\hat{\boldsymbol{y}}_j^{\text{out}}$ as

$$\hat{\boldsymbol{y}}_j^{\text{out}} = \phi_L(\boldsymbol{W_L}\tilde{\boldsymbol{h}}_{j,\text{dec}}^{(L-1)} + \boldsymbol{b_L}), \tag{2.10}$$

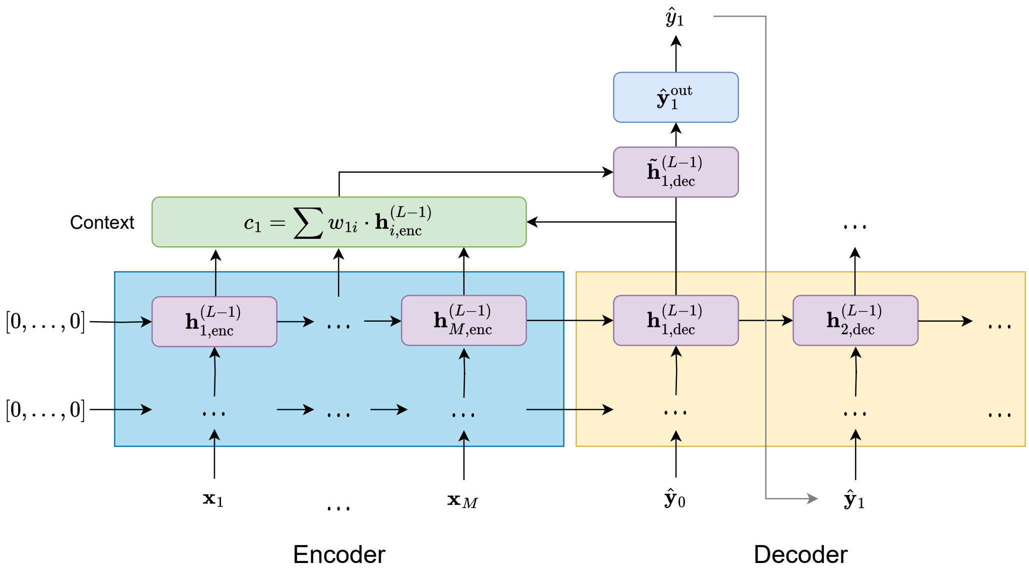


FIGURE 2.4: Encoder-Decoder RNN architecture with attention. The attention mechanism is applied before calculating the output $\hat{\boldsymbol{y}}_j$ and provides the decoder with additional information about the input features in every decoding step j. To this end, a context vector c_j is calculated as a weighted sum of the last hidden vectors of the encoder.

$$\tilde{\boldsymbol{h}}_{j,\text{dec}}^{(L-1)} = \phi_c(\boldsymbol{W}_c \cdot \boldsymbol{c}_j \oplus \boldsymbol{h}_{j,\text{dec}}^{(L-1)}), \tag{2.11}$$

where $\boldsymbol{W}_c \in \mathbb{R}^{d_{L-1} \times 2d_{L-1}}$ is a learnable weight matrix. In contrast to before, we thus have an additional layer that incorporates information from a context vector $\boldsymbol{c}_j$. The context vector is supposed to provide information of the input sequence that is beneficial for the prediction of $\hat{y}_j$. We calculate the context vector $\boldsymbol{c}_j$ as a weighted sum of input feature vectors

$$\boldsymbol{c}_j = \sum_i w_{ji} \cdot \boldsymbol{h}_{i,\text{enc}}^{(L-1)} \in \mathbb{R}^{d_{L-1}}, \quad \text{where} \quad \sum_i w_{ji} = 1. \tag{2.12}$$

The weights w_{ji} are called attention weights since they determine how much attention should be put on each hidden vector $\boldsymbol{h}_{i,\text{enc}}^{(L-1)}$. While there are multiple ways to calculate the attention weights (Luong et al., 2015), the simplest way is given by a dot-product followed by a softmax for normalization

$$w_{ji} = \phi_{\text{softmax}}([\boldsymbol{h}_{j,\text{dec}}^{(L-1)} (\boldsymbol{h}_{i,\text{enc}}^{(L-1)})^T]_{i=1}^M])_i. \tag{2.13}$$

The corresponding attention mechanism is referred to as dot-product attention. The decision of how to calculate the attention weights and where the context vector should be added exactly is a design decision that can differ between methods (Bahdanau et al., 2015; Luong et al., 2015). The overarching novelty comes from the idea to incorporate in every decoding step j a weighted sum of encoded feature vectors of the input sequence $(\boldsymbol{x}_1, ..., \boldsymbol{x}_M)$. The attention mechanism relieves the final hidden vectors $\boldsymbol{h}_{M,\text{enc}}^{(l)}$ from the requirement of memorizing the entire input sequence and provides the decoder more access to the input sequence. As an example, imagine reading a book and being asked questions afterwards. If we internally use RNNs without attention, we are required to memorize the entire content of the book. If we additionally use attention, we still have access to the book and only need to know at which parts of the book we need to look at depending on the question being asked. The attention mechanism will be heavily exploited in the Transformer architecture that we explain next.

2.4 Transformer Networks

For an input sequence of length M, the number of sequential operations in a RNN is $\mathcal{O}(M)$, which leads to high time-complexity for long sequences. The problem arises from the inherently sequential nature of RNNs that precludes parallelization across steps i. Moreover, the RNN has only a limited ability to model long-range dependencies of the inputs $\boldsymbol{x}_i$ due to potential information loss over time. The seminal work by Vaswani et al. (2017) proposed the Transformer network in order to overcome these issues by relying entirely on the attention mechanism that we have already seen in Section 2.3.1. In contrast to the attention mechanism for an Encoder-Decoder RNN, where an output vector puts attention on a sequence of input vectors, the Transformer introduces `self-attention`. The self-attention mechanism allows an input sequence to put attention on *itself*, which opens up parallelization across steps i and modeling of long-term dependencies as we will see in the following. The Transformer is composed of multiple components, summarized in the following and explained in detail below.

- **Attention:** the attention mechanism is the main ingredient in the Transformer and will be used in self-attention as well as cross-attention for the encoder-decoder Transformer. It allows parallelization across steps and the modeling of long-term dependencies.
- **Multi-Head Attention**: this will calculate multiple attentions in parallel using different linear transformations. It allows the model to extract different features of the input before applying attention.
- **Feed-forward Networks**: a feed-forward network is used to equip the Transformer with more powerful non-linear feature extraction capabilities.
- **Positional Encoding**: this will provide the Transformer information about the position of each input in the input sequence.

2.4.1 Self-Attention

Our motivation is the following. Given a sequence $(\boldsymbol{x}_1, ..., \boldsymbol{x}_M)$ of input vectors of dimension d, we want to produce a sequence of hidden feature vectors $(\boldsymbol{h}_1, ..., \boldsymbol{h}_M)$ (as for RNNs). While RNNs achieve this by sequential calculations, we strive for the following:

- the calculation of $\boldsymbol{h}_i$ should have access to all inputs $\boldsymbol{x}_j$
- the calculation of all $\boldsymbol{h}_i$ should be done in parallel

We have already seen in the attention mechanism of Section 2.3.1 how an output vector can have access to a sequence of input vectors. We can apply the same idea to provide information of the inputs $\boldsymbol{x}_j$ to the calculation of $\boldsymbol{h}_i$ as follows:

$$\boldsymbol{h}_i = \sum_j w_{ij} \cdot \boldsymbol{x}_j \in \mathbb{R}^d, \quad \text{where} \quad \sum_j w_{ij} = 1. \tag{2.14}$$

As before, the weights w_{ij} are called attention weights and determine how much each input $\boldsymbol{x}_j$ should contribute to the calculation of $\boldsymbol{h}_i$. To compute the attention weights, we again use a dot-product followed by a softmax for normalization

$$w_{ij} = \phi_{\text{softmax}}([\boldsymbol{x}_i(\boldsymbol{x}_j)^T]_{j=1}^M)_j. \tag{2.15}$$

The higher a weight w_{ij}, the more attention $\boldsymbol{x}_i$ puts on $\boldsymbol{x}_j$, and the more influence $\boldsymbol{x}_j$ has in the sum $\boldsymbol{h}_i$. We can think of the calculation of $\boldsymbol{h}_i$ as follows: position i is sending a query to all other positions j in form of its query vector $\boldsymbol{q}_i = \boldsymbol{x}_i$. Each position j has a key $\boldsymbol{k}_j$ and a value $\boldsymbol{v}_j$ and wants to send back its value based on how similar the query is to the key. In our case, the key and the value are both given by $\boldsymbol{x}_j$, i.e. $\boldsymbol{k}_j = \boldsymbol{v}_j = \boldsymbol{x}_j$. With the terminology of queries, keys and values, we can rewrite the calculation of $\boldsymbol{h}_i$ as

$$\boldsymbol{h}_i = \sum_j \phi_{\text{softmax}}(\boldsymbol{q}_i \boldsymbol{K}^T)_j \cdot \boldsymbol{v}_j, \tag{2.16}$$

where $\boldsymbol{K} = [\boldsymbol{x}_j]_{j=1}^M$ is the matrix with rows $\boldsymbol{x}_j$. Moreover, this method allows us to calculate all $\boldsymbol{h}_i$ in parallel using efficient matrix-multiplication

$$[\boldsymbol{h}_1, ..., \boldsymbol{h}_M] = \phi_{\text{softmax}}(\boldsymbol{Q}\boldsymbol{K}^T) \cdot \boldsymbol{V}, \tag{2.17}$$

where $\boldsymbol{Q} = [\boldsymbol{x}_i]_{i=1}^M$ is the matrix with rows $\boldsymbol{q}_i = \boldsymbol{x}_i$, $\boldsymbol{V} = [\boldsymbol{x}_j]_{j=1}^M$ is the matrix with rows $v_j = \boldsymbol{x}_i$ and ϕ_{softmax} is applied row-wise. As we use the input sequence $(\boldsymbol{x}_1, ..., \boldsymbol{x}_M)$ for queries, keys and values, the attention mechanism is called *self-attention*.

In conclusion, self-attention provides us parallelization across positions i and thus only requires $\mathcal{O}(1)$ sequential operations. Moreover, it gives every position i access to all other positions j, which enables the modeling of long-term dependencies. Self-attention thus mitigates the limitations of RNNs.

2.4.2 Scaled Dot-Product Attention

In the previous section, we calculated a sequence of hidden vectors as

$$[\boldsymbol{h}_1, ..., \boldsymbol{h}_M] = \phi_{\text{softmax}}(\boldsymbol{Q}\boldsymbol{K}^T) \cdot \boldsymbol{V}, \quad \boldsymbol{Q} = \boldsymbol{K} = \boldsymbol{V} = [\boldsymbol{x}_i]_{i=1}^M, \tag{2.18}$$

which we called self-attention. If we use arbitrary matrices $\boldsymbol{Q} \in \mathbb{R}^{M \times d_q}, \boldsymbol{K} \in \mathbb{R}^{N \times d_q}, \boldsymbol{V} \in \mathbb{R}^{N \times d_v}$ in Equation 2.18, we obtain the general attention calculation (which also covers the attention in Section 2.3.1).

The dot-product can grow large in magnitude for large dimensions d_q due to the following reason. If we assume that the elements of the query and key vector $\boldsymbol{q}, \boldsymbol{k} \in \mathbb{R}^{d_q}$ are independent random variables with mean 0 and variance 1, then the dot-product

$$\boldsymbol{q}\boldsymbol{k}^T = \sum_{i=1}^d \boldsymbol{q}_i \boldsymbol{k}_i \tag{2.19}$$

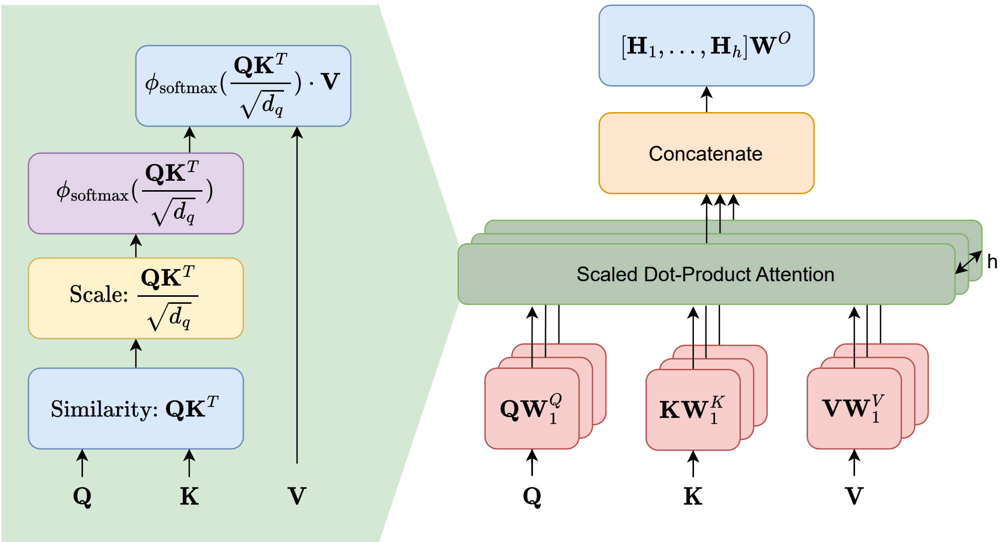


FIGURE 2.5: Left: Scaled Dot-Product Attention. Given a query matrix $\boldsymbol{Q}$ and key matrix $\boldsymbol{K}$, we use the dot-product to calculate the similarity between the rows of $\boldsymbol{Q}$ and $\boldsymbol{K}$. Subsequently, the elements of the resulting matrix are scaled and a softmax function is applied row-wise for normalization. The resulting attention weights are then used to weight the entries of the value matrix $\boldsymbol{V}$. Right: Multi-head Attention. The scaled dot-product attention is executed h times, each time with different linear projections of $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$. The h outputs of different heads are then concatenated and projected once again using a weight matrix $\boldsymbol{W}^O$.

has mean 0 and variance d_q. In order to reduce the variance, the authors propose to use a *scaled dot-product attention*

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \phi_{\text{softmax}}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_q}}) \cdot \boldsymbol{V}, \tag{2.20}$$

for matrices $\boldsymbol{Q} \in \mathbb{R}^{M \times d_q}, \boldsymbol{K} \in \mathbb{R}^{N \times d_q}, \boldsymbol{V} \in \mathbb{R}^{N \times d_v}$. The scaled dot-product attention calculation is depicted on the left side of Figure 2.5 and will be used for self-attention on the input sequence and cross-attention from the Transformer decoder to the encoder.

2.4.3 Multi-Head Attention

Instead of performing a single attention, Vaswani et al. (2017) proposed to perform the attention calculation h times, where the query, key and value vectors are first transformed using different, learned weight matrices $\boldsymbol{W}_s^Q \in \mathbb{R}^{d_{q_1} \times d_q}, \boldsymbol{W}_s^K \in \mathbb{R}^{d_k \times d_q}, \boldsymbol{W}_s^V \in \mathbb{R}^{d_{v_1} \times d_v}$. This allows the model to extract different features for queries, keys and values of the input before applying attention and enlarges its modeling capabilities. The different attention outputs are subsequently concatenated and further multiplied by yet another weight matrix $\boldsymbol{W}^O \in \mathbb{R}^{h \cdot d_v \times d_o}$. The full process is called multi-head attention and given as follows:

$$\text{MHA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = [\boldsymbol{h}_i^{(1)} \oplus ... \oplus \boldsymbol{h}_i^{(h)}]_{i=1}^M \cdot \boldsymbol{W}^O \in \mathbb{R}^{M \times d_o}, \tag{2.21}$$

$$\boldsymbol{H}_s = [\boldsymbol{h}_i^{(s)}]_{i=1}^M = \text{Attention}(\boldsymbol{Q}\boldsymbol{W}_s^Q, \boldsymbol{K}\boldsymbol{W}_s^K, \boldsymbol{V}\boldsymbol{W}_s^V) \in \mathbb{R}^{M \times d_v} \tag{2.22}$$

The multi-head attention calculation is depicted on the right side of Figure 2.5.

2.4.4 Position-wise Feed-forward Networks

The output of the multi-head attention $\text{MHA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ is a matrix, where each row is a linear transformation of the value vectors $\boldsymbol{v}_j$ in the value matrix $\boldsymbol{V} = [\boldsymbol{v}_j]_{j=1}^M$. In order to approximate non-linear functions, we need to introduce non-linearity. While this could be done by applying a non-linear activation function ϕ on the output of the multi-head attention module, the Transformer leverages a feed-forward neural network for a more powerful feature extraction. More specifically, the Transformer utilizes a two layer feed-forward network with ReLU activation function in between:

$$\text{FFN}(\boldsymbol{x}) = \max(0, \boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2, \tag{2.23}$$

where the same feed-forward network is applied for every position, i.e. row.

2.4.5 Transformer Blocks

We have now introduced all ingredients for defining a Transformer block. The Transformer block comes in two flavors, with self-attention and the additional possibility of cross-attention.

Transformer Block with Self-Attention

A Transformer block with self-attention obtains a matrix $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_M] \in \mathbb{R}^{M \times d}$ as input and produces an output matrix $\boldsymbol{H} = [\boldsymbol{h}_1, ..., \boldsymbol{h}_M] \in \mathbb{R}^{M \times d}$ as depicted on the left side of Figure 2.6 and explained below.

The input matrix $\boldsymbol{X}$ is first fed into the multi-head attention module, where $\boldsymbol{Q} = \boldsymbol{K} = \boldsymbol{V} = \boldsymbol{X}$, i.e. we use self-attention. Before we apply the feed-forward network for introducing non-linearity, we utilize a residual connection (He et al., 2016) followed by layer normalization (Ba et al., 2016).

Residual connection: For a function $f : \mathbb{R}^n \to \mathbb{R}^n$ and vector $\boldsymbol{x} \in \mathbb{R}^n$, a residual connection is given by $\boldsymbol{x} + f(\boldsymbol{x})$, i.e. we add $\boldsymbol{x}$ and $f(\boldsymbol{x})$. Residual connections allow the input $\boldsymbol{x}$ to bypass the function f and has been introduced in order to train deep neural networks with many layers. Note that the residual connection requires the function f to have the same input and output dimension n.

Layer normalization: For a vector $\boldsymbol{x} = [x_1, ..., x_n]$, layer normalization is defined as

$$\text{Norm}(\boldsymbol{x}) = \boldsymbol{\gamma} \odot \frac{(\boldsymbol{x} - \mu)}{\sigma} + \boldsymbol{\beta}, \quad \mu = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}, \tag{2.24}$$

where $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^n$ are learnable weights and $\odot$ denotes element-wise multiplication. Layer normalization normalizes different inputs to a similar distribution, which aids in learning of the neural network.

After applying multi-head attention, residual connection and layer normalization, we obtain a matrix

$$\boldsymbol{H}' = \text{Norm}(\boldsymbol{X} + \text{MHA}(\boldsymbol{X}, \boldsymbol{X}, \boldsymbol{X})) \in \mathbb{R}^{M \times d}. \tag{2.25}$$

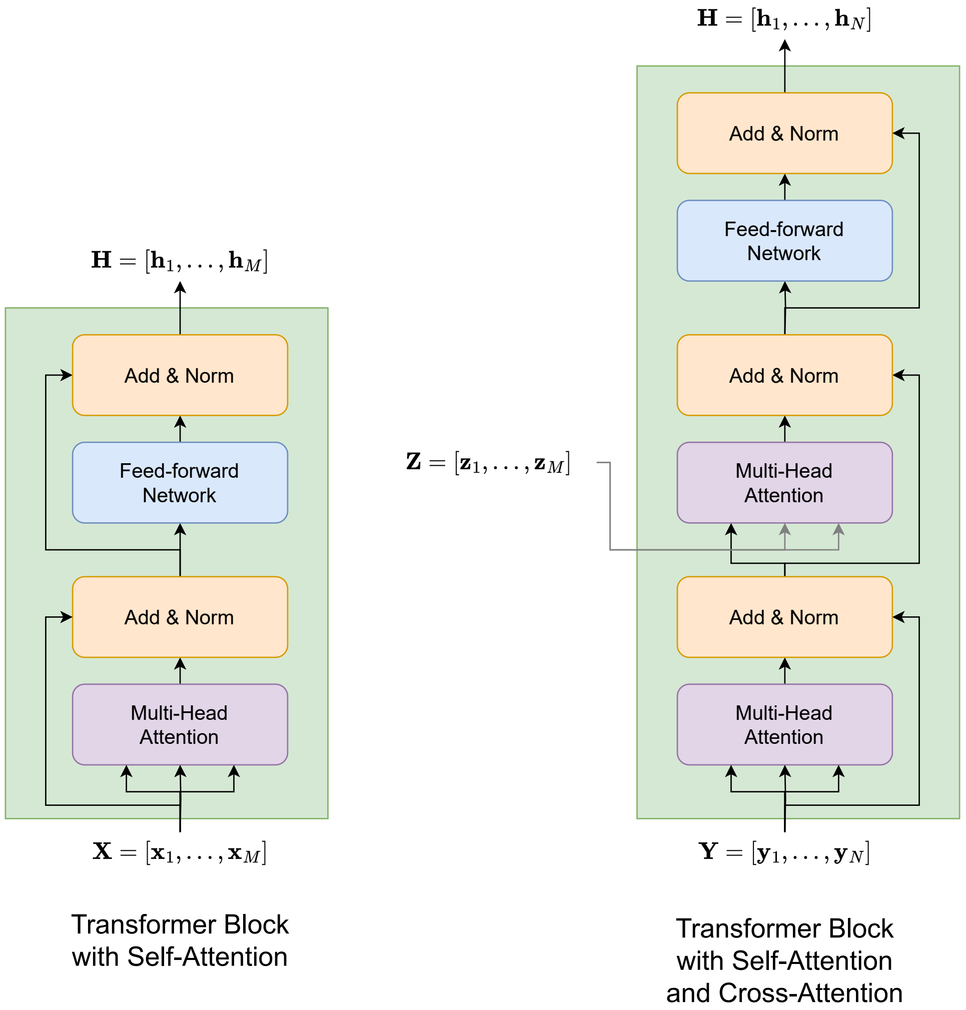


FIGURE 2.6: Left: Transformer block with self-attention obtains a sequence of vectors as input (stacked in a matrix $\boldsymbol{X}$) and produces a sequence of hidden vectors stacked in a matrix $\boldsymbol{H}$. The Transformer block starts with a multi-head attention where queries, keys and values are given by the matrix $\boldsymbol{X}$. Subsequently, a residual connection (Add) and layer normalization (Norm) is applied. The output is fed into a feed-forward neural network for non-linear function approximation. Right: the Transformer block with self-attention and cross-attention follows a similar procedure as with only self-attention. Nevertheless, a second sequence of vectors stacked in a matrix $\boldsymbol{Z}$ is given as input. $\boldsymbol{Z}$ is incorporated in the second multi-head attention module, where $\boldsymbol{Z}$ serves for both the key and value.

Each row vector of $\boldsymbol{H}'$ is then fed into the feed-forward network FFN as explained in Section 2.4.4 and another residual connection followed by layer normalization is applied subsequently, leading to the final output

$$\boldsymbol{H} = \mathrm{Norm}(\boldsymbol{H}' + \mathrm{FFN}(\boldsymbol{H}')) \in \mathbb{R}^{M \times d}, \tag{2.26}$$

where the weights in the FFN have dimensions $\boldsymbol{W}_1 \in \mathbb{R}^{d_1 \times d}, \boldsymbol{W}_2 \in \mathbb{R}^{d \times d_1}, \boldsymbol{b}_1 \in \mathbb{R}^{d_1}, \boldsymbol{b}_2 \in \mathbb{R}^d$. The dimension d_1 is typically chosen to be larger than d in order to memorize information.

Transformer Block with Self-Attention and Cross-Attention

A Transformer block with self-attention and cross-attention obtains a matrix $\boldsymbol{Y} = [\boldsymbol{y}_1, ..., \boldsymbol{y}_N] \in \mathbb{R}^{N \times d}$ and $\boldsymbol{Z} = [\boldsymbol{z}_1, ..., \boldsymbol{z}_M] \in \mathbb{R}^{M \times d}$ as input and produces an output matrix $\boldsymbol{H} = [\boldsymbol{h}_1, ..., \boldsymbol{h}_N] \in \mathbb{R}^{N \times d}$ as depicted on the right side of Figure 2.6 and explained below.

The matrix $\boldsymbol{Y}$ is first fed into the multi-head attention module, where $\boldsymbol{Q} = \boldsymbol{K} = \boldsymbol{V} = \boldsymbol{Y}$, i.e. we apply self-attention. Afterwards, a residual connection is applied followed by layer normalization, which leads to a matrix

$$\tilde{\boldsymbol{H}} = \mathrm{Norm}(\boldsymbol{Y} + \mathrm{MHA}(\boldsymbol{Y}, \boldsymbol{Y}, \boldsymbol{Y})) \in \mathbb{R}^{N \times d}. \tag{2.27}$$

In contrast to the Transformer block with only self-attention, we additionally incorporate the matrix $\boldsymbol{Z}$ into the calculation by applying another multi-head attention, where $\boldsymbol{Q} = \tilde{\boldsymbol{H}}, \boldsymbol{K} = \boldsymbol{V} = \boldsymbol{Z}$, which is called cross-attention. This results in a matrix

$$\boldsymbol{H}' = \mathrm{Norm}(\tilde{\boldsymbol{H}} + \mathrm{MHA}(\tilde{\boldsymbol{H}}, \boldsymbol{Z}, \boldsymbol{Z})) \in \mathbb{R}^{N \times d}. \tag{2.28}$$

We note that the row vectors of $\boldsymbol{Z}$ and $\tilde{\boldsymbol{H}}$ need to have the same dimension since we use a scaled dot-product between them. Analogous to the Transformer block with only self-attention, we then feed $\boldsymbol{H}'$ into a feed-forward network FFN as explained in Section 2.4.4 and another residual connection followed by layer normalization is applied subsequently, leading to the final output matrix

$$\boldsymbol{H} = \mathrm{Norm}(\boldsymbol{H}' + \mathrm{FFN}(\boldsymbol{H}')) \in \mathbb{R}^{N \times d}. \tag{2.29}$$

In summary, the Transformer block can process a sequence of vectors in parallel due to the (self-)attention mechanism and efficient matrix-matrix multiplication. Moreover, (self-)attention simplifies the learning of long-term dependencies between input vectors compared to RNNs. The Transformer block leverages a feed-forward neural network for producing non-linear functions and utilizes layer normalization and residual connections for better learning stability.

Transformer blocks can be stacked together analogously to the layers of a feed-forward neural network, where each block has its dedicated weights for the multi-head attention, feed-forward network and layer normalization. The output of one Transformer block becomes the input to the subsequent Transformer block as depicted in Figure 2.7. We remark that the input and output dimension of a Transformer block coincide due to the residual connections.

2.4.6 Positional Encoding

The Transformer block treats every input $\boldsymbol{x}_i$ in the same way, regardless of the position i. Nevertheless, positional information can be important, for instance when treating a sequence of words in a sentence. If we incorporate the position $i \in \mathbb{N}$ directly, this becomes unbounded and large for long sequences. Alternatively, we can normalize the positions by dividing by the sequence length N. Nevertheless, the positional encoding will then dependent on the specific sequence length.

Instead, the authors propose a positional vector as follows. Let d be the dimension of input vector $\boldsymbol{x}$. For a position $i \in \mathbb{N}$, the positional encoding $\boldsymbol{p}^{(i)} \in \mathbb{R}^d$ is defined as

$$\boldsymbol{p}^{(i)} = [\sin(w_0 i), \cos(w_0 i), ..., \sin(w_k i), \cos(w_k i), ..., \sin(w_{d/2-1} i), \cos(w_{d/2-1} i)], \tag{2.30}$$

$$w_k = \frac{1}{10000^{2k/d}} \tag{2.31}$$

The positional encoding has the following properties. Firstly, the elements of $\boldsymbol{p}^{(i)}$ are bounded between -1 and 1 due to the boundedness of sin and cos. Secondly, the elements of the positional encoding are sin and cos functions of increasing periods

$$2\pi, 2\pi \cdot 10000^{2/d}, 2\pi \cdot 10000^{4/d}, ..., 2\pi \cdot 10000^{1-2/d}, \tag{2.32}$$

which means that the positional encodings are unique as long as the length of the sequence is smaller than $2\pi \cdot 10000^{1-2/d} \approx 2\pi \cdot 10000$.

Furthermore, the positional encodings have the same length and the distance is symmetric for $n \in \mathbb{N}$ (see Appendix Theorem 2), i.e.

$$||p^{(i)}||_2 = \sqrt{\frac{d}{2}}, \quad ||p^{(i)} - p^{(i+n)}||_2 = ||p^{(i)} - p^{(i-n)}||_2, \quad \forall n \in \mathbb{N}. \tag{2.33}$$

Lastly, for any offset $n \in \mathbb{N}$, $\boldsymbol{p}^{(i+n)}$ can be represented as a linear function of $\boldsymbol{p}^{(i)}$

$$\boldsymbol{p}^{(i+n)} = \boldsymbol{M}^{(n)}\boldsymbol{p}^{(i)}, \tag{2.34}$$

where $\boldsymbol{M}^{(n)}$ depends on n but not on i, see Appendix Theorem 3. According to Vaswani et al. (2017), this linear relationship allows the model to easily learn to attend by relative positions. The positional encoding is added to the input $\boldsymbol{X}$ of the Transformer block, i.e. the new input becomes

$$[\boldsymbol{x_i} + \boldsymbol{p}^{(i)}]_{i=1}^{M}. \tag{2.35}$$

If we use a stack of Transformer blocks, the positional encoding is only added to the input of the first Transformer block but not subsequent blocks. This is because the residual connections already add the positional encoding to subsequent blocks.

2.4.7 Transformer Encoder-Decoder

The Transformer architecture was originally proposed for the task of machine translation. Analogously to the encoder-decoder RNN that we already saw, an encoder-decoder Transformer is composed of two Transformer architectures.

Encoder

Given an input vector sequence $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_M] \in \mathbb{R}^{M \times d}$ viewed as a matrix $\boldsymbol{X}$, the Transformer encoder adds positional encoding as explained in Section 2.4.6. The resulting matrix $[\boldsymbol{x_i} + \boldsymbol{p}^{(i)}]_{i=1}^{M}$ is fed into a stack of Transformer blocks with self-attention, which produces a matrix $\boldsymbol{Z} = [\boldsymbol{z}_1, ..., \boldsymbol{z}_M] \in \mathbb{R}^{M \times d}$ as depicted on the left side of Figure 2.7. We emphasize again that in contrast to the RNN, the self-attention mechanism allows us to calculate all vectors $\boldsymbol{z}_1, ..., \boldsymbol{z}_M$ in parallel.

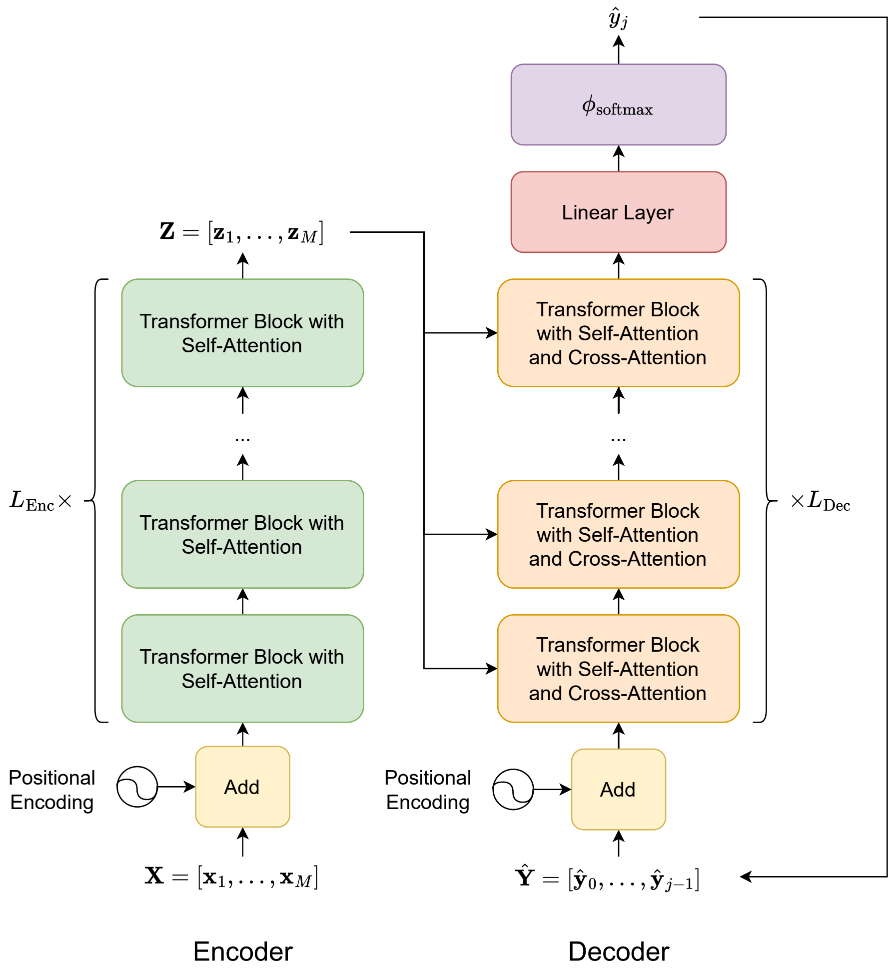


FIGURE 2.7: The Transformer Encoder-Decoder architecture. Left: the encoder is composed of a stack of L_{Enc} Transformer blocks. The input to the first Transformer block is summed with positional encodings that provide positional information. The output $\boldsymbol{Z}$ of the encoder will be leveraged by the decoder through attention. Right: the decoder generates the outputs $\hat{y}_j$ auto-regressively, one element at a time, as in the RNN decoder. In every step, the input to the decoder is a sequence of features vectors of the previous decoder predictions, stacked in a matrix $\boldsymbol{Y}$.

Decoder

The goal of the decoder is to leverage the encoded feature vectors in $\boldsymbol{Z}$ in order to produce the output sequence $y_1, ..., y_N$, one element at a time. In every decoding step $j = 1, ..., N$, the input to the Transformer decoder is a matrix $\hat{\boldsymbol{Y}} = [\hat{\boldsymbol{y}}_0, ..., \hat{\boldsymbol{y}}_{j-1}] \in \mathbb{R}^{j \times d}$ that consists of feature vectors for the previous predictions $\hat{y}_0, ..., \hat{y}_{j-1}$, where $\hat{\boldsymbol{y}}_0$ is initialized as a vector of zeros or a dedicated vector indicating the start of the decoding process for instance. As for the encoder, positional encoding is leveraged to produce a matrix $[\hat{\boldsymbol{y}}_i + \boldsymbol{p}^{(i)}]_{i=1}^{j-1}$. This matrix, together with $\boldsymbol{Z}$, becomes the input to a stack of Transformer blocks with self-attention and cross-attention as depicted on the right side of Figure 2.7. This results in a matrix $\hat{\boldsymbol{H}} = [\hat{\boldsymbol{h}}_0, ..., \hat{\boldsymbol{h}}_{j-1}]$. The final output $\hat{\boldsymbol{y}}_j^{\text{out}}$ is then obtained by feeding $\hat{\boldsymbol{h}}_{j-1}$ into a linear layer followed by a softmax activation function ϕ_{softmax} as depicted in Figure 2.7. We can then sample an output $\hat{y}_j$ and the decoding process continues.

2.5 Neural Network Optimization

After discussing several neural network architectures with neural network parameters $\boldsymbol{\theta}$, we require a method for optimization. At the heart of the optimization lies a so-called loss function

$$\mathcal{L} : \mathbb{R}^d \to \mathbb{R}, \quad \boldsymbol{\theta} \mapsto \mathcal{L}(\boldsymbol{\theta}), \tag{2.36}$$

which maps neural network parameters to a real-valued number. The goal is to find a global minimum $\boldsymbol{\theta}_{\text{min}}$ of $\mathcal{L}$. The minimization of $\mathcal{L}$ is supposed to be equivalent to finding the neural network weights that solve our task at hand. Consequently, the appropriate definition of $\mathcal{L}$ is crucial. The two most commonly used loss functions are cross-entropy and mean-squared error, which are used for the task of classification and regression, respectively. Given a dataset $\mathcal{D} = \{(x_i^{\text{in}}, y_i^{\text{out}})\}_{i=1}^N$, we have:

Classification: let the task be given by classification, i.e. every input x_i^{in} has to be classified into one of N_{cat} categories $y_1, ..., y_{N_{\text{cat}}}$. Let $f(x; \boldsymbol{\theta}) = (p(y_1 \mid x; \theta), ..., p(y_{N_{\text{cat}}} \mid x; \theta))$ be the neural network output probabilities as for instance obtained by ϕ_{softmax}. The goal is to find $\boldsymbol{\theta}^*$ that maximizes the likelihood of the data, i.e.

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{N} p(y_i^{\text{out}} \mid x_i^{\text{in}}; \boldsymbol{\theta}) \tag{2.37}$$

$$= \arg\min_{\boldsymbol{\theta}} - \sum_{i=1}^{N} \log p(y_i^{\text{out}} \mid x_i^{\text{in}}; \boldsymbol{\theta}). \tag{2.38}$$

The loss $\mathcal{L}(\mathcal{D}; \boldsymbol{\theta}) = -\sum_{i=1}^N \log p(y_i^{\text{out}} \mid x_i^{\text{in}}; \boldsymbol{\theta})$ is referred to as cross-entropy loss.

Regression: let the task be given by regression, i.e. $y_i^{\text{out}} \in \mathbb{R}$. We can minimize the error between the neural network output $f(x_i^{\text{in}}; \boldsymbol{\theta})$ and y_i^{out} using the mean-squared error loss

$$\mathcal{L}(\mathcal{D}; \boldsymbol{\theta}) = \sum_{i=1}^{N} (f(x_i^{\text{in}}; \boldsymbol{\theta}) - y_i^{\text{out}})^2. \tag{2.39}$$

2.5.1 Gradient Descent

For a given function $f : \mathbb{R}^d \to \mathbb{R}$, we want to find a solution $\boldsymbol{x}^*$ such that $f(\boldsymbol{x}^*)$ is minimal. Taylor's theorem (Marsden, 2012) shows that we can approximate f around a point $\boldsymbol{x} \in \mathbb{R}^d$ with the first-order Taylor series

$$f(\boldsymbol{x} + \Delta\boldsymbol{x}) \approx f(\boldsymbol{x}) + \Delta\boldsymbol{x}^T \nabla f|_{\boldsymbol{x}}, \tag{2.40}$$

where $\Delta\boldsymbol{x} \in \mathbb{R}^d$ is small enough. Our aim is to find $\Delta\boldsymbol{x}$, such that $f(\boldsymbol{x} + \Delta\boldsymbol{x}) \leq f(\boldsymbol{x})$. This can be achieved by choosing $\Delta\boldsymbol{x} = -\eta \nabla f|_{\boldsymbol{x}}$, for a small $\eta \in \mathbb{R}$, since $(\nabla f|_{\boldsymbol{x}})^T \nabla f|_{\boldsymbol{x}} \geq 0$. We thus obtain an algorithm that starts with an arbitrary $\boldsymbol{x}_0$ and updates according to

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 - \eta \nabla f|_{\boldsymbol{x}_0}. \tag{2.41}$$

The resulting algorithm is called optimization by gradient descent. On the one hand, we want a small η so that the first-order Taylor series provides a good approximation. On the other hand, a small η will result in slow convergence. Moreover, we can observe that using the gradient can potentially result in a local instead of global optimum of f. Gradient descent is guaranteed to converge to a local minima for an appropriate update schedule for η (Bertsekas and Tsitsiklis, 1999).

Gradient descent is the core algorithm for training neural networks with parameters $\boldsymbol{\theta}$. Given neural network parameters $\boldsymbol{\theta}_t$, we obtain new parameters $\boldsymbol{\theta}_{t+1}$ given by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{g}_t, \tag{2.42}$$

where $\boldsymbol{g}_t = \nabla \mathcal{L}|_{\boldsymbol{\theta}_t}$, $\mathcal{L}$ denotes a loss function (e.g. cross-entropy or mean-squared error) and η is referred to as the learning rate. In the following, we provide some advanced optimization versions of gradient descent that accelerate learning in practice.

2.5.2 Momentum

Gradient descent has issues navigating areas where the surface curves are much steeper in one direction than another. The gradient then oscillates in one direction while only making small steps in the other direction (Ruder, 2016). Momentum (Qian, 1999) mitigates this issue by accelerating the gradient in consistent directions while slowing down the gradient in directions where oscillations occur. For a hyperparameter β, the update is given by

$$\boldsymbol{m}_t = \beta \boldsymbol{m}_{t-1} + \boldsymbol{g}_t \tag{2.43}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{m}_t \tag{2.44}$$

We can see that if the entries of $\boldsymbol{g}_t$ and $\boldsymbol{m}_{t-1}$ have the same sign (e.g. for small gradients in the same direction), the update is increased, whereas it is decreased for the case of different signs (e.g. for oscillations).

2.5.3 Adagrad

The updates using the gradient highly depend on the magnitude of the gradient entries. If an entry of the gradient has very high values, we need to reduce the learning rate η in order to not diverge. In contrast, if an entry of the gradient has very small values, we require higher learning rates for faster convergence. The Adagrad optimizer (Duchi et al., 2011) solves this issue by introducing an adaptive learning rate for every entry of the parameter vector θ_t. For a hyperparameter β the update is given by

$$\boldsymbol{v}_t = \beta \boldsymbol{v}_{t-1} + (1-\beta) \boldsymbol{g}_t \odot \boldsymbol{g}_t, \tag{2.45}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{\boldsymbol{g}_t}{\sqrt{\boldsymbol{v}_t} + \epsilon}, \tag{2.46}$$

where $\epsilon > 0$ is a small constant and the vector division is element-wise.

2.5.4 Adam

Adaptive Moment Estimation (Adam) (Kingma and Ba, 2015) combines the strengths of momentum and adaptive learning rates as in Adagrad. Adam starts off by calculating an exponential average of the gradients as well as squared gradients as in momentum and Adagrad

$$\boldsymbol{m}_0 = \boldsymbol{0}, \tag{2.47}$$
$$\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t, \tag{2.48}$$
$$\boldsymbol{v}_0 = \boldsymbol{0}, \tag{2.49}$$
$$\boldsymbol{v}_t = \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t \odot \boldsymbol{g}_t. \tag{2.50}$$

Typically the values of β_1, β_2 are chosen to be close to 1 since this gives smoother estimates that are not highly governed by the current noisy gradient $\boldsymbol{g}_t$. Nevertheless, the estimates of $\boldsymbol{m}_t$ and $\boldsymbol{v}_t$ are biased towards the initial values for small t and values of β_1, β_2 close to 1. For instance, if $\beta_1 = 0.9$, we have

$$\boldsymbol{m}_1 = 0.9\boldsymbol{m}_0 + 0.1\boldsymbol{g}_1 = 0.1\boldsymbol{g}_1, \quad \boldsymbol{m}_2 = 0.9 \cdot 0.1\boldsymbol{g}_1 + 0.1\boldsymbol{g}_2, \tag{2.51}$$

and more generally

$$\boldsymbol{m}_t = \beta_1^t \boldsymbol{m}_0 + (1-\beta_1)\sum_{i=0}^{t-1} \beta_1^i \boldsymbol{g}_{t-i} = (1-\beta_1)\sum_{i=0}^{t-1} \beta_1^i \boldsymbol{g}_{t-i}. \tag{2.52}$$

If we want to reduce the influence of $\boldsymbol{m}_0 = \boldsymbol{v}_0 = \boldsymbol{0}$ for small t and β_1, β_2 close to 1, we can introduce a bias correction term as follows

$$\hat{\boldsymbol{m}}_t = \frac{\boldsymbol{m}_t}{1-\beta_1^t}, \tag{2.53}$$
$$\hat{\boldsymbol{v}}_t = \frac{\boldsymbol{v}_t}{1-\beta_2^t}, \tag{2.54}$$

which counteracts the decreasing factor of $1-\beta_1$ and $1-\beta_2$ for small values of t. Another way to derive the bias correction terms $1-\beta_1^t$ and $1-\beta_2^t$ is the following. Imagine we have a noisy loss function $\mathcal{L}(\boldsymbol{\theta})$ and we are interested in minimizing $\mathbb{E}[\mathcal{L}(\boldsymbol{\theta})]$. The stochasticity in $\mathcal{L}$ can arise for instance if we do not use all data points $(x^{\text{in}}, y^{\text{out}})$ in our dataset $\mathcal{D}$, but sample a subset $\mathcal{D}_{\text{sub}} \subsetneq \mathcal{D}$ in every optimization step (see Section 2.6). Let us denote by $\mathcal{L}_t$ a realization of $\mathcal{L}$ at optimization step t and let $\boldsymbol{g}_t = \nabla_{\boldsymbol{\theta}}\mathcal{L}_t$. We estimate the expected gradient $\mathbb{E}[\boldsymbol{g}_t]$ using an exponential weighted average

$$\boldsymbol{m}_0 = \boldsymbol{0}, \tag{2.55}$$
$$\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t. \tag{2.56}$$

In order to estimate the difference between our expected weighted average and the true expectation, we can calculate

$$\mathbb{E}[\boldsymbol{m}_t] = \mathbb{E}[(1-\beta_1)\sum_{i=0}^{t-1}\beta_1^i \boldsymbol{g}_{t-i}] \tag{2.57}$$

$$= (1-\beta_1)\sum_{i=0}^{t-1}\beta_1^i \mathbb{E}[\boldsymbol{g}_{t-i}]. \tag{2.58}$$

If the expectation $\mathbb{E}[\boldsymbol{g}_i]$ is stationary, we obtain the bias correction term $1-\beta_1^t$ since

$$\mathbb{E}[\boldsymbol{m}_t] = (1-\beta_1)\sum_{i=0}^{t-1}\beta_1^i \mathbb{E}[\boldsymbol{g}_t] = (1-\beta_1^t)\mathbb{E}[\boldsymbol{g}_t]. \tag{2.59}$$

An analogous derivation holds for $\boldsymbol{v}_t$. In conclusion, the update of neural network parameters $\boldsymbol{\theta}_t$ according to Adam is given by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t}+\epsilon}, \tag{2.60}$$

where $\epsilon > 0$ is a small constant and the vector division is element-wise.

Algorithm 1 Neural Network Training Procedure

Require: dataset $\mathcal{D}$, neural network $f(\cdot;\boldsymbol{\theta})$, optimizer Ω, loss function $\mathcal{L}(\mathcal{D};\boldsymbol{\theta})$, number of mini-batches N_b, number of epochs K

```
Initialize neural network weights θ randomly
for epoch= 1, ..., K do
    Shuffle D
    Split D into N_b mini-batches D_batch,1, ..., D_batch,N_b
    for n = 1, ..., N_b do
        Calculate g = ∇_θ L(D_batch,n; θ)
        Update θ using optimizer Ω und gradient g
    end for
end for
```

2.6 Neural Network Training Procedure

In this section, we give a brief overview of how neural networks are being trained. Given a dataset $\mathcal{D} = \{(x_i^{\text{in}}, y_i^{\text{out}})\}_{i=1}^N$, we perform the following decisions before starting the training. We first decide on a neural network architecture to use that is appropriate for the given data, e.g. a feed-forward network or Transformer. Secondly, the neural network weights $\boldsymbol{\theta} \in \mathbb{R}^d$ are initialized. This is typically done based on a normal distribution or uniform distribution (Glorot and Bengio, 2010; He et al., 2015). Afterwards, we select an optimizer with learning rate η for updating the neural network weights, e.g. Adagrad or Adam. Lastly, we select a loss function $\mathcal{L}(\mathcal{D};\boldsymbol{\theta})$ that we want to minimize, e.g. cross-entropy loss or mean-squared error loss.

Since our goal is to minimize the loss $\mathcal{L}(\mathcal{D};\boldsymbol{\theta})$, it seems natural to leverage the entire data $\mathcal{D}$ at once for optimization, i.e. update the weights $\boldsymbol{\theta}$ using the gradient $\boldsymbol{g} = \nabla_{\boldsymbol{\theta}}\mathcal{L}(\mathcal{D};\boldsymbol{\theta})$.

Nevertheless, $\mathcal{D}$ can be large, which makes it very slow to calculate the gradients or even impossible due to memory constraints. In practice, we instead use a subset $\mathcal{D}_{\text{batch}} \subsetneq \mathcal{D}$, also referred to as mini-batch, and update the neural network weights using the gradient $\boldsymbol{g} = \nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}_{\text{batch}}; \boldsymbol{\theta})$. This approach offers the benefit of greater computational efficiency while still providing a good approximation of the complete dataset. In addition, it results in faster convergence, since instead of using the entire dataset for one update, we can split it into N_b mini-batches and perform N_b consecutive updates. Furthermore, employing mini-batches can enhance convergence towards an improved solution compared to utilizing the entire dataset, as the diverse mini-batches assist in escaping local minima.

The neural network training is performed in a number of K epochs. In every epoch, the dataset $\mathcal{D}$ is first shuffled and split into N_b mini-batches. Afterwards, we traverse the list of mini-batches. For every mini-batch $\mathcal{D}_{\text{batch}}$, we calculate the gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}_{\text{batch}}; \boldsymbol{\theta})$ and update the neural network weights using our chosen optimizer. The epoch ends once every mini-batch has been used, i.e. after all the data in $\mathcal{D}$ has been leveraged. A summary of the training procedure is provided in Algorithm 1.

In the following chapter, we discuss reinforcement learning where there is no dataset to begin with, but data is produced through interaction with an environment. In this case, the neural network is updated after a certain amount of interactions. The data for updating either comes from the interaction data that was collected since the last neural network update or from a so-called replay buffer that acts as a dataset of previous interactions.

2.7 Conclusion

This chapter introduced the fundamental concepts of deep learning. We started by defining neural network architectures such as feed-forward neural networks, RNNs and Transformers for function approximation. While feed-forward networks process a single input vector, RNNs and Transformers can process a sequence of inputs and produce a sequence of outputs. The Transformer introduced the notion of self-attention to overcome limitations in the RNN such as parallelization and modeling of long-term dependencies among inputs. Afterwards, we discussed the optimization of neural networks based on gradient descent and more advanced methods such as Adam and concluded with a section about the neural network training procedure.

Chapter 3

Reinforcement Learning

In reinforcement learning (RL), an agent interacts with an environment in discrete time steps (Sutton et al., 1999). In every step, the agent is located in a specific state, takes an action and observes a reward signal. Both state information and reward signal are emitted by the environment. A visualized representation can be observed in Figure 3.1. The action selection of the agent is driven by a policy, which defines a mapping from states to probability distributions over actions. The goal of the agent is to select actions in order to maximize the total sum of reward. The reward signal in every turn can be interpreted as reinforcing or penalizing the taken action. Yet, reinforcement learning is a sequential-decision making task and sacrificing immediate reward can move the agent into states that provide higher reward in the long run. The reward is hence not a signal that tells how to act optimally in every turn but the whole sequence of states, actions, and rewards has to be taken into account to evaluate the decisions. The reward signal encodes what should be achieved but not necessarily how. This puts apart reinforcement learning from supervised learning that informs the system in every state with the labeled action that should be taken. Instead, a reinforcement learning agent obtains information about how to act best in every state by interacting with the environment in a trial-and-error process. Moreover, compared to supervised learning that expects a data set to begin with, the RL agent produces the data itself by interacting with the environment.

3.1 Markov Decision Process

To tackle the reinforcement learning task in a precise way, let us state it more formally in the following. The reinforcement learning problem is defined by a Markov decision process (MDP) $\mathcal{M} =< \mathcal{S}, \mathcal{A}, r, p, p_0, \gamma >$ (Puterman, 1994), where

- $\mathcal{S}$ denotes the state space, i.e. the set of all states that the agent can visit.
- $\mathcal{A}$ denotes the action space defining all possible actions the agent can take.
- $r : S \times \mathcal{A} \to \mathbb{R}$ denotes the reward function, which maps a tuple (s, a) of state and action to a real value $r(s, a)$.
- $p(s'|s, a)$ is the transition probability function and models the probability of transitioning to state s' after executing action a in state s.
- $p_0(s)$ is the starting state probability and gives the probability of starting in state s.
- γ denotes the discount factor.

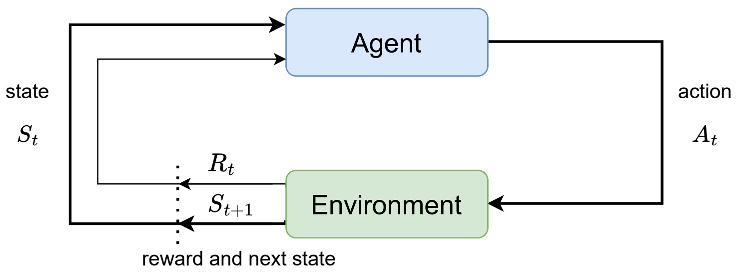


FIGURE 3.1: Interaction of the RL agent with its environment. In every step, the environment emits a state S_t upon which the agent takes an action A_t. The agent then receives a reward R_t and the next state S_{t+1}. Figure taken from (Sutton and Barto, 2018).

The Markov property comes into play here as only the last state and action is necessary to determine the transition to the next state. The aim of the discount factor $\gamma \in [0,1]$ is to trade-off the importance of immediate and future rewards. The MDP describes the environment the agent lives in. Lastly, a policy $\pi(a|s)$ defines a probability distribution over actions, given states, i.e. a mapping

$$\pi : \mathcal{S} \times \mathcal{A} \to [0,1], \quad (s,a) \mapsto \pi(a|s). \tag{3.1}$$

The policy π fully determines the behavior of the agent. To summarize the process, at a given time step t, the agent observes a state s_t, chooses an action $a_t \sim \pi(\cdot|s_t)$, observes a reward signal $r_t = r(s_t, a_t) \in \mathbb{R}$ and transitions to a new state s_{t+1}. We emphasize that in this chapter, the letters s and a always refer to an element of the state space $\mathcal{S}$ and action space $\mathcal{A}$, respectively.

3.2 Return and Value Functions

In the last section we have defined the environment that the agent interacts with, which is given by the MDP. We now define what the goal of the agent is and introduce the notion of value functions that are fundamental in reinforcement learning. Informally, the goal of the agent is to maximize the discounted sum of rewards in expectation. To define it formally, we need the notion of discounted return.

Definition 1. *For a time step t and a (possibly infinite sequence) $S_t, A_t, R_t, S_{t+1}, A_{t+1}, R_{t+1}, \ldots$ of states, actions and rewards, we define the discounted return as*

$$G_t = \sum_{i \geq 0} \gamma^i R_{t+i}. \tag{3.2}$$

In episodic environments, there is a distinct terminal state that the agent arrives in after finitely many steps. This is the case in games and dialogue for instance. The discounted return will then become a finite sum $\sum_{i=0}^{T-1} \gamma^i R_{t+i}$ that ends with the terminal state S_T and is well-defined even in the case $\gamma = 1$. In contrast, continual or non-episodic environments have no terminal state and G_t is actually an infinite sum. A discount factor of $\gamma < 1$ then ensures that the sum is still well-defined (given the reward R_t is bounded).

The agent uses its policy π for taking an action in every state, which produces a trajectory of states, actions, and rewards

$$\tau = s_0, a_0, r_0, , s_1, a_1, r_1, ..., s_T. \tag{3.3}$$

with probability distributions

$$p_\pi(\tau) = p_0(s_0) \prod_{t=0}^{T-1} \pi(a_t|s_t) p(s_{t+1}|s_t, a_t), \tag{3.4}$$

$$p_\pi(\tau|s_0) = \prod_{t=0}^{T-1} \pi(a_t|s_t) p(s_{t+1}|s_t, a_t) \tag{3.5}$$

Definition 2. *The state-value function $V^\pi(s)$ for a policy π in state s is defined as the expected return when being in state s and following policy π, i.e.*

$$V^\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]. \tag{3.6}$$

The action-value function $Q^\pi(s, a)$ for a policy π in state s with action a is defined as the expected return when executing action a in state s and following policy π afterwards, i.e.

$$Q^\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]. \tag{3.7}$$

The advantage function $A^\pi(s, a)$ for a state s and action a is defined as

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s). \tag{3.8}$$

We use the notation $\mathbb{E}_\pi$ to indicate that the actions are sampled from π. The action-value function is also known as Q-function and we will use these two terms interchangeably. The state-value function and Q-function give a measure of how good it is to be in state s (and using action a) when following the policy π. The advantage function expresses how much advantage in terms of expected return we obtain when taking action a in state s compared to being in state s. The state-value function, Q-function and advantage function are related via

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_\pi[Q^\pi(s, A)], & (3.9)\\ Q^\pi(s, a) &= \mathbb{E}_\pi[R_t + \gamma V^\pi(S_{t+1})|S_t = s, A_t = a], & (3.10)\\ A^\pi(s, a) &= \mathbb{E}_\pi[R_t + \gamma V^\pi(S_{t+1}) - V^\pi(S_t)|S_t = s, A_t = a]. & (3.11) \end{aligned}$$

Equation 3.11 implies that we can obtain an estimate of the advantage function through $r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)$, i.e. only using the state-value function. Moreover, the expected advantage is

$$\mathbb{E}_\pi[A^\pi(s, A)] = \sum_a \pi(a|s) \cdot A(s, a) = \sum_a \pi(a|s) \cdot (Q^\pi(s, a) - V^\pi(s)) = V^\pi(s) - V^\pi(s) = 0. \tag{3.12}$$

The state-value and Q-function are used in order to define what it means to be optimal.

Definition 3. *A policy π_1 is considered better than another policy π_2 if*

$$V^{\pi_1}(s) \geq V^{\pi_2}(s), \quad \forall s \in \mathcal{S}. \tag{3.13}$$

The optimal state-value and action-value function are defined by

$$V^*(s) = \max_\pi V^\pi(s), \qquad Q^*(s,a) = \max_\pi Q^\pi(s,a). \tag{3.14}$$

A policy π is said to be optimal if $V^\pi(s) = V^*(s), \quad \forall s \in \mathcal{S}$.

The state-value function hence gives us a measurement of how good a policy is. The optimal state-value function defines the best possible performance in a MDP. While the functions V^* and Q^* are unique, this does not need to hold for the optimal policy.

3.3 Policy Evaluation

As we have now defined the measurement for how a policy can be evaluated, we require algorithms that can calculate the state-value function as well as the Q-function in practice.

3.3.1 Monte Carlo Estimation

We can obtain an estimate of Q^π or V^π using the so-called Monte Carlo method, which is given as follows. Let X be a random variable with distribution P. In order to obtain an estimate of the expectation $\mathbb{E}_{X\sim P}[X]$, we can produce n independent samples $x_1, ..., x_n$ of X and take an average

$$\bar{X}_n = \frac{x_1 + ... + x_n}{n}. \tag{3.15}$$

The estimate $\bar{X}_n$ is referred to as Monte Carlo estimate and is guaranteed to converge according to the law of large numbers (Rosenthal, 2006), i.e.

$$\lim_{n\longrightarrow\infty} \bar{X}_n = \mathbb{E}_{X\sim P}[X]. \tag{3.16}$$

Since the state-value and Q-function are also given by expectations, we can apply the Monte Carlo method to produce estimates via an average that will eventually converge in the limit. In the following we will concentrate on the state-value function but analogous methods hold for the Q-function. To approximate $V^\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$, we require for every state s an estimate $V(s)$. In order to obtain samples for the expectation, we produce trajectories using the policy π

$$\tau = s_0, a_0, r_0, , s_1, a_1, r_1, ..., s_T. \tag{3.17}$$

We can then calculate returns $g_t = \sum_{i\geq 0} \gamma^i r_{t+i}$ and update our current average $V(s_t)$ by

$$V(s_t) \leftarrow V(s_t) + \frac{1}{N(s_t)+1}(g_t - V(s_t)), \tag{3.18}$$

where $N(s_t)$ is the number of times the state s_t has been visited. It can be proven that the average eventually converges to the expected value, even if a state s occurs multiply times in the trajectory, which breaks the independence assumption (Bertsekas and Tsitsiklis, 1996). The corresponding algorithm is called Monte Carlo policy evaluation.

In practice, it is more common to track a running average using a learning rate $0 < \alpha < 1$ instead of the visitation count $N(s_t)$:

$$V(s_t) \leftarrow V(s_t) + \alpha(g_t - V(s_t)). \tag{3.19}$$

We hence update our average based on the difference between our current estimate and the observed outcome g_t.

3.3.2 General Strategy

The return g_t that we used above for updating the average is more generally called the target (in this case the Monte Carlo target). The general strategy for approximating the state-value function or Q-function is to move the approximations closer towards a target v_{target}

$$V(s_t) \leftarrow V(s_t) + \alpha(v_{\text{target}} - V(s_t)), \tag{3.20}$$

and different algorithms for approximating V^π can be distinguished by the target they use. In addition to the Monte-Carlo target, other prominent examples for targets are the TD-target and n-step return:

$$\text{TD-target:} \quad v_{\text{target}} = G_{t:t+1} = R_t + \gamma V(S_{t+1}) \tag{3.21}$$

$$n\text{-step Return:} \quad v_{\text{target}} = G_{t:t+n} = \sum_{i=0}^{n-1} \gamma^i R_{t+i} + \gamma^n V(S_{t+n}) \tag{3.22}$$

3.3.3 Off-Policy Evaluation

Instances of the above mentioned targets are obtained by generating trajectories using the policy π that we seek to evaluate, since the expectation is with respect to π. If samples for evaluating policy π are generated by π, this is referred to as on-policy evaluation. Yet, in many domains such as medicine or dialogue it might be not possible to run an arbitrary policy π due to the severity of the decisions or because producing trajectories is very costly. Instead, we can only generate trajectories using another policy μ that is maybe more safe or conservative. Our objective is then to evaluate the policy π, i.e. approximate Q^π or V^π, by using trajectories produced by μ, which is called off-policy evaluation. The policies π and μ are then called target policy and behavior policy, respectively.

We can obtain off-policy targets for π using the idea of importance sampling. Let X be a random variable and P_1, P_2 two probability distributions. We then have

$$\mathbb{E}_{X\sim P_1}[X] = \sum_x P_1(X=x)X = \sum_x P_2(X=x)\frac{P_1(X=x)}{P_2(X=x)}X = \mathbb{E}_{X\sim P_2}\left[\frac{P_1(X)}{P_2(X)}X\right]. \tag{3.23}$$

The weight $\frac{P_1(X=x)}{P_2(X=x)}$ is called importance sampling weight. In addition, for a trajectory

$$\tau = s_t, a_t, r_t, , s_{t+1}, a_{t+1}, r_{t+1}, ..., s_{t+n}. \tag{3.24}$$

we have

$$\frac{p_\pi(\tau|S_t = s_t)}{p_\mu(\tau|S_t = s_t)} = \prod_{j=0}^{n-1} \frac{\pi(a_{t+j}|s_{t+j})p(s_{t+j+1}|s_{t+j}, a_{t+j})}{\mu(a_{t+j}|s_{t+j})p(s_{t+j+1}|s_{t+j}, a_{t+j})} \tag{3.25}$$

$$= \prod_{j=0}^{n-1} \frac{\pi(a_{t+j}|s_{t+j})}{\mu(a_{t+j}|s_{t+j})}. \tag{3.26}$$

We can apply the above calculations for our targets in Section 3.3.2 to obtain off-policy versions $G_t^{\pi/\mu}$, $G_{t:t+1}^{\pi/\mu}$ and $G_{t:t+n}^{\pi/\mu}$, where the superscript π/μ indicates that this is an off-policy target:

$$v_{\text{target}} = G_t^{\pi/\mu} = \prod_{j\geq 0} \frac{\pi(A_{t+j}|S_{t+j})}{\mu(A_{t+j}|S_{t+j})} \cdot \sum_{i\geq 0} \gamma^i R_{t+i} \tag{3.27}$$

$$v_{\text{target}} = G_{t:t+1}^{\pi/\mu} = \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)} \cdot (R_t + \gamma V(S_{t+1})) \tag{3.28}$$

$$v_{\text{target}} = G_{t:t+n}^{\pi/\mu} = \prod_{j=0}^{n-1} \frac{\pi(A_{t+j}|S_{t+j})}{\mu(A_{t+j}|S_{t+j})} \cdot (\sum_{i=0}^{n-1} \gamma^i R_{t+i} + \gamma^n V(S_{t+n})) \tag{3.29}$$

Off-policy algorithms are typically more sample-efficient as they can train on experience generated by arbitrary policies multiple times, while being often more unstable due to higher variance. The higher variance arises due to accounting for the mismatch between target and behavior policy through the importance sampling weights.

V-trace Target

V-trace (Espeholt et al., 2018) is an off-policy method for approximating the state-value function V^π of a target policy π based on data generated by a behavior policy μ. V-trace addresses the issue of high variance arising from importance sampling through truncation.

Given a trajectory $\tau = s_t, a_t, r_t, ..., s_{t+n}$ generated by a behavior policy μ, the n-steps V-trace target v_t is defined by

$$v_t = V(s_t) + \sum_{i=t}^{t+n-1} \gamma^{t-i} (\prod_{j=t}^{i-1} c_j) \delta_i, \tag{3.30}$$

where $\delta_i = \rho_i[r_i + \gamma V(s_{i+1}) - V(s_i)]$ is a temporal difference term. Moreover, $\rho_i = \min(\overline{\rho}, \frac{\pi(a_i|s_i)}{\mu(a_i|s_i)})$ and $c_j = \min(\overline{c}, \frac{\pi(a_j|s_j)}{\mu(a_j|s_j)})$ are truncated importance sampling weights, where $\overline{\rho}$ and $\overline{c}$ are hyperparameters satisfying $\overline{\rho} \geq \overline{c}$.

In the on-policy case of $\pi = \mu$ and $\overline{c} \geq 1$, the V-trace target becomes the n-step return since

$$v_t = V(s_t) + \sum_{i=t}^{t+n-1} \gamma^{t-i} \delta_i = \sum_{i=t}^{t+n-1} \gamma^{t-i} r_i + \gamma^n V(s_{t+n}). \tag{3.31}$$

Before we mention the properties of the target, we build intuition about its derivation. For a state s_t, the expected off-policy n-step return conditioned on s_t is given by

$$\mathbb{E}_\mu[G^{\pi/\mu}_{t:t+n}|S_t = s_t] \tag{3.32}$$
$$= \mathbb{E}_\mu\left[\prod_{j=0}^{n-1}\frac{\pi(A_{t+j}|S_{t+j})}{\mu(A_{t+j}|S_{t+j})}\cdot(\sum_{i=0}^{n-1}\gamma^i R_{t+i} + \gamma^n V(S_{t+n}))\Big|S_t = s_t\right] \tag{3.33}$$
$$= \mathbb{E}_\mu\left[\prod_{j=0}^{n-1}\frac{\pi(A_{t+j}|S_{t+j})}{\mu(A_{t+j}|S_{t+j})}\cdot(\sum_{i=0}^{n-1}\gamma^i (R_{t+i} + V(S_{t+i}) - V(S_{t+i})) + \gamma^n V(S_{t+n}))\Big|S_t = s_t\right] \tag{3.34}$$
$$= \mathbb{E}_\mu\left[\prod_{j=0}^{n-1}\frac{\pi(A_{t+j}|S_{t+j})}{\mu(A_{t+j}|S_{t+j})}\cdot(V(S_t) + \sum_{i=0}^{n-1}\gamma^i (R_{t+i} + \gamma V(S_{t+i+1}) - V(S_{t+i})))\Big|S_t = s_t\right] \tag{3.35}$$
$$= V(s_t) + \sum_{i=0}^{n-1}\gamma^i\cdot\mathbb{E}_\mu\left[\prod_{j=0}^{n-1}\frac{\pi(A_{t+j}|S_{t+j})}{\mu(A_{t+j}|S_{t+j})}\cdot((R_{t+i} + \gamma V(S_{t+i+1}) - V(S_{t+i})))\Big|S_t = s_t\right] \tag{3.36}$$
$$= V(s_t) + \sum_{i=0}^{n-1}\gamma^i\cdot\mathbb{E}_\mu\left[\prod_{j=0}^{i}\frac{\pi(A_{t+j}|S_{t+j})}{\mu(A_{t+j}|S_{t+j})}\cdot((R_{t+i} + \gamma V(S_{t+i+1}) - V(S_{t+i})))\Big|S_t = s_t\right] \tag{3.37}$$
$$= V(s_t) + \mathbb{E}_\mu\left[\sum_{i=0}^{n-1}\gamma^i\cdot\prod_{j=0}^{i}\frac{\pi(A_{t+j}|S_{t+j})}{\mu(A_{t+j}|S_{t+j})}\cdot((R_{t+i} + \gamma V(S_{t+i+1}) - V(S_{t+i})))\Big|S_t = s_t\right] \tag{3.38}$$

We can thus obtain a sample of $\mathbb{E}_\mu[G^{\pi/\mu}_{t:t+n}|S_t = s_t]$ as

$$V(s_t) + \sum_{i=0}^{n-1}\gamma^i\cdot\prod_{j=0}^{i}\frac{\pi(A_{t+j}|S_{t+j})}{\mu(A_{t+j}|S_{t+j})}\cdot((R_{t+i} + \gamma V(S_{t+i+1}) - V(S_{t+i}))) \tag{3.39}$$
$$= V(s_t) + \sum_{i=0}^{n-1}\gamma^i\cdot\left(\prod_{j=0}^{i-1}\frac{\pi(A_{t+j}|S_{t+j})}{\mu(A_{t+j}|S_{t+j})}\right)\cdot\frac{\pi(A_{t+i}|S_{t+i})}{\mu(A_{t+i}|S_{t+i})}((R_{t+i} + \gamma V(S_{t+i+1}) - V(S_{t+i}))) \tag{3.40}$$

Since the importance sampling product exhibits a high variance, we truncate the importance weights by the hyperparameters $\overline{\rho}$ and $\overline{c}$, which then leads to the V-trace target.

The two truncation parameters $\overline{\rho}$ and $\overline{c}$ play different roles in the target. Firstly, $\overline{\rho}$ determines the value that the approximation converges to. More specifically, using the V-trace target for policy evaluation will not converge to V^π but to $V^{\pi_{\overline{\rho}}}$, where

$$\pi_{\overline{\rho}}(a|s) = \frac{\min(\overline{\rho}\mu(a|s), \pi(a|s))}{\sum_{a'}\min(\overline{\rho}\mu(a'|s), \pi(a'|s))}. \tag{3.41}$$

For large values of $\overline{\rho}$ the minimum is attained at $\pi(a|s)$ and the approximation converges to V^π since $\pi_{\overline{\rho}} = \pi$. On the other hand, for very small values of $\overline{\rho}$ the approximation converges towards V^μ.

In contrast, the parameter $\overline{c}$ has no influence on the value at convergence, but significant influence on the variance of the V-trace target since it truncates every element of the product

$$\prod_{j=0}^{i-1}\frac{\pi(A_{t+j}|S_{t+j})}{\mu(A_{t+j}|S_{t+j})}. \tag{3.42}$$

One strategy is to set a low value for $\bar{c}$ in order to reduce variance and a high value for $\bar{\rho}$ in order to converge towards V^{π}. The V-trace target will be leveraged in Chapter 6 and 7.

3.3.4 From Tables to Neural Networks

Until now we have maintained a table with an entry $V(s)$ (or $Q(s,a)$) for every state s (or pair (s,a)) that stored our current approximations. While this tabular approach provides us with convergence results in theory, it prohibits generalization across states and state-action pairs.

Instead of maintaining a table, deep RL uses a neural network for approximating the state-values or Q-values, which facilitates generalization in large state and action spaces. The neural network $V_{\boldsymbol{\theta}}$ with neural network parameters $\boldsymbol{\theta}$ is updated according to

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha(v_{\text{target}} - V_{\boldsymbol{\theta}}(s_t)) \cdot \nabla_{\boldsymbol{\theta}} V_{\boldsymbol{\theta}}(s_t), \tag{3.43}$$

i.e. the neural network solves a regression task with mean-squared error loss

$$\mathcal{L} = \frac{1}{2}(v_{\text{target}} - V_{\boldsymbol{\theta}}(s_t))^2. \tag{3.44}$$

We remark the resemblance with Equation 3.20. In fact, Equation 3.20 is a special case.

3.4 Policy Learning

In the last section, we described how the value function V^{π} or Q^{π} can be approximated for evaluating a policy π. In this section we will have a look at possible methods that can be used to find an optimal policy. The methods can be distinguished into value-based, policy-based, and actor-critic methods.

3.4.1 Value-based Methods

Value-based methods construct a policy indirectly through learning a Q-function $Q(s,a)$. The selected action in a state s is then given by

$$\pi(s) = \arg\max_a Q(s,a), \tag{3.45}$$

which constitutes a greedy action selection because we always sample according to the maximum value. Algorithms to find the optimal action-value function Q^* either use policy iteration or a version of Q-learning.

Policy Iteration

Policy iteration algorithms alternate between an *evaluation* step and an *improvement* step. They start with a randomly initialized Q-function and use the argmax operation as shown in Equation 3.45 to define the corresponding policy π. They then use policy evaluation as described in Section 3.3 to find the corresponding Q-function Q^{π}. The improvement step is performed by defining a new policy

$$\pi'(s) = \arg\max_a Q^{\pi}(s,a) \tag{3.46}$$

and the process repeats afterwards. The policy improvement theorem (see Appendix Corollary 5.1) ensures that the new policy π' is at least as good as the old one. Moreover, the optimal Q-function is found once the new policy is not better than before. During the generation of trajectories, actions are sampled with an ϵ-greedy approach to ensure exploration, i.e. with probability ϵ, a random action is taken.

Q-Learning

Different to alternating between estimating Q^π and acting greedily with respect to the estimate as done in policy iteration, Q-learning (Watkins and Dayan, 1992) approximates the optimal action-value function Q^* directly. Q-learning is based on the Bellman optimality equation (Bellman, 1958)

$$Q^*(s,a) = \mathbb{E}_{S_{t+1}\sim p(\cdot|s,a)}[R_t + \gamma \max_{a'} Q^*(S_{t+1}, a')|S_t = s, A_t = a]. \tag{3.47}$$

The corresponding update equation for the current Q-function Q is given by

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)). \tag{3.48}$$

Note that Q-learning is an off-policy algorithm and does not require importance sampling since the next action a' is taken by the target policy and the expectation in Equation 3.47 is only dependent on the probability distribution $p(\cdot|s,a)$. Analogous to policy iteration, during the generation of trajectories, actions are sampled with an ϵ-greedy approach.

Replay Buffer

During estimation of a value function, we generate a trajectory $\tau = s_0, a_0, r_0, ..., s_T$, calculate update targets and update our value function estimates. Afterwards, the trajectory is typically discarded and a new trajectory is produced for the next update. We thus leverage every trajectory only once, which is very sample-inefficient.

Instead of discarding the generated experience, we can store it in a so-called replay buffer (Lin, 1992). The strategy is then to generate a new trajectory, save it in our replay buffer, sample experience from the buffer and perform an update. In this way, we can reuse generated experience multiple times, which increases the sample efficiency. The process of updating our estimates based on sampling from a replay buffer is referred to experience replay. Experience replay is an important ingredient in off-policy algorithms since these algorithms can leverage data that has been generated by a different policy. Experience replay is in particular used by the deep Q-learning algorithm that we explain next.

Deep Q-Learning

Q-learning has been advanced to deep Q-learning (DQN) (Mnih et al., 2015) which leverages a deep neural network $Q_{\boldsymbol{\theta}}$ as function approximator. In order to facilitate sample efficiency and stability, DQN uses a replay buffer $\mathcal{D}$ (Lin, 1992) that stores generated experience (s, a, r, s') of states, actions, rewards, and next states. Since Q-learning is an off-policy algorithm, we can leverage experience in the buffer that has been generated by an arbitrary behavior policy. In addition to the replay buffer, a target network $Q_{\boldsymbol{\theta}'}$ is leveraged for calculating the target, which additionally increases the stability. The goal of $Q_{\boldsymbol{\theta}}$ is to minimize the loss

$$\mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}[(r + \gamma \max_{a'} Q_{\boldsymbol{\theta}'}(s',a') - Q_{\boldsymbol{\theta}}(s,a))^2]. \tag{3.49}$$

The target network is updated after a certain number of update steps. DQN constitutes one of the most popular value-based RL algorithms and has experienced multiple advancements such as double DQN (Hasselt, 2010), dueling DQN (Wang et al., 2016) or Rainbow (Hessel et al., 2018). The DQN will be used in Chapter 5 for maximizing information gain.

3.4.2 Policy-based Methods

In contrast to value-based methods that retrieve the policy through the Q-function, policy-based methods parameterize the policy $\pi_{\boldsymbol{\theta}}$ directly using policy parameters $\boldsymbol{\theta}$. Policy-based methods are able to learn a stochastic policy and do not rely on the arg max operation for action selection, which allows them to change their behavior more smoothly.

In episodic environments, such as dialogue, the goal of the policy is to maximize the expected return in every starting state s_0. More formally, the objective is to maximize

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right] = \sum_{s_0} p_0(s_0) V^{\pi_{\boldsymbol{\theta}}}(s_0). \tag{3.50}$$

Our goal is to take the gradient of $J(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ in order to optimize $\pi_{\boldsymbol{\theta}}$ using gradient descent. The policy gradient theorem (Sutton et al., 1999) gives us an analytical expression on how to achieve this. In order to state it, we define the occupancy measure as follows.

Definition 4. *For a policy π, the occupancy measure $d^\pi(s,a)$ is defined as*

$$d^\pi(s,a) = d^\pi(s) \cdot \pi(a|s), \quad \textit{where } d^\pi(s) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t p_\pi(S_t = s), \tag{3.51}$$

where $p_\pi(S_t = s)$ is the probability that s occurs in time step t when generating trajectories using π.

The occupancy measure thus expresses how likely it is to observe a specific state-action pair when producing trajectories using π. The likelihood for the occurance in time step t is discounted, where the term $(1-\gamma)$ is a normalization so that $d^\pi(s,a)$ defines a probability distribution. The policy gradient theorem is given as follows.

Theorem 1 (Policy Gradient Theorem)**.** *For a policy $\pi_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$, the gradient of $J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right]$ is given by*

$$\nabla J(\boldsymbol{\theta}) = \frac{1}{1-\gamma} \mathbb{E}_{S,A\sim d^{\pi_{\boldsymbol{\theta}}}}[A^{\pi_{\boldsymbol{\theta}}}(S,A) \nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(A|S)]. \tag{3.52}$$

We provide a proof of the policy gradient theorem in the Appendix in Theorem 6. The policy gradient has the intuitive interpretation that it will make actions more or less likely based on the advantage when taking the action. The theorem has many different versions, where the advantage function is substituted by other values that are based on the return (Schulman et al., 2016). One version substitutes the advantage function by the Q-function, which can be derived as follows.

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{1-\gamma}\mathbb{E}_{S,A\sim d^{\pi_{\boldsymbol{\theta}}}}[A^{\pi_{\boldsymbol{\theta}}}(S,A)\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(A|S)] \tag{3.53}$$

$$= \frac{1}{1-\gamma}\mathbb{E}_{S\sim d^{\pi_{\boldsymbol{\theta}}}}[\sum_a \pi_{\boldsymbol{\theta}}(a|S) \cdot A^{\pi_{\boldsymbol{\theta}}}(S,a)\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(a|S)] \tag{3.54}$$

$$= \frac{1}{1-\gamma}\mathbb{E}_{S\sim d^{\pi_{\boldsymbol{\theta}}}}[\sum_a \pi_{\boldsymbol{\theta}}(a|S) \cdot (Q^{\pi_{\boldsymbol{\theta}}}(S,a) - V^{\pi_{\boldsymbol{\theta}}}(S))\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(a|S)] \tag{3.55}$$

$$= \frac{1}{1-\gamma}\mathbb{E}_{S\sim d^{\pi_{\boldsymbol{\theta}}}}[\sum_a \pi_{\boldsymbol{\theta}}(a|S) \cdot Q^{\pi_{\boldsymbol{\theta}}}(S,a)\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(a|S)] \tag{3.56}$$

$$= \frac{1}{1-\gamma}\mathbb{E}_{S,A\sim d^{\pi_{\boldsymbol{\theta}}}}[Q^{\pi_{\boldsymbol{\theta}}}(S,A)\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(A|S)] \tag{3.57}$$

where Equation 3.56 followed since

$$\sum_a \pi_{\boldsymbol{\theta}}(a|S) \cdot V^{\pi_{\boldsymbol{\theta}}}(S)\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(a|S) = V^{\pi_{\boldsymbol{\theta}}}(S)\sum_a \nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(a|S) \tag{3.58}$$

$$= V^{\pi_{\boldsymbol{\theta}}}(S)\nabla_{\boldsymbol{\theta}}\sum_a \pi_{\boldsymbol{\theta}}(a|S) \tag{3.59}$$

$$= V^{\pi_{\boldsymbol{\theta}}}(S)\nabla_{\boldsymbol{\theta}} 1 \tag{3.60}$$

$$= 0. \tag{3.61}$$

Off-Policy Policy Gradient

Similar to the off-policy targets for the value-functions that we have discussed in Section 3.3.3, we can derive an off-policy version of the policy gradient through importance sampling:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{1-\gamma}\mathbb{E}_{S,A\sim d^{\mu}}\left[\frac{d^{\pi_{\boldsymbol{\theta}}}(S)}{d^{\mu}(S)}\frac{\pi_{\boldsymbol{\theta}}(A|S)}{\mu(A|S)}A^{\pi_{\boldsymbol{\theta}}}(S,A)\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(A|S)\right]. \tag{3.62}$$

Nevertheless, the term $\frac{d^{\pi_{\boldsymbol{\theta}}}(S)}{d^{\mu}(S)}$ is difficult to estimate in practice (Imani et al., 2018), which is why the term is typically omitted (Degris et al., 2012), leading to

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{1-\gamma}\mathbb{E}_{S,A\sim d^{\mu}}\left[\frac{\pi_{\boldsymbol{\theta}}(A|S)}{\mu(A|S)}A^{\pi_{\boldsymbol{\theta}}}(S,A)\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(A|S)\right]. \tag{3.63}$$

In the V-trace algorithm (Espeholt et al., 2018) that will be used in Chapter 6 and 7, the importance weights are additionally truncated by the hyperparameter $\bar{\rho}$, which leads to

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{1-\gamma}\mathbb{E}_{S,A\sim d^{\mu}}\left[\min(\bar{\rho}, \frac{\pi_{\boldsymbol{\theta}}(A|S)}{\mu(A|S)})A^{\pi_{\boldsymbol{\theta}}}(S,A)\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(A|S)\right]. \tag{3.64}$$

3.4.3 Actor-Critic Methods

In order to calculate the policy gradient, we require an estimate of $A^{\pi_{\boldsymbol{\theta}}}(s_t, a_t)$ or $Q^{\pi_{\boldsymbol{\theta}}}(s_t, a_t)$, which can be obtained by using a sampled return g_t directly. Nevertheless, the usage of the return has the disadvantage that it exhibits high variance. Instead of the return, we can use a function approximater, called a critic, that provides an estimate of $A^{\pi_{\boldsymbol{\theta}}}(s_t, a_t)$ or $Q^{\pi_{\boldsymbol{\theta}}}(s_t, a_t)$. The function approximator is called a critic since the policy actions are reinforced or suppressed in the policy gradient based on its judgment. If a critic is used for calculating the policy gradient, the policy is called an actor and the corresponding algorithm an actor-critic algorithm. The critic is required to approximate $A^{\pi_{\boldsymbol{\theta}}}(s_t, a_t)$ or $Q^{\pi_{\boldsymbol{\theta}}}(s_t, a_t)$, which has been covered in Section 3.3. In order to estimate the advantage function, we can train a critic $V_{\boldsymbol{\psi}}$ to approximate the state-value function $V^{\pi_{\boldsymbol{\theta}}}$ and obtain a sample of the advantage function through

$$r_t + \gamma V_{\boldsymbol{\psi}}(s_{t+1}) - V_{\boldsymbol{\psi}}(s_t) \tag{3.65}$$

due to Equation 3.11. Since actor-critic methods learn both a value-function and a parameterized policy, they can be thought of as a fusion of policy-based and value-based methods.

Algorithm 2 V-trace Actor-Critic

Require: neural networks for actor $\pi_{\boldsymbol{\theta}}$ and critic $V_{\boldsymbol{\psi}}$, learning rates α_a and α_c, truncation parameters $\bar{\rho}$ and $\bar{c}$, gradient weights $\lambda_a, \lambda_c, \lambda_e$, number of new trajectories n_{new}

$\mathcal{D} \leftarrow \{\}$
while true **do**
  Generate n_{new} trajectories $\tau = s_0, a_0, r_0, \ldots$ using $\pi_{\boldsymbol{\theta}}$
  Store trajectories $s_0, a_0, \pi_{\boldsymbol{\theta}}(a_0|s_0), r_0, \ldots$ in $\mathcal{D}$
  Sample a mini-batch $\mathcal{D}_{\text{batch}}$ of trajectories from $\mathcal{D}$
  $N, g_{\boldsymbol{\psi}}, g_{\boldsymbol{\theta}} \leftarrow 0$
  for trajectory $\tau = s_0, a_0, \mu(a_0|s_0), r_0, \ldots, s_T$ in $\mathcal{D}_{\text{batch}}$ **do**
    $v_T \leftarrow 0$
    for $t = T-1, \ldots, 0$ **do**
      Calculate V-trace target v_t according to Equation 3.30
      $g_{\boldsymbol{\psi}} \leftarrow g_{\boldsymbol{\psi}} + \lambda_c(v_t - V_{\boldsymbol{\psi}}(s_t)) \cdot \nabla_{\boldsymbol{\psi}} V_{\boldsymbol{\psi}}(s_t, a_t)$ ▷ towards V-trace target
      $g_{\boldsymbol{\theta}} \leftarrow g_{\boldsymbol{\theta}} + \lambda_a \rho_t [r_t + \gamma v_{t+1} - V_{\boldsymbol{\psi}}(s_t)] \nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(a_t|s_t)$ ▷ policy gradient
      $g_{\boldsymbol{\theta}} \leftarrow g_{\boldsymbol{\theta}} + \lambda_e \nabla_{\boldsymbol{\theta}} \mathcal{H}(\pi_{\boldsymbol{\theta}}(\cdot|s))$ ▷ entropy
    end for
    $N \leftarrow N + T$
  end for
  $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi} + \alpha_c \cdot \frac{1}{N} g_{\boldsymbol{\psi}}$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_a \cdot \frac{1}{N} g_{\boldsymbol{\theta}}$
end while

V-trace Actor-Critic

The V-trace actor-critic algorithm leverages an actor $\pi_{\boldsymbol{\theta}}$ and a critic $V_{\boldsymbol{\psi}}$ that approximates the state-value function $V^{\pi_{\boldsymbol{\theta}}}$. The state-value function is updated using the V-trace target v_t explained in Section 3.3.3:

$$\boldsymbol{\psi} \leftarrow \boldsymbol{\psi} + \alpha_c \cdot (v_t - V_{\boldsymbol{\psi}}(s_t)) \cdot \nabla_{\boldsymbol{\psi}} V_{\boldsymbol{\psi}}(s_t), \tag{3.66}$$

where α_c is a learning rate. Given a tuple (s_t, a_t, r_t, s_{t+1}), the policy is updated based on the off-policy policy gradient in Equation 3.64 and an estimate of the advantage function (Equation 3.65) as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_a \cdot \rho_t [r_t + \gamma v_{t+1} - V_{\psi}(s_t)] \nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(a_t|s_t), \tag{3.67}$$

where α_a is a learning rate, $\rho_t = \min(\overline{\rho}, \frac{\pi_{\boldsymbol{\theta}}(a_t|s_t)}{\mu(a_t|s_t)})$ and v_{t+1} is the V-trace target at time step $t+1$. In addition, in order to circumvent premature convergence and facilitate exploration, the policy is optimized towards high entropy. For a state s, the entropy $\mathcal{H}$ of $\pi_{\boldsymbol{\theta}}(\cdot|s)$ is defined as

$$\mathcal{H}(\pi_{\boldsymbol{\theta}}(\cdot|s)) = -\sum_a \pi_{\boldsymbol{\theta}}(a|s) \log(\pi_{\boldsymbol{\theta}}(a|s)). \tag{3.68}$$

The entropy attains its maximum if $\pi_{\boldsymbol{\theta}}$ is a uniform distribution and is minimal if there is an action a such that $\pi_{\boldsymbol{\theta}}(a|s) = 1$. The higher the entropy, the more actions are likely to be sampled by $\pi_{\boldsymbol{\theta}}(\cdot|s)$. The entropy loss function thus guarantees that the policy persists in selecting multiple actions rather than focusing exclusively on a single, potentially suboptimal action.

As in deep Q-learning, V-trace leverages a replay buffer $\mathcal{D}$ for increased sample efficiency. Since the algorithm is off-policy, it is possible to use experience that has been generated by another behavior policy that is for instance a previous version of $\pi_{\boldsymbol{\theta}}$. During generation, when the policy $\pi_{\boldsymbol{\theta}}$ takes action a_t in state s_t, the value $\mu(a_t|s_t)$ is defined as $\pi_{\boldsymbol{\theta}}(a_t|s_t)$ and additionally saved in the replay buffer. The neural network weights are updated after generating n_{new} new trajectories. We provide the V-trace algorithm with mini-batch sampling from the replay buffer in Algorithm 2.

3.5 Continual Reinforcement Learning

In the previous section we were concerned with how a RL agent could optimize its behavior in a fixed, given MDP $\mathcal{M}$. While this leads to optimal behavior in a narrow, static task of interest, it is only the first step towards truly intelligent behavior. As the world is ever-changing, RL agents require the ability to continue learning over their lifetime, constantly adapting to changes in the environment. They need to learn new and more complex tasks over time, while building on previously acquired knowledge, just like humans. The study of how RL agents can learn in ever-changing environments is known as continual reinforcement learning. The general continual RL problem can be formulated through a MDP $\mathcal{M}_{\text{CRL}}$ that changes throughout time (Khetarpal et al., 2022):

$$\mathcal{M}_{\text{CRL}}(t) = < \mathcal{S}(t), \mathcal{A}(t), r(t), p(t), p_0(t), \gamma > . \tag{3.69}$$

A common assumption is local stationarity of the time-dependent MDP $\mathcal{M}_{\text{CRL}}$ (Khetarpal et al., 2022; Padakandla et al., 2019). This is the case if there is a sequence of time-steps $t_0, t_1, ..., t_k, ...$ such that for all k, we have

$$\mathcal{M}_{\text{CRL}}(t) = \mathcal{M}_{\text{CRL}}(t_k), \quad t_k \leq t < t_{k+1} \tag{3.70}$$

Locally stationary MDPs $\mathcal{M}_{\text{CRL}}$ give rise to a sequence of MDPs

$$\mathcal{M}_0 \rightarrow \mathcal{M}_1 \rightarrow \mathcal{M}_2 \rightarrow ... \rightarrow \mathcal{M}_k \rightarrow ...$$

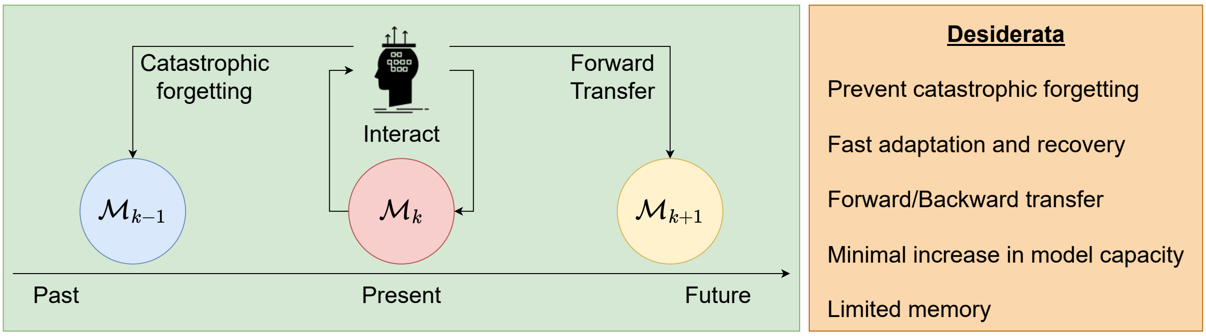


FIGURE 3.2: In continual RL, an agent is exposed to a sequence of MDPs and requires various capabilities, summarized in the desiderata.

where $\mathcal{M}_k = \mathcal{M}_{\text{CRL}}(t_k)$. The MDPs $\mathcal{M}_k$ are often referred to as tasks. Many works in continual RL formulate the general continual RL problem by stating that the RL agent is exposed to a sequence of MDPs, which coincides with the formulation as a time-dependent MDP. We call the two different views the sequence view and time-dependent view of continual RL. The time-dependent view is more intuitive when the $\mathcal{M}_{\text{CRL}}$ changes smoothly throughout time whereas the sequence view is more intuitive if $\mathcal{M}_{\text{CRL}}$ changes at specific time-points (i.e. when it is locally stationary). Examples for smooth changes are given by changing friction, preferences, or wind conditions (Chandak et al., 2020; Liotet et al., 2022; Xie et al., 2021). Examples where the sequence view is more intuitive is when the agent plays different games one after the other or conducts conversations about different domains (Madotto et al., 2021; Powers et al., 2022; Rolnick et al., 2019; Wołczyk et al., 2021). During continual learning of dialogue policies, the MDP is unlikely to change after every conducted dialogue. Consequently, we will henceforth focus on local stationarity and use the sequence view of continual RL.

3.5.1 Objectives and Challenges in Continual RL

Imagine that the RL agent has interacted with the MDP $\mathcal{M}_{k-1}$ and is now exposed to $\mathcal{M}_k$. The agent should adapt as fast as possible to the new environment given by $\mathcal{M}_k$. At the same time, the learner should not forget how to act in $\mathcal{M}_{k-1}$. The RL agent thus requires stability in order to not forget how to act in previous MDPs while exhibiting plasticity for adapting to the current MDP, which is known as stability-plasticity dilemma in neuroscience. The failure mode of forgetting how to solve previous MDPs is known as catastrophic forgetting (McCloskey and Cohen, 1989). Catastrophic forgetting in neural networks occurs when the gradient updates for learning on the current MDP move the neural network parameters away from solutions that work well on all MDPs. In addition to fast adaptation and preventing forgetting, the learning agent should be able to transfer what it has learned from previous experiences to new situations and make use of more recent experiences to improve performance on capabilities learned earlier. In other words, the learner should be capable of forward and backward transfer learning. All of this should be done with only limited memory and neural network capacity (Hadsell et al., 2020). In summary, the continual learner should fulfill the following points (see also Figure 3.2)

- **Prevent catastrophic forgetting:** Adapting to a new MDP should not significantly reduce performance on previously observed MDPs.

- **Fast adaptation and recovery:** The learner should be capable of fast adaptation to novel MDPs and fast recovery when presented with past MDPs.
- **Forward/Backward Transfer:** The learner should be capable of transferring previously learned knowledge to new situations and make use of recent experience to improve capabilities learned earlier.
- **Minimal increase in model capacity:** The learning method must be scalable and consequently work with only minimal increase in model capacity.
- **Limited memory:** The learner has only limited memory in practice.

Due to its multitude of challenges, continual learning has intersections with many other disciplines such as multi-task learning (Teh et al., 2017), meta-learning (Beck et al., 2023), transfer-learning (Taylor and Stone, 2009) and representation learning (Javed and White, 2019).

Evaluation

The selection of metrics for evaluating continually learning agents typically depend on the problem setup. Conventional metrics that are always applicable include the overall reward or return during a lifetime and the average reward or return per step in a time-window (Platanios et al., 2020; Rolnick et al., 2019; Schaul et al., 2018). If the continual learning problem is given by learning from a sequence of distinctive MDPs, it is common to have evaluation phases during learning (in particular at the time of the MDP change), where the learner is evaluated on all MDPs that will be seen during the lifetime. In particular, the learner is evaluated on the performance for unseen MDPs for measuring forward transfer as well as on how much has been forgotten from previously seen MDPs (Powers et al., 2022; Wołczyk et al., 2021). We emphasize that this evaluation requires access to all MDPs that will occur in the lifetime as well as dedicated evaluation phases that are neither always available in real-world applications nor adequately reflect the future performance of the agent (Khetarpal et al., 2022).

3.5.2 Continual Learning Approaches

Continual learning methods can be broadly categorized into the following approaches, where many of them focus on the problem of catastrophic forgetting.

Regularization-based methods (Aljundi et al., 2018; Kirkpatrick et al., 2017; Zenke et al., 2017) are single-model approaches that penalize the movement of neural network parameters that are important for solving previous MDPs to circumvent catastrophic forgetting.

Architectural methods dynamically expand the neural network with new MDPs. A naive approach to prevent catastrophic forgetting is by training an individual model for each arising MDP. Nevertheless, this has multiple disadvantages such as an increased neural network parameter size for every new MDP as well as no possibility for forward or backward transfer. The idea of neural network expansion has led to various more advanced proposals such as Progressive Networks (Rusu et al., 2016), Dynamically Expandable Networks (Yoon et al., 2018), Reinforced Continual Learning (Xu and Zhu, 2018) or Supermasks in Superposition (Wortsman et al., 2020).

Another family of approaches are built on the idea of knowledge distillation from a teacher into a student model (Hinton et al., 2015). Distillation for continual learning can be used to retain previous knowledge by providing a new auxiliary target for the network being trained (Kaplanis et al., 2019; Li and Hoiem, 2018; Schwarz et al., 2018; Traoré et al., 2019). In contrast to methods that expand the neural network with new MDPs, the neural network size for knowledge distillation methods can remain constant.

Instead of storing knowledge inside of neural network parameters, rehearsal-based methods store previous experience or training examples in a replay buffer (Lin, 1992), so that the model can continue optimizing on previous MDPs (Aljundi et al., 2019; Chaudhry et al., 2019; Lopez-Paz and Ranzato, 2017; Riemer et al., 2019; Rolnick et al., 2019). In order to tackle limited storage capacity, different selection strategies (Aljundi et al., 2019; Isele and Cosgun, 2018) or pseudo-rehearsal through generative replay (Atkinson et al., 2021; Daniels et al., 2022; Shin et al., 2017; Ven and Tolias, 2018) have been proposed.

Continual Learning with Experience and Replay (CLEAR)

In this section, we discuss the rehearsal-based continual RL method CLEAR (Rolnick et al., 2019) that will be used in Chapter 6 and 7. CLEAR addresses the two main challenges of catastrophic forgetting and fast adaptation and has been shown to achieve state-of-the-art performance despite being significantly less complicated compared to previous approaches. CLEAR is an actor-critic algorithm (see Section 3.4.3) that aims for fast adaptation to new situations and prevention of catastrophic forgetting by training the policy and critic on a mixture of new and replayed experience. It leverages the off-policy algorithm V-trace that was discussed in Section 3.4.3 and proposes the following additions.

Fast adaptation: In order to adapt fast to new situations, CLEAR not only learns from sampled experience of the replay buffer $\mathcal{D}$ that likely contains many old trajectories, but also leverages new experience that has been just generated. More specifically, in every update step, CLEAR retrieves a dataset $\mathcal{D}_{\text{rec}}$ consisting of the n_{rec} most recent trajectories from $\mathcal{D}$ and samples data $\mathcal{D}_{\text{old}}$ of size n_{old} from the remaining experience in $\mathcal{D} \setminus \mathcal{D}_{\text{rec}}$. Analogous to V-trace in Section 3.4.3 and Algorithm 2, we then use the experience in $\mathcal{D}_{\text{both}} = \mathcal{D}_{\text{rec}} \cup \mathcal{D}_{\text{old}}$ and calculate update gradients

$$\hat{g}_{\psi} = \lambda_c \cdot \mathbb{E}_{\mathcal{D}_{\text{both}}}[(v_t - V_{\psi}(s_t)) \cdot \nabla_{\psi} V_{\psi}(s_t)], \tag{3.71}$$

$$\hat{g}_{\boldsymbol{\theta}} = \mathbb{E}_{\mathcal{D}_{\text{both}}}[\lambda_a \cdot \rho_t[r_t + \gamma v_{t+1} - V_{\psi}(s_t)]\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(a_t|s_t) + \lambda_e \nabla_{\boldsymbol{\theta}} \mathcal{H}(\pi_{\boldsymbol{\theta}}(\cdot|s_t))], \tag{3.72}$$

where $\lambda_a, \lambda_c, \lambda_e \in \mathbb{R}$ are hyperparameters. The usage of experience in $\mathcal{D}_{\text{rec}}$ is supposed to facilitate fast adaption.

Prevent catastrophic forgetting: In order to prevent catastrophic forgetting of already acquired knowledge, CLEAR proposes a simple regularization as follows. For a state s and action a in the replay buffer, let $V_{\text{old}}(s)$ denote the prediction of V_{ψ} at the moment of collecting state s and μ the policy that took action a in state s (i.e. an old version of $\pi_{\boldsymbol{\theta}}$). We can mitigate forgetting of our previous knowledge if the critic prediction $V_{\psi}(s)$ is close to $V_{\text{old}}(s)$ and the policy distribution $\pi_{\boldsymbol{\theta}}(\cdot|s)$ is close to $\mu(\cdot|s)$. This can be achieved by minimizing the mean-squared error $(V_{\text{old}}(s) - V_{\psi}(s))^2$ and the KL-divergence

$$\text{KL}[\mu(\cdot|s)||\pi_{\boldsymbol{\theta}}(\cdot|s)] = \sum_a \mu(a|s) \ln \frac{\mu(a|s)}{\pi_{\boldsymbol{\theta}}(a|s)}. \tag{3.73}$$

The KL-divergence is zero if and only if $\mu(\cdot|s) = \pi_{\boldsymbol{\theta}}(\cdot|s)$ (Cover and Thomas, 2006) and thus defines an adequate regularization for $\pi_{\boldsymbol{\theta}}$. Since the objective is to prevent forgetting of previous knowledge, we apply the regularizations only to data in $\mathcal{D}_{\text{old}}$. This results in the following update gradients

$$g_{\psi} = \hat{g}_{\psi} + \lambda_{rc} \cdot \mathbb{E}_{\mathcal{D}_{\text{old}}}[(V_{\text{old}}(s_t) - V_{\psi}(s_t)) \cdot \nabla_{\psi} V_{\psi}(s_t)] \tag{3.74}$$

$$g_{\boldsymbol{\theta}} = \hat{g}_{\boldsymbol{\theta}} - \lambda_{ra} \cdot \mathbb{E}_{\mathcal{D}_{\text{old}}}[\nabla_{\boldsymbol{\theta}} \text{KL}[\mu(\cdot|s_t)||\pi_{\boldsymbol{\theta}}(\cdot|s_t)]], \tag{3.75}$$

where $\lambda_{rc}, \lambda_{ra} \in \mathbb{R}$ are hyperparameters. The terms $\lambda_{rc}, \lambda_{ra}$ determine the strength of regularization towards previously acquired knowledge and thus strike a balance between fast adaptation and the prevention of forgetting. Finally, we use the gradients for updating our neural network weights as

$$\psi \leftarrow \psi + \alpha_c \cdot g_{\psi}, \tag{3.76}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_a \cdot g_{\boldsymbol{\theta}}. \tag{3.77}$$

We perform an update after we have generated n_{new} many trajectories. A summary is given in Algorithm 3, where the main differences to the V-trace algorithm (Algorithm 2) are in line 5, 12 and 13.

Algorithm 3 Continual Learning with Experience and Replay (CLEAR)

Require: neural networks for actor $\pi_{\boldsymbol{\theta}}$ and critic V_{ψ}, learning rates α_a and α_c, truncation parameters $\overline{\rho}$ and $\overline{c}$ for V-trace, gradient weights $\lambda_a, \lambda_c, \lambda_e, \lambda_{rc}, \lambda_{ra}$, number of trajectories $n_{\text{rec}}, n_{\text{old}}, n_{\text{new}}$

1: $\mathcal{D} \leftarrow \{\}$
2: **while** true **do**
3: Generate n_{new} trajectories $\tau = s_0, a_0, r_0, ...$ using $\pi_{\boldsymbol{\theta}}$
4: Store trajectories $s_0, V_{\psi}(s_t), a_0, \pi_{\boldsymbol{\theta}}(\cdot|s_0), r_0, ...$ in $\mathcal{D}$
5: Retrieve $\mathcal{D}_{\text{rec}}$ of n_{rec} most recent trajectories from $\mathcal{D}$ ▷ for fast adaptation
6: Sample a mini-batch $\mathcal{D}_{\text{old}}$ of size n_{old} from $\mathcal{D} \setminus \mathcal{D}_{\text{rec}}$
7: $\mathcal{D}_{\text{both}} \leftarrow \mathcal{D}_{\text{rec}} \cup \mathcal{D}_{\text{old}}$
8: $g_{\psi}, g_{\boldsymbol{\theta}} \leftarrow 0$
9: $g_{\psi} \leftarrow g_{\psi} + \lambda_c \cdot \mathbb{E}_{\mathcal{D}_{\text{both}}}[(v_t - V_{\psi}(s_t)) \cdot \nabla_{\psi} V_{\psi}(s_t)]$ ▷ towards V-trace target
10: $g_{\boldsymbol{\theta}} \leftarrow g_{\boldsymbol{\theta}} + \lambda_a \cdot \mathbb{E}_{\mathcal{D}_{\text{both}}}[\rho_t [r_t + \gamma v_{t+1} - V_{\psi}(s_t)] \nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(a_t|s_t)]$ ▷ policy gradient
11: $g_{\boldsymbol{\theta}} \leftarrow g_{\boldsymbol{\theta}} + \lambda_e \cdot \mathbb{E}_{\mathcal{D}_{\text{both}}}[\nabla_{\boldsymbol{\theta}} \mathcal{H}(\pi_{\boldsymbol{\theta}}(\cdot|s_t))]$ ▷ entropy
12: $g_{\psi} \leftarrow g_{\psi} + \lambda_{rc} \cdot \mathbb{E}_{\mathcal{D}_{\text{old}}}[(V_{\text{old}}(s_t) - V_{\psi}(s_t)) \cdot \nabla_{\psi} V_{\psi}(s_t)]$ ▷ mitigate forgetting of critic
13: $g_{\boldsymbol{\theta}} \leftarrow g_{\boldsymbol{\theta}} - \lambda_{ra} \cdot \mathbb{E}_{\mathcal{D}_{\text{old}}}[\nabla_{\boldsymbol{\theta}} \text{KL}[\mu(\cdot|s_t)||\pi_{\boldsymbol{\theta}}(\cdot|s_t)]]$ ▷ mitigate forgetting of actor
14: $\psi \leftarrow \psi + \alpha_c \cdot g_{\psi}$
15: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_a \cdot g_{\boldsymbol{\theta}}$
16: **end while**

3.6 Partially Observable Markov Decision Process (POMDP)

This chapter centered around a Markov decision process with state space $\mathcal{S}$, action space $\mathcal{A}$, reward function r, transition probability p and starting probability p_0. In every step, the RL agent obtains the environment state for action selection (Figure 3.1). Consequently, there is the assumption that the environment state is fully observable, which is often times not the case, for instance in dialogue as we will see in the next chapter.

A generalization of a MDP that does not rely on the assumption of fully observability is a partially observable Markov decision process (POMDP) (Kaelbling et al., 1998). In this setup, the RL agent only obtains an observation that is omitted based on the actual environment state. The POMDP is formally given by $\mathcal{M} =< \mathcal{S}, \mathcal{A}, \mathcal{O}, r, p, p_0, p_{\text{obs}}, \gamma >$, where the additional terms are

- $\mathcal{O}$: set of observations
- p_{obs} : observation probability function, where $p_{\text{obs}}(o|s)$ denotes the probability of observation o given state s

Since the agent in a POMDP can not observe the true state but only the observations, the agent has to act under uncertainty of the environment state. Based on the perceived observations, the agent can attempt to produce a *belief state* b, which is defined by a probability distribution over states that encapsulates the agent's belief of the true state. For a given belief state b_t, the agent takes an action a_t. The environment then transitions to a new state s_{t+1} according to $p(\cdot|s_t, a_t)$ and emits an observations o_{t+1} according to $p_{\text{obs}}(\cdot|s_{t+1})$. The belief state b_t can then be updated to b_{t+1} as follows. Utilizing Bayes theorem (Harney, 2003), we obtain for state $s' \in \mathcal{S}$

$$\begin{aligned} b_{t+1}(s') &= p(s'|a_t, o_{t+1}, b_t) && (3.78) \\ &= \frac{p(o_{t+1}|s', a_t, b_t) \cdot p(s'|a_t, b_t)}{p(o_{t+1}|b_t, a_t)} && (3.79) \\ &= \frac{1}{p(o_{t+1}|b_t, a_t)} p_{\text{obs}}(o_{t+1}|s') \cdot \sum_s p(s'|s, a_t, b_t) p(s|a_t, b_t) && (3.80) \\ &= \frac{1}{p(o_{t+1}|b_t, a_t)} p_{\text{obs}}(o_{t+1}|s') \cdot \sum_s p(s'|s, a_t) \cdot b_t(s), && (3.81) \end{aligned}$$

where the term $p(o_{t+1}|b_t, a_t)$ can be calculated as

$$\begin{aligned} p(o_{t+1}|b_t, a_t) &= \sum_{s'} p(o_{t+1}|s', b_t, a_t) \cdot p(s'|b_t, a_t) && (3.82) \\ &= \sum_{s'} p_{\text{obs}}(o_{t+1}|s') \cdot \sum_s p(s'|s, b_t, a_t) \cdot p(s|b_t, a_t) && (3.83) \\ &= \sum_{s'} p_{\text{obs}}(o_{t+1}|s') \cdot \sum_s p(s'|s, a_t) \cdot b_t(s). && (3.84) \end{aligned}$$

For the emitted observation o of a starting state, the belief state b_0 can be inferred as

$$\begin{aligned} b_0(s) = p(s|o) &= \frac{p_{\text{obs}}(o|s) \cdot p(o)}{p_0(s)}, \text{where} && (3.85) \\ p(o) &= \sum_s p_{\text{obs}}(o|s) \cdot p_0(s). && (3.86) \end{aligned}$$

Note that these calculations require the summation over all state, which is mostly intractable in practice.

3.6.1 Belief MDP

The POMDP can be recast into a MDP, a so-called belief MDP, such that our methods discussed in this chapter still apply. The belief MDP is given by $\mathcal{M} =< \mathcal{B}, \mathcal{A}, r_b, p_b, p_{b_0}, \gamma >$, where

- $\mathcal{B}$ is the set of belief states over $\mathcal{S}$
- $\mathcal{A}$ is the action space as in the POMDP
- $r_b(b,a) = \sum_s b(s) \cdot r(s,a)$ is the reward function
- $p_b(b'|b,a)$ defines the transition probabilities, where

$$p_b(b'|b,a) = \sum_o p(b'|b,a,o) \cdot p(o|b,a) = \sum_o \mathbb{1}_{b'=b_{t+1}} \cdot p(o|b,a), \tag{3.87}$$

  $\mathbb{1}$ denotes the indicator function and b_{t+1} was derived in Equation 3.78.

- p_{b_0} defines the starting belief state probability, where

$$p_{b_0}(b) = \sum_o p(b|o) \cdot p(o) = \sum_o \mathbb{1}_{b=b_0} \cdot p(o), \tag{3.88}$$

  and b_0 was derived in Equation 3.85.

The belief state will be leveraged in Chapter 5 for calculating information gain. Note that even though the state space itself can be finite, the belief state is infinitely large since there are infinitely many distributions over $\mathcal{S}$. This can result in seeing the exact same belief state only once and thus additionally requires the need for generalization across states, i.e. using function approximators.

3.7 Conclusion

This chapter introduced reinforcement learning, including its objectives and algorithms for optimizing RL agents. The environment of the RL agent is defined by a MDP, which provides states and rewards to the agent. The objective of the agent is to take actions in order to maximize its expected sum of rewards. RL algorithms for optimization were distinguished by value-based, policy-based and actor-critic approaches, where we discussed off-policy methods such as DQN and V-trace. Subsequently, we looked into continual RL, where an agent is exposed to a dynamically changing MDP $\mathcal{M}_{\text{CRL}}$. We discussed the continual RL algorithm CLEAR that adapts the V-trace algorithm in order to improve fast adaptation to new situations and prevention of catastrophic forgetting. We concluded the chapter with POMDPs that will be used in the next chapter for making RL amenable for dialogue system optimization.

Chapter 4

Dialogue Policy

The dialogue policy is the decision-making component of the task-oriented dialogue system and selects in every turn the appropriate response in order to steer the conversation towards task success. This thesis centers around dialogue policy optimization using reinforcement learning.

4.1 Dialogue as Belief MDP

In task-oriented dialog, we always assume an underlying user goal that drives the user's desire to conduct a conversation. Given the goal and the user's specific behavior, the user produces an utterance that is passed to the dialogue system. In turn, the system produces a response based on the first user utterance and the process repeats until the goal has been fulfilled or the user ends the conversation. The system response is selected by the dialogue policy, which is the decision-making component of the dialogue system. The difficulty is that the dialogue system does not know the user goal in advance. Instead, it needs to be inferred during the conversation based on the user utterances.

In that sense, dialogue is effectively defined through a partially observable Markov decision process (POMDP), see Section 3.6. The environment the dialogue policy interacts with is determined by the user, who will define the transition probabilities through its behavior and user goals. The environment state that incorporates the user goal is unobservable and needs to be inferred based on the observations, which are given by the user utterances. As a consequence, the dialogue system is required to produce its own belief state, given the observations (Williams and Young, 2007). In order to apply the RL methods that we have developed in Chapter 3, we view the POMDP as a belief MDP where the state space is substituted by the belief state space (see Section 3.6.1). The dialogue/belief state tracker is responsible for producing the belief state. In this work, we always assume the belief state is given and are not concerned with the challenge of producing it.

4.2 Ontology

The scope of what the dialogue system can understand and talk about is defined by an underlying ontology. As depicted in Figure 4.1, the ontology is comprised of domains, e.g. restaurants or hotels, domain-specific slots, e.g. the area or price, and values that a slot can take, e.g west, east, north, south, and center for the area slot. These concepts together define the scope of information that can be understood by the system, such as the restaurant area or the number of hotel stars. In particular, it defines the user goals that can be understood. A user goal is defined by defining a value for every domain-slot pair in the ontology, where a dedicated value none indicates no value. In addition, some slots are only requestable by the user and only have the possible values none and ?. For instance, if a user needs the address of a hotel that is located in the north with 4 stars and wifi, this can be defined as

$$\mathcal{G} = \{\text{Hotel: [stars=4, area=north, wifi=yes, address=?, parking=none, ...]}, \\ \text{Restaurant: [price=none, area=none, ...], ...}\}.$$

If there is only a single domain, which has at least one value not being none, the arising dialogue is called single domain dialogue, and multi-domain dialogue else. Multi-domain dialogues are more difficult since there are more domains included in the conversation, which makes the fulfillment of the complete dialogue more challenging.

In addition to the user goal definition, the ontology defines for every domain possible semantic user actions that can be retrieved from a user utterance. If the user informs about the hotel area being in the north, this can be semantically represented as

Hotel-inform-area-north.

Lastly, the ontology defines the actions of the system for each domain, such as booking a hotel room or informing on the phone number or name of the requested restaurant, semantically represented for instance as

Hotel-inform-address-St. Andrew street 18.

In this thesis, the policy works with semantic representations for both states and actions that are defined by the ontology. Semantic representations have the advantage that they are both interpretable and controllable and are the common choice in modular task-oriented dialogue systems.

4.3 States

4.3.1 State Information

The decision of a dialogue policy is commonly based on the hidden information state of the dialogue system (Williams and Young, 2007; Young et al., 2007). This hidden information state consists of the following information:

- User goal state: it summarizes the current goal of the user and is central to the fulfillment of the task. Based on the user goal state, the policy can decide to inform on values or request more information first.

- User action: the user action is important as it includes information related to requests made by the user. This allows reactive behavior by the policy, e.g. informing on a requested slot.

- Dialogue history: the dialogue history is important for ensuring the Markov property and can consist of past user and system actions.

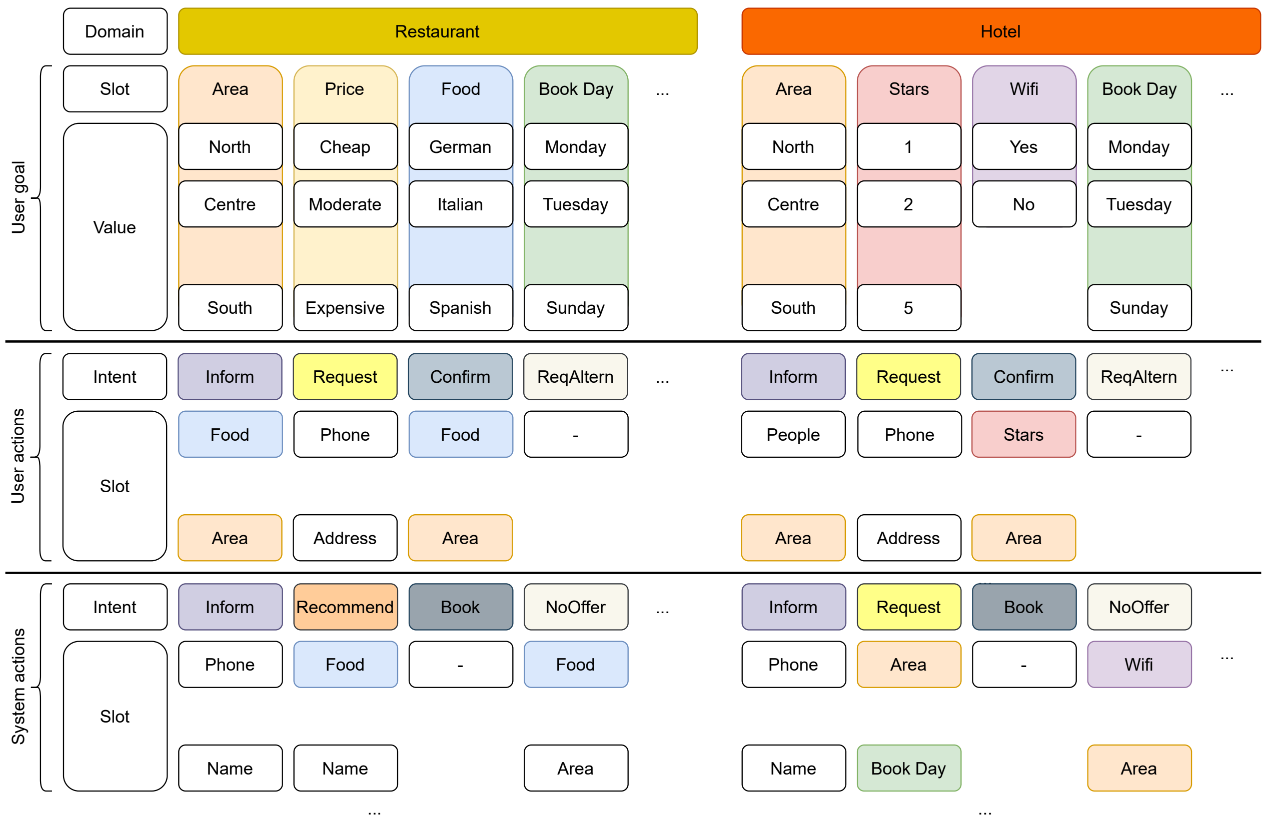


FIGURE 4.1: The ontology defines the scope of the dialogue system. The ontology is comprised of domains, slots within the domain, and values that a slot can take. A specific user goal is then constructed by specifying the values for each slot in each domain. In addition, the ontology defines user and system actions for every domain comprised of an intent and slot (and associated value). Once another domain will be added to the capabilities of the dialogue system, the ontology grows, leading to a growth in the state and action space of the dialogue system.

In addition, the state often contains information about the database results, for instance the number of entities that are available given the current predicted user goal (Weisz et al., 2018; Zhang et al., 2020; Zhu et al., 2022b). Moreover, since the user goal state already provides a form of dialogue summary, it is common to truncate the dialogue history information to only consider the last system action in order to have a fixed length history (Xu et al., 2020; Zhu et al., 2020). Figure 4.3 depicts an example state with the information that is typically included. While this information is essential for acting optimally, it is not necessarily sufficient. Humans for instance take into account emotions of the interlocutor that can additionally support decision-making.

User Goal Belief State

So far, the above state information does not include any probability distribution that measures our uncertainty about the environment state. Belief state trackers predict the user goal state based on the observations and infuse uncertainty in the following way (Mrkšić et al., 2017; Ramadan et al., 2018; van Niekerk et al., 2021). For every domain-slot pair d-s in the ontology, the belief tracker produces a probability $p_{\text{d}-\text{s}}$ over possible values. For instance, the probability distribution for the domain-slot pair restaurant-price could be given as

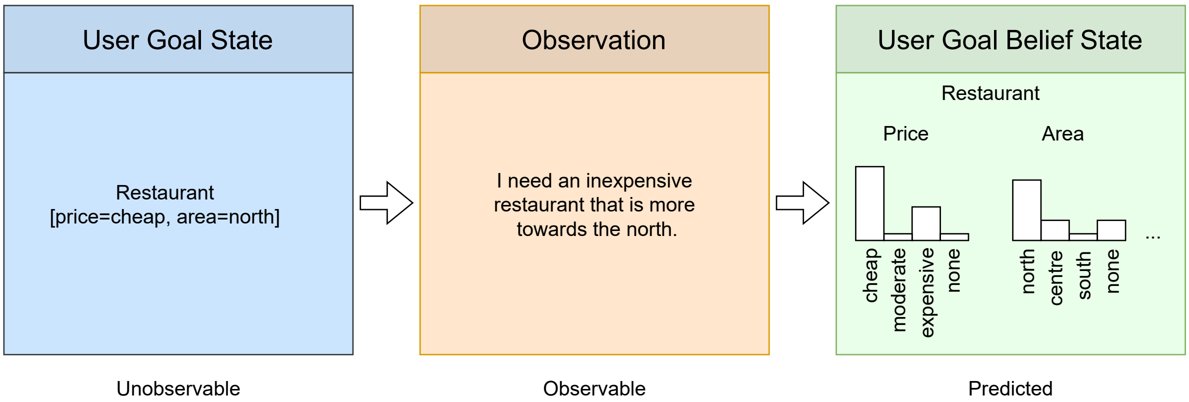


FIGURE 4.2: The difference between the unobservable user goal state and the predicted user goal belief state. The belief state is constructed based on the observations and is composed of a probability distribution for every domain-slot pair in the ontology. The distribution information helps the policy to resolve uncertainty.

$$p_{\text{rest}-\text{price}} = [\text{cheap} : 0.5, \text{moderate} : 0.1, \text{expensive} : 0.3, \text{none} : 0.1],$$

indicating uncertainty about the value being `expensive` or `cheap`. The distribution for every domain-slot pair informs the policy about possible uncertainties, which can be subsequently reduced by the policy through, for instance, confirm actions. Nevertheless, for action selection the policy typically does not leverage the full distribution in its state representation since that representation would be very large. Instead, the policy uses only the n highest probability values in $p_{\text{d}-\text{s}}$ (excluding the probability for the slot-value `none`), where n is a hyperparameter. The uncertainty information in $p_{\text{d}-\text{s}}$ will be also leveraged in Chapter 5 for the definition of information gain.

If we assume that the predictions for the different domain-slot pairs are conditionally independent, we can produce a probability distribution over possible user goal states, i.e. a user goal belief state, as follows. For a user goal $\mathcal{G} = \{d_i : s_i = v_i\}_i$, which encodes that domain-slot pair (d_i, s_i) takes value v_i, the probability of $\mathcal{G}$ is given by

$$p(\mathcal{G}) = \prod_i p_{d_i - s_i}(v_i). \tag{4.1}$$

The difference between user goal state and user goal belief state is depicted in Figure 4.2.

4.3.2 State Representation

As neural networks are leveraged for function approximation, it is necessary to map the semantic state into a vectorized representation, i.e. we require a vectorization strategy. The appropriate choice of state representation is key to the success of any form of RL (Madureira and Schlangen, 2020) and thus constitutes the first modeling challenge. This is particularly important in task-oriented dialogue since information for different domains often encompass the same or similar meaning, e.g. the slots `Price` and `Book Day` both appear in the restaurant and hotel domain, which can be exploited for transfer or continual learning.

The first group of state representation methods produce a single vector v by directly vectorizing information in the state (Takanobu et al., 2019; Weisz et al., 2018; Wesselmann et al., 2019; Zhu et al., 2020). For every information in the state, a single value is assigned

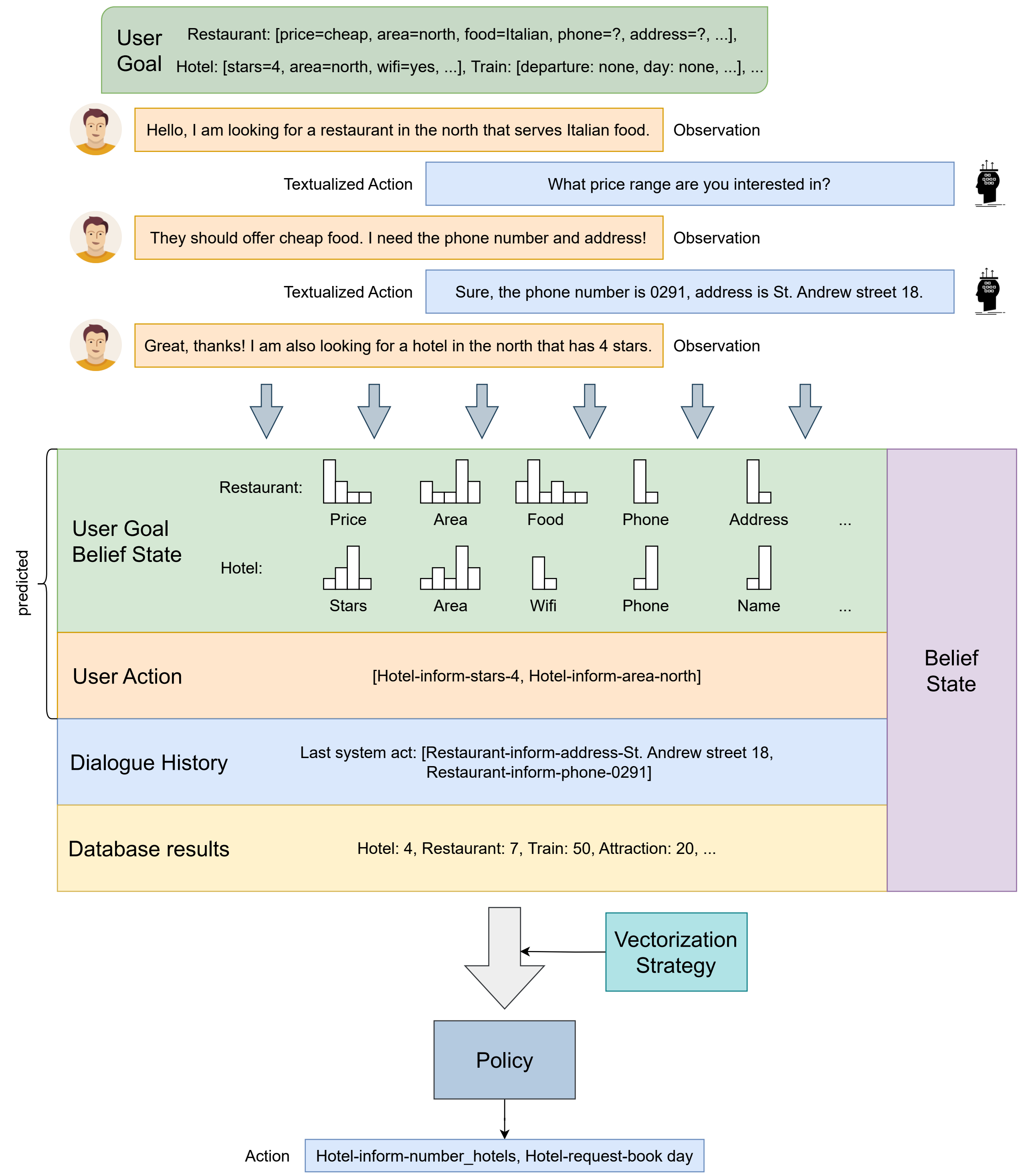


FIGURE 4.3: Example of state information that is typically used for decision making. The state is comprised of 1) the user goal belief state, 2) the user act retrieved from the user utterance, 3) the dialogue history, and 4) the results of the database query. In order to leverage neural networks for action selection we need a numerical representation, i.e. a vectorization strategy, which defines the first modelling challenge in dialogue policy learning. Given the state, the policy produces a semantic action which is comprised of a list of atomic actions.

which encodes the likelihood of absence or presence. For instance, for every domain-slot pair d-s in the ontology, the assigned value in v is given by the highest probability of $p_{\text{d}-\text{s}}$

(excluding the value `none`) in the belief state. This approach results in a state representation growth if new information needs to be added and ignores the semantic meaning of the information. It ignores similarities between information such as hotel and restaurant area and consequently allows only little transfer to other domains.

Another group of methods produces a vector by utilizing hand-coded domain-independent features for the information of each domain (Casanueva et al., 2018; Chen et al., 2018; Chen et al., 2020; Lin et al., 2021; Wang et al., 2015). Since the features are domain-independent, they allow transfer to another domain. However, the development of such features requires manual work of experts. Moreover, they assume that the domains share their semantic state structure, which would otherwise hinder domain-independent features.

In order to leverage semantic similarities and facilitate domain transfer without manual feature engineering, Xu et al. (2020) propose to use learnable vectors for domains, intents, slots, and values. These vectors are leveraged to produce a fixed size state representation by averaging over all domains. However, as the number of domains the dialogue policy can talk about becomes larger, averaging will likely result in information loss. Moreover, their architecture still largely depends on predefined feature categories (such as user act or user goal state). It thus does not model how novel information such as emotion can be included.

4.4 Actions

An atomic action is defined as a triplet `domain` − `intent` − `slot`. Examples for atomic actions are `Hotel` − `request` − `area` or `Restaurant` − `inform` − `address`, see also Figure 4.1. An associated value for an atomic action (e.g. the specific restaurant address) is subsequently added in a lexicalization process based on database results. Many works define the action space as the set of atomic actions (Weisz et al., 2018; Williams and Young, 2007). Nevertheless, the emergence of more complex dialogues such as multi-domain dialogues (Budzianowski et al., 2018) in dialogue policy learning requires the policy to take multiple atomic actions within a turn (e.g. for informing the address and phone number of a restaurant). We thus define a dialogue policy action as a list of atomic actions (Jhunjhunwala et al., 2020; Shu et al., 2019; Zhang et al., 2020). We note that the permission of a list of atomic actions enlarges the action space considerably. Moreover, if the policy is permitted to select an arbitrary large amount of atomic actions, this can lead to an information overflow for the user.

4.4.1 Action Generation

The requirement of producing a list of atomic actions motivates the usage of RNNs in dialogue policy research (Shu et al., 2019; Zhang et al., 2020), where each atomic action is generated by predicting domain, intent and slot sequentially. Nevertheless, these works do not model how new actions can be integrated.

Works that take into account new actions to be added to the action set compare the encoded state and action embeddings with each other (Lee, 2017; Vlasov et al., 2019; Xu et al., 2020), suggesting that exploiting similarities is key not only for state representations but also for action prediction.

4.5 Rewards

The reward function encodes the two major objectives of a task-oriented dialogue system: dialogue success and dialogue efficiency. Dialogue efficiency is encouraged by a small negative reward in every turn of the conversation. In addition, once the dialogue finished, the environment emits a large positive or negative reward for dialogue success or failure, respectively. The sparse reward decelerates the learning speed as the dialogue agent requires many environment interactions to figure out which actions should be reinforced or suppressed.

In an attempt to obtain dense rewards, researchers investigated the usage of inverse reinforcement learning (Hou et al., 2021; Huang et al., 2020; Li et al., 2020; Park et al., 2020; Takanobu et al., 2019). Inverse RL assumes a dataset of expert interactions and retrieves a reward function that these experts implicitly maximized. Once an approximate reward function is retrieved, the policy can be optimized using the dense reward to learn to imitate the experts. However, this requires expert demonstrations and results in a policy that imitates the experts but does not exceed their performance. Moreover, it is unclear whether the learned reward function would provide correct feedback when applied to new domains.

Another solution for obtaining more informative feedback is the usage of reward shaping. In reward shaping, the reward r in every turn is augmented by a shaping function F and the RL agent uses the shaped reward

$$r' = r + F \tag{4.2}$$

instead of r for optimization (Ng et al., 1999). The shaping function F often encodes expert knowledge that complements the sparse reward r. For instance, a positive reward can be provided if the policy requests a slot that has not been requested before or informs on a slot that has been requested by the user (Kwan et al., 2023; Tseng et al., 2021). In contrast to the hand-crafted expert rules, Su et al. (2015) leverages the scalar outputs of a RNN for modeling the shaping function F. The RNN is trained such that the sum of outputs matches the overall return of the dialogue. However, this method only leverages the environment reward r, also called extrinsic reward, but no additional information. Instead of solely leveraging the extrinsic reward of the environment, the RL agent can produce its own reward, called intrinsic reward, and utilize it for reward shaping. Dialogue policy research has explored the usage of curiosity (or surprise) as intrinsic reward (Lipton et al., 2018; Wesselmann et al., 2019). It's goal is to drive exploration of the agent, thus learning more about the underlying environment. Curiosity aids learning by encouraging to collect more diverse, previously unknown experience so that the agent can find the best possible behavior. Curiosity thus does not directly guide the agent towards useful behavior for the task at hand but only provides more diverse experience. Consequently, it does not serve as an additional feedback signal towards the goal.

4.6 Feudal Dialogue Management

The sparse reward, which encodes task success or failure, decelerates the learning speed as the dialogue agent requires many environment interactions to figure out which actions contributed to success or failure. This is especially problematic in environments with large action spaces such as dialogue due to the trial-and-error process of RL, where the evaluation of an action requires taking the action and observing the outcome. Intuitively, in task-oriented dialogue, the task of fulfilling a user goal can be decomposed into 1) learning about the user goal, and 2) fulfilling the user goal. The hierarchical approach Feudal Dialogue Management (FDM) (Casanueva et al., 2018) utilizes this idea and takes a step towards more efficient learning by decomposing the dialogue policy action space into two sub-spaces $\mathcal{A}_i$ and $\mathcal{A}_g$. The purpose of actions in $\mathcal{A}_i$ is to obtain more information about the user goal by `confirming` or `requesting` the value of a slot for instance. The second action space $\mathcal{A}_g$ is comprised of all remaining actions in $\mathcal{A} \setminus \mathcal{A}_i$, which will be used for fulfilling the user goal, such as informing the value of a slot or booking an entity. Moreover, a master action space $\mathcal{A}_m = \{a_i, a_g\}$ is defined for choosing between actions in $\mathcal{A}_i$ and $\mathcal{A}_g$. FDM optimizes three different policies π_i, π_g and π_m that operate on the action spaces $\mathcal{A}_i, \mathcal{A}_g$ and $\mathcal{A}_m$. Given a belief state b_t, the policy π_m selects an action from $\mathcal{A}_m = \{a_i, a_g\}$, thereby deciding whether to use policy π_i or π_g for selecting the action that will be passed to the environment.

This construction results in smaller action spaces for the individual policies and has been shown to lead to a better final performance. However, the policies are optimized using solely the extrinsic reward measuring task success or failure, which can lead to training instabilities and reduced convergence speed due to the following reason. Since only the policy π_g is able to take actions that lead to goal fulfillment, appropriate actions of π_i can be incorrectly suppressed. For instance, imagine that π_i correctly requested information for unknown slots and reduced uncertainty through confirmations. Nevertheless, π_g failed to inform on requested values, which led to a dialogue failure and thus the suppression of the taken actions. The behavior of π_i will be consequently suppressed in this case, while the same behavior will be reinforced in the case that π_g managed to provide the correct information.

4.7 Continual Reinforcement Learning for Dialogue

The amount of tasks a dialogue system can help users with is almost endless as dialogue is the fundamental way of communication for humans. Moreover, the dialogue system will encounter different user behaviors over time as well as fluctuating user goals, for instance due to seasonal changes or unexpected circumstances such as pandemics. Consequently, a dialogue system constantly needs to adapt to the current circumstances and grow as the amount of tasks it is required to assist with grows. In other words, a dialogue system needs to possess continual learning capabilities. In terms of continual RL, the MDP $\mathcal{M}_{\text{CRL}}$ that defines the environment for the dialogue policy is changing in every aspect:

- the state and action space grow with every newly introduced domain as new information must be comprehended and new actions must be taken
- the transition probabilities change over time depending on the user goal distributions and user behaviors
- the reward function can change if multiple domains are included in a user goal.

Furthermore, different domains possibly exhibit overlapping state information and actions. For instance, the hotel area and restaurant area convey the same meaning, only for different domains. Similarly, the act of informing the address is the same for both hotel or restaurant. A dialogue system that has learned how to answer requests for hotels can thus use this knowledge to improve on other domains as well. This fact urges the need for the dialogue system to possess forward and backward transfer capabilities.

Continual learning is a thriving research field with multiple published surveys (De Lange et al., 2022; Hadsell et al., 2020; Khetarpal et al., 2022; Mazumder and Liu, 2022; Parisi et al., 2019) and frameworks (Lomonaco et al., 2021; Normandin et al., 2021; Powers et al., 2022; Wołczyk et al., 2021). Despite that progress and the pressing need for task-oriented dialogue systems to continually learn over their lifetime, these systems have been barely touched by the topic. There have been so far only few works considering continual supervised learning of dialogue system components. Wu et al. (2019) proposes an architecture based on RNNs for dialogue state tracking that does not require a pre-defined ontology and prevents forgetting by leveraging a regularization based method (Kirkpatrick et al., 2017). Zhu et al. (2022a) address continual learning for dialogue state tracking and learn input embeddings for every new task that are fed into a frozen pre-trained model. While this approach is parameter-efficient, it relies on the fact that the pre-trained model is powerful enough to potentially solve any task without further training if only the input is appropriate. Liu et al. (2021) proposes a combination of replaying old data and multiple distillation losses for preventing forgetting in dialogue state tracking.

Continual learning has been studied in natural language generation, where previous utterances are saved for replay, leading to a continual growth in memory requirements (Mi et al., 2020). Geng et al. (2021) proposes a pruning, expanding and masking strategy and evaluates it on continual learning for natural language generation. In the pruning step, neural network weights with low absolute value are set to zero, effectively freeing up neural network capacities. Nevertheless, every pruning step requires a subsequent re-training to regain the original performance. Lee (2017) proposes a task-independent neural architecture with an action selector. The action selector is a ranking model that calculates the similarity between state and candidate actions and enables action selection even if the action space grows. Madotto et al. (2021) introduce an architecture called AdapterCL and train it in a supervised fashion for intent prediction, state tracking, generation and end-to-end learning, consequently omitting the dialogue policy component. The AdapterCL network becomes larger with every additional task and focuses on preventing catastrophic forgetting by freezing network weights for previous tasks, which precludes backward transfer. As opposed to the above mentioned approaches, this thesis considers continual RL to optimize a dialogue policy.

4.8 Conclusion

This chapter framed dialogue as a reinforcement learning problem by viewing it as a POMDP and subsequently a belief MDP. The dialogue system has to retrieve a dialogue belief state based on the observations, which are given through the user utterances. The dialogue state contains information about the predicted user goal, the predicted user action as well as the dialogue history and database results. Based on this information, the dialogue policy selects an action, which is comprised of a list of atomic actions. The objective of the dialogue policy is to take a sequence of actions in order to maximize the sum of rewards, where the reward encodes dialogue success and efficiency. The reward definition renders dialogue as a sparse reward problem, which requires additional methods such as intrinsic rewards to increase sample efficiency. Furthermore, we discussed the urge to equip dialogue systems with continual learning capabilities due to the large amount of tasks a task-oriented dialogue system can potentially assist with.

Chapter 5

What does the User Want? Information Gain for Hierarchical Dialogue Policy Optimisation

This chapter summarizes our work on information gain as intrinsic reward for hierarchial dialogue policy optimisation and gives a verbatim copy of our paper (Geishauser et al., 2021):

5.1 Summary

Feudal dialogue management (see Section 4.6) is a hierarchical method that aims for more efficient learning by optimizing sub-policies π_i and π_g for the tasks of information seeking and goal fulfillment. The optimization of π_i based on the extrinsic reward measuring dialogue success can lead to training instabilities since dialogue success is highly affected by the behavior of π_g.

In this work we propose the usage of an intrinsic reward based on information gain in order to address this issue. Information gain compares the user goal belief states inferred from the belief state tracker across successive states, yielding a higher reward for greater dissimilarity. This encodes the fact that a change in the user goal can only happen if new information has been obtained. Information gain reinforces actions if they gather new information of the user or resolve uncertainty, such as request or confirmation actions for appropriate slots. It thus defines a reward signal that directly aids in learning about the actual task at hand since any goal can be only fulfilled if it is known in the first place. We train the information seeking policy π_i to maximize information gain instead of the extrinsic reward measuring goal success, consequently correctly rewarding the behavior of π_i.

We show in a multitude of experiments with various domains and noise setups that information gain leads to improved learning speed as well as better final performance. Moreover, interactions with humans reveal that the policy using information gain appropriately asks for information if necessary, leading to a better user experience.

5.2 Personal contributions

The implementation, technical results and writing are my contribution. Songbo Hu and Hsien-Chin Lin were significantly involved in the human evaluation. Co-authors assisted in writing and proofreading.

WHAT DOES THE USER WANT? INFORMATION GAIN FOR HIERARCHICAL DIALOGUE POLICY OPTIMISATION

C. Geishauser[1], *S. Hu*[2], *H. Lin*[1], *N. Lubis*[1], *M. Heck*[1], *S. Feng*[1], *C. van Niekerk*[1], *M. Gašić*[1]
[1]*Heinrich Heine University Düsseldorf*, Germany
[2]*Department of Computer Science and Technology, University of Cambridge*, UK
{geishaus, linh, lubis, heckmi, shutong.feng, niekerk, gasic}@hhu.de, sh2091@cam.ac.uk

ABSTRACT

The dialogue management component of a task-oriented dialogue system is typically optimised via reinforcement learning (RL). Optimisation via RL is highly susceptible to sample inefficiency and instability. The hierarchical approach called Feudal Dialogue Management takes a step towards more efficient learning by decomposing the action space. However, it still suffers from instability due to the reward only being provided at the end of the dialogue. We propose the usage of an intrinsic reward based on information gain to address this issue. Our proposed reward favours actions that resolve uncertainty or query the user whenever necessary. It enables the policy to learn how to retrieve the users' needs efficiently, which is an integral aspect in every task-oriented conversation. Our algorithm, which we call FeudalGain, achieves state-of-the-art results in most environments of the PyDial framework, outperforming much more complex approaches. We confirm the sample efficiency and stability of our algorithm through experiments in simulation and a human trial.

Index Terms— Dialogue systems, reinforcement learning, information gain

1. INTRODUCTION

Task-oriented dialogue systems are characterised by an underlying task or a goal that needs to be achieved during the conversation in order to help a user, such as managing a schedule or finding and booking a restaurant. For that, a spoken dialogue system needs two key abilities: maintaining the current state of the dialogue (*tracking*) and foreseeing how its actions will impact the conversation (*planning*). Modular dialogue systems therefore have a tracking component to maintain information about the dialogue belief state, and a planning component that models the underlying policy, i.e. the selection of actions [1, 2, 3, 4]. The dialogue belief state defines a probability distribution over states that includes information about user preferences, for instance that a user wants a cheap, Italian restaurant, with the distribution encoding different levels of uncertainty.

To deal with planning, current state-of-the-art dialogue systems [5, 6, 7] optimise the policy via some form of reinforcement learning (RL) [8]. However, dialogue policy optimisation using RL is often sample inefficient and unstable, which is exacerbated by the sparse reward typically given in task-oriented dialogue systems [9, 10, 11, 4]. To tackle the problem of sample inefficiency, hierarchical reinforcement learning has been proposed that subdivides the task temporally or spatially [12, 13], thereby reducing complexity of the task and accelerating learning.

For spoken dialogue systems that help users accomplish any kind of tasks, it is important to understand what the actual user's goal is by asking appropriate and targeted questions. Acquiring information is the first important building block of a conversation. Due to its significance, it is reasonable to learn a dedicated policy π_i that deals with this sub-task. This has been proposed in the hierarchical approach called Feudal Dialogue Management [14], where the extrinsic reward is used for optimising π_i. However, we argue and show that the extrinsic reward may provide misleading feedback signal for π_i that leads to unstable and less efficient learning.

Instead, we equip the policy with an intrinsic reward based on information gain that measures the change in probability distributions between consecutive turns. The dense reward signal encourages the policy to produce actions that resolve uncertainty and to query the user in cases where it is necessary. The policy π_i together with our proposed reward explicitly models how a system can learn to obtain information about the dialogue partner, which is an integral aspect in every conversation.

We conduct our experiments using the PyDial benchmarking environment [15]. Our algorithm *FeudalGain* achieves state-of-the-art results in terms of sample efficiency and final performance in 14 out of 18 environments. We confirm the effectiveness of our method in a human trial, where our system directly interacts with humans.

2. RELATED WORK

Information gain (also known as mutual information) measures the amount of information obtained about one random variable through observing another random variable. Information gain has already been used as feature as well as reward signal for reinforcement learning. In [16], a dialogue policy for clarification is trained using information gain as a policy feature. Information gain has also been used to build decision trees for dialogue systems [17, 18].

Different to the extrinsic reward that is produced by the environment, intrinsic reward is produced by the RL agent itself. The purpose of intrinsic reward is to additionally guide the learning of the agent. Curiosity-driven learning, which encourages exploration of the state-action space by learning more about the environment dynamics, has been interpreted as information gain [19, 20]. In contrast to curiosity that will decrease the more the agent learns about the environment, the information gain we use will always be provided. In [21], the policy is divided into an "explorer" and "exploiter", using an intrinsic reward signal for the explorer that is different from the extrinsic reward for the exploiter. While [21] focus on exploration, we define an intrinsic reward to foster fast learning of an information seeking policy. Note that even when the space is fully explored, one still needs to gather sufficient information about user needs.

"Answerer in Questioner's Mind" [22] selects the question which maximises the information gain of a target class and an answer given the question in a goal-oriented visual dialogue task. In [23, 24], information gain is used as a reward in goal-oriented visual dialogues by leveraging a responder model or guesser. The belief tracker in dialogue can be interpreted as a guesser that needs to guess the correct value for each slot in the belief state, where the policy can ask questions. Different from [23, 24], we use information gain in a hierarchical setting and provide it only to a sub-policy as (intrinsic) reward.

In [14], there is a dedicated sub-policy for information gathering as in our case. It is however only optimised with the extrinsic reward. We introduce an intrinsic reward based on information gain in the hierarchical setting and thus enable fast learning of the user's needs, an integral ability that is often neglected in task-oriented dialogue systems.

3. BACKGROUND

3.1. Dialogue Policy Optimisation via RL

The formal framework in RL is given by a Markov decision process $\mathcal{M} = \{\mathcal{B}, \mathcal{A}, r, p, p_0, \gamma\}$. Here, $\mathcal{B}$ denotes the (continuous) belief state space, $\mathcal{A}$ is the action space, r is the reward function, $p(b'|b, a)$ models the probability of transitioning to state b' after executing action a in state b, and $p_0(b)$ is the probability of starting in state b. The discount factor $\gamma \in [0, 1]$ trades off the importance of immediate and future rewards. At time step t, the agent observes a state b_t, chooses an action a_t according to a policy $\pi(a|b_t)$, transitions to a new state b_{t+1} and observes a reward signal $r_t = r(b_t, a_t, b_{t+1}) \in \mathbb{R}$. The goal of the agent is to maximise the discounted return $R_t = \sum_{i \geq 0} \gamma^i r_{t+i}$ in expectation.

Value-based RL methods [8] optimise the policy by maximising the Q-values $Q^\pi(b_t, a_t) = \mathbb{E}_{b_{t+1:\infty}, a_{t+1:\infty}}[R_t|b_t, a_t]$ for every state-action pair $(b_t, a_t) \in \mathcal{B} \times \mathcal{A}$. One example here is Dueling deep Q-networks (DDQN) [25] that optimises a parameterised Q-network using tuples (b_t, a_t, r_t, b_{t+1}) and a target network for a stochastic gradient step. Policy gradient methods on the other hand parameterise the policy directly and aim at maximising $\mathbb{E}_{b_0}[R_0]$. Actor-critic algorithms [8] build upon policy gradient methods and approximate Q^π with a function approximator, also called critic. The ACER [9] algorithm is one such instance that uses the whole trajectory in order to do an update step for its critic.

To foster exploration during learning, noisy networks [26] inject noise into the neural network by substituting each weight w in the neural network by $\mu + \sigma \cdot \epsilon$, where μ and σ are weights and ϵ is a noise random variable.

We view dialogue as a sequence of turns between a user and a dialogue system. The belief state in dialogue typically includes a probability distribution over values for every slot in the ontology, which expresses how likely it is that the user wants a specific value such as "Italian" food or "expensive" price-range. In each turn of the dialogue, for a given belief state, the system decides which action to take in order to successfully complete the user's goal. This sequential decision-making task can be optimised with RL algorithms. A range of RL algorithms have already been applied to dialogue management, including ACER [27].

3.2. Feudal Dialogue Management

Feudal Dialogue Management [14, 28] is a hierarchical approach for dialogue policy learning that divides the action space $\mathcal{A}$ into two subsets $\mathcal{A}_i$ and $\mathcal{A}_g$. The purpose of actions in $\mathcal{A}_i$ is to obtain more information from the user by *confirming*, *requesting* or *selecting* the value of a slot. The second action set $\mathcal{A}_g$ comprises all other actions, such as general actions like *goodbye* or informing about requested values. Moreover, a master action space $\mathcal{A}_m = \{a_i, a_g\}$ for choosing between actions in $\mathcal{A}_i$ and $\mathcal{A}_g$ is defined. In order to produce an action, a master policy π_m first selects an action from $\mathcal{A}_m$, after which the associated policy π_i or π_g corresponding to $\mathcal{A}_i$ and $\mathcal{A}_g$ is consulted for the final action selection. An additional *pass* action is added to $\mathcal{A}_i$ and $\mathcal{A}_g$, which is taken whenever the other sub-policy is executed. The policy π_i is optimised using a value-based method where the Q-values are produced for every slot $s \in \{s_1, ..., s_n\}$ independently using associated policies π_s. The input to each π_s is the belief state including the value distribution of s. The parameters of the Q-functions are shared among the slots.

To optimise each of the policies, the external reward r_e provided by the environment is used. The reward is -1 in each turn to enforce more efficient dialogues and 0 or 20 in the very last turn for failure or success of the dialogue.

We note that adding the *pass* action for π_i is very important. The reward for a tuple (b_t, a_t, r_t, b_{t+1}) with $a_t \neq$ *pass* that we use for updating π_i will always be -1 or 0 since suc-

	Utterance	p_s for $s =$ *pricerange*	r_e	r_i	π_i	π_g
User	I need a restaurant.	[0,0,0,0,1]				
Sys (π_i)	What pricerange do you like?	[0,0,0,0,1]	-1.0	1.0	*request-pricerange*	*pass*
User	Something cheap please. $\rightarrow$	[0.5,0.3,0.2,0,0]				
Sys (π_i)	Can I confirm that you mean cheap?	[0.5,0.3,0.2,0,0]	-1.0	0.22	*confirm-pricerange*	*pass*
User	Yes. I need the address. $\rightarrow$	[0.95,0.05,0,0,0]				
Sys (π_g)	Goodbye.	[0.95,0.05,0,0,0]	0.0	0.0	*pass*	*bye*

Table 1: Example dialogue in the Cambridge restaurant domain. p_s denotes the probability distribution over values [*cheap*, *moderate*, *expensive*, *dontcare*, *none*] for slot *pricerange* given in the belief state. r_e and r_i denote extrinsic and intrinsic reward given to π_i during the conversation. π_i tries to retrieve information and resolve uncertainty, yet the extrinsic reward gives no guidance at all since π_g ends the dialogue too early. In contrast, our proposed reward r_i correctly rewards the behaviour of π_i. Information gain r_i given in Definition 1 is computed by Jensen-Shannon divergence between consecutive value distributions.

cess can be only achieved if information is provided (which only π_g can do). We hence need to update the policy using tuples where the *pass* action was taken, i.e. reward the policy for doing nothing. We empirically verify the necessity for the *pass* action in Section 6.2.

In the following, we will work with the FeudalACER algorithm [28] as our baseline and abbreviate it as Feudal. Feudal uses ACER for policies π_m and π_g and DDQN for π_i.

4. INFORMATION GAIN IN POLICY LEARNING

4.1. Drawbacks of Extrinsic Reward in Feudal

Recall that π_i merely outputs actions for obtaining information about the user preferences. As this is not enough to complete a task, the policy π_g mainly determines dialogue success or failure. While it is reasonable to provide π_g and π_m with external reward, it is less obvious for π_i. The behaviour of π_i can lead either to reinforcement or suppression if π_g misbehaves. As an illustrative example, Table 1 shows a dialogue where π_i acted correctly but does not obtain any positive feedback from r_e due to dialogue failure in the end.

4.2. Information Gain

How can we fairly reward π_i then? We propose the usage of an intrinsic reward for π_i based on information gain similar to [23]. The idea is that if we take an action to query information about a certain slot (e.g. *request-area*) leading to a change in the value distribution for that slot, new information has been gathered and that behaviour should be reinforced. Formally, we define the intrinsic reward r_i as follows.

Definition 1 *Let $(b, a, b') \in \mathcal{B} \times \mathcal{A}_i \times \mathcal{B}$ be a tuple of state, action and next state where a includes slot s. Let p_s and p'_s be the probability distributions over values for s in b and b', respectively. Let d be a distance function between probability distributions. We define*

$$r_i(b, a, b') := d(p_s, p'_s)$$

as the information gain (IG) when executing action a in state b and observing b'.

This reward encodes the goal of π_i by reinforcing actions that gather new information or resolve uncertainty. It separates learning of π_i from the behaviour of π_g and independently models how a system can learn to obtain information about the user's needs. Moreover, the reward guides the policy at every step in contrast to the sparse reward that first has to be back-propagated. Due to the immediate feedback, the additional *pass* action becomes obsolete for π_i and we do not need to update with tuples $(b_t, pass, r_t, b_{t+1})$ anymore. The policy can now quickly learn how to obtain the user preferences, which is the first important step towards a successful dialogue. An example for computing r_i together with the chosen actions is depicted in Table 1. In contrast to the external reward, information gain reinforces the desired behaviour of π_i even though the dialogue failed. Since the probability distribution is part of the input to the policy, π_i can easily build the relation between the state and the reward. We note that this reward is only defined and should be only used for actions that seek to obtain information of the user. Otherwise it might happen that the user pro-actively provides information and an unrelated action gets rewarded. As a result, the reward can be applied to all scenarios where obtaining information from a user is important. This is especially the case for task-oriented dialogue systems where our focus lies, but also holds in many more scenarios (conducting an interview, getting to know a person in chit-chat) as dialogue is generally an exchange of information.

We remark that the usage of our reward is not restricted to hierarchical RL and it can also be used as an additional signal to the external reward. We also emphasise that our dense reward aids the policy in learning to reach the main goal, i.e. task completion. It thus differs from rewards based on curiosity or surprise [19, 29] that aim for enhanced exploration (which decreases as the agent learns more about the environment). Our reward can be used in tandem with other rewards, in particular the ones for exploration.

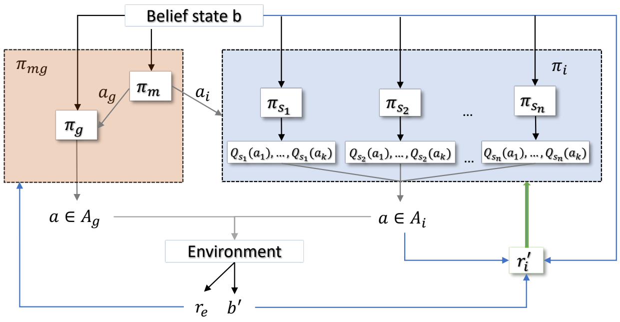


Fig. 1: The architecture of our FeudalGain algorithm together with the action selection process and reward computation of r_i'. FeudalGain uses the intrinsic reward r_i' as given in Equation (1) instead of the external reward r_e for optimising π_i. Also different from Feudal, FeudalGain merges the policies π_m and π_g into a single policy π_{mg} with action space $\mathcal{A}_g \cup \{a_i\}$.

4.3. FeudalGain

After introducing our intrinsic reward based on information gain, we now present our FeudalGain algorithm. Most importantly, we substitute the extrinsic reward for our proposed information gain to optimise the policy π_i.

We choose Jensen-Shannon divergence (JS) as our distance function d, since it is bounded between 0 and 1, symmetric and defined everywhere, in contrast to Kullback-Leibler-divergence (KL) [30]. JS for two probability distributions p and q is defined as

$$\begin{aligned} \mathrm{JS}(p,q) &= \frac{1}{2}(\mathrm{KL}[p||m] + \mathrm{KL}[q||m]), \\ m &= \frac{1}{2}(p+q) \end{aligned}$$

In our experiments, we found that rewarding behaviour if information gain exceeds a certain threshold δ works better than directly using r_i. We henceforth work with the reward

$$r_i'(b,a,b') = \begin{cases} 1, & \text{if } r_i(b,a,b') \geq \delta \\ -1, & \text{otherwise} \end{cases} \quad (1)$$

with variables as in Definition 1.

ACER uses the full trajectory to update its critic, which is why π_g needs to take an action in every turn. Feudal solves this by π_g taking the *pass* action whenever π_i takes an action as shown in Table 1. However, we can avoid that additional action by merging the policies π_m and π_g into a single policy π_{mg} with action space $\mathcal{A}_g \cup \{a_i\}$, which we employ in our full algorithm.

Our final algorithm FeudalGain thus uses π_{mg} and π_i together with intrinsic reward r_i' for π_i. Our full algorithm is depicted in Figure 1.

5. EXPERIMENTAL SETUP

We implement FeudalGain[1] in the PyDial toolkit [31]. The performance is evaluated using the PyDial benchmarking environments [15] comprising 18 settings, which are distinguished by domain and different semantic error rates, action masks and user simulator configurations. Unlike other publicly available dialogue toolkits, PyDial uses a belief tracker that outputs probability distributions rather than binary states, which enables more expressive distribution comparisons.

Instead of the ϵ-greedy approach for exploration that was used for Feudal, we use noisy networks similar to [5]. The threshold δ for all our experiments in Section 6.1 is set to 0.2. The FeudalGain policies π_{mg} and π_i are trained with ACER and DDQN, respectively. For each environment, the algorithms are trained on 4000 dialogues with 10 different seeds. After every 200 training dialogues, the algorithms are evaluated on 500 dialogues. The average reward that is shown is always the extrinsic reward r_e. The dialogues for our human trial are collected using DialCrowd [32]. The simulated user experiments are done on semantic level using the default focus belief tracker, while the user trial is performed on text-level using an additional template based natural language generation module.

We compare FeudalGain to Feudal [28] and the current state-of-the-art algorithm STRAC [5]. STRAC uses a hierarchical decision-making model for policy optimisation with implicit policy decomposition and noisy networks for exploration. STRAC offers two different modes: a single-domain version STRAC-S that is trained and evaluated on a single domain and STRAC-M that is trained on three different domains but evaluated only on a single domain. STRAC-M trains on three times as many dialogues as STRAC-S and FeudalGain. We do not compare to the work of [6] as they use a hand-coded expert during training. For completeness, we also add the performance of a hand-coded policy (HDC), which is already implemented in PyDial.

6. RESULTS

6.1. Results on FeudalGain

We compare FeudalGain to STRAC-S in terms of sample efficiency and final performance. Table 2 shows success rate and average reward after 400 and 4000 dialogues.

FeudalGain has higher sample efficiency than STRAC-S in almost all settings and is comparable to STRAC-M although STRAC-M uses three times as many dialogues. This can be attributed to the immediate reward provided by our information gain that correctly guides π_i in every turn.

Similar conclusions can be drawn after 4000 dialogues, where FeudalGain is even able to outperform STRAC-M. Information gain hence not only helps in securing more sam-

[1] Our code will be released at https://pydial.cs.hhu.de/.

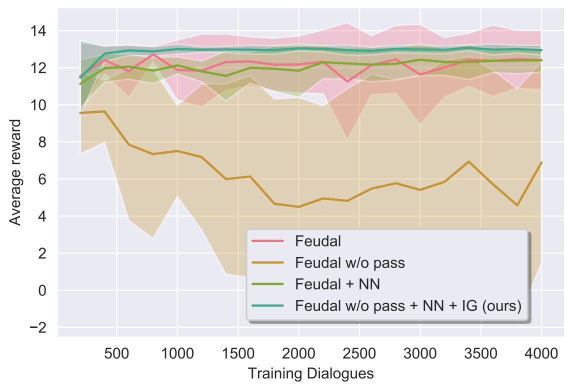


Fig. 2: Ablation study for FeudalGain. "W/o pass" ablates the additional *pass* action for π_i. NN denotes noisy networks. IG denotes our proposed information gain addition.

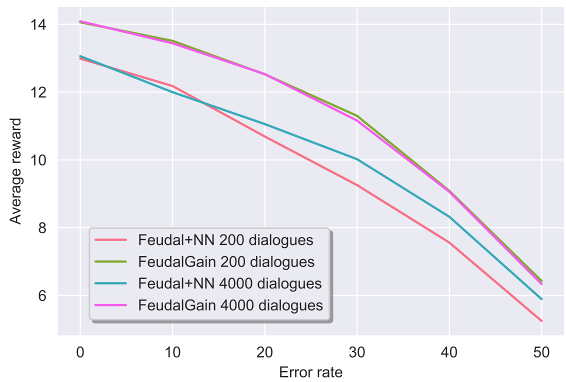


Fig. 3: Robustness test of FeudalGain against Feudal+NN for increasing semantic error rate. We use policies that trained for 200 dialogues and 4000 dialogues averaged over 15 seeds.

Task		FeudalGain Suc.	FeudalGain Rew.	STRAC-S Suc.	STRAC-S Rew.	STRAC-M Suc.	STRAC-M Rew.	HDC Suc.	HDC Rew.
After 400 dialogues									
	CR	**99.8**	**14.2**	97.7	13.1	99.7	14.0	100.0	14.1
Env1	SFR	95.8	11.6	**98.2**	**12.3**	99.2	12.9	97.6	12.1
	LAP	96.1	11.5	**98.5**	**12.3**	98.6	12.2	97.2	11.8
	CR	**86.8**	**10.7**	65.5	5.0	90.3	10.2	100.0	14.1
Env2	SFR	**89.6**	**10.5**	69.8	4.4	87.5	9.0	97.6	12.4
	LAP	**84.7**	**9.7**	56.9	1.6	89.2	9.1	97.8	11.7
	CR	**97.6**	**12.7**	97.2	12.5	97.3	12.7	95.2	10.8
Env3	SFR	**91.4**	**9.1**	90.4	8.9	93.6	10.5	90.2	8.9
	LAP	91.2	9.3	**92.5**	**9.7**	92.4	9.6	88.2	8.4
	CR	**82.9**	**8.9**	71.0	5.2	75.3	6.6	97.0	11.1
Env4	SFR	**84.1**	**8.7**	72.7	5.0	77.2	6.4	89.2	8.2
	LAP	**82.3**	**8.0**	65.9	3.1	79.8	6.9	88.6	8.4
	CR	95.0	**11.1**	**95.3**	10.6	95.6	10.8	94.6	9.2
Env5	SFR	**87.1**	**6.9**	80.6	4.5	88.8	7.5	87.6	6.3
	LAP	87.6	**6.6**	**87.8**	6.1	86.0	5.6	82.8	4.6
	CR	**92.8**	**10.9**	91.9	10.3	90.7	9.9	91.2	9.5
Env6	SFR	**79.4**	**5.6**	78.5	4.9	83.8	6.6	80.2	6.5
	LAP	81.7	6.1	**84.6**	**6.6**	81.7	5.7	76.6	5.6
	CR	**92.5**	**11.4**	86.4	9.5	91.5	10.7	96.3	11.5
Mean	SFR	**87.9**	**8.7**	81.7	6.7	88.3	8.8	90.4	9.1
	LAP	**87.3**	**8.5**	81.0	6.6	88.0	8.2	88.5	8.4
After 4000 dialogues									
	CR	**99.9**	**14.1**	99.8	**14.1**	99.8	14.1	100.0	14.1
Env1	SFR	**99.0**	**12.7**	98.7	**12.7**	98.5	12.7	97.6	12.1
	LAP	**97.6**	**12.0**	**97.6**	**12.0**	97.8	12.0	97.2	11.8
	CR	**98.0**	**13.3**	97.9	13.1	98.4	13.1	100.0	14.1
Env2	SFR	**98.8**	**13.7**	95.6	12.1	97.5	13.0	97.6	12.4
	LAP	**98.5**	**13.3**	92.6	11.6	98.0	12.8	97.8	11.7
	CR	**98.6**	**13.0**	98.1	**13.0**	97.9	12.9	95.2	10.8
Env3	SFR	**95.1**	10.4	91.9	**10.5**	93.0	10.6	90.2	8.9
	LAP	**91.2**	9.5	90.7	**9.7**	92.1	9.9	88.2	8.4
	CR	**98.0**	**12.5**	92.9	11.5	91.3	10.8	97.0	11.1
Env4	SFR	**93.1**	**11.2**	90.2	10.7	89.2	10.5	89.2	8.2
	LAP	**91.8**	**11.1**	86.3	9.2	89.7	10.4	83.6	8.4
	CR	**97.6**	**11.9**	97.1	11.8	96.5	11.7	94.6	9.2
Env5	SFR	88.4	7.1	**89.6**	**8.4**	90.1	8.4	87.6	6.3
	LAP	87.6	6.6	**88.2**	**6.9**	88.5	7.0	82.8	4.6
	CR	**94.0**	**11.0**	92.5	**11.0**	91.5	10.7	91.2	9.5
Env6	SFR	**88.2**	**7.9**	81.6	7.0	84.5	7.3	80.2	6.5
	LAP	**83.4**	**6.8**	83.3	6.7	83.2	6.7	76.6	5.6
	CR	**97.7**	**12.6**	96.4	12.4	95.9	12.2	96.3	11.5
Mean	SFR	**93.8**	**10.5**	91.3	10.2	92.1	10.4	90.4	9.1
	LAP	**91.7**	**9.9**	89.8	9.4	91.6	9.8	88.5	8.4

Table 2: Success rate and average reward (using only r_e) for our proposed approach FeudalGain against STRAC-S. Best performance is marked in bold. Algorithms were tested in the Cambridge Restaurant (CR), San-Francisco Restaurant (SFR) and Laptops (LAP) domain. We included STRAC-M and a hand-coded policy (HDC) as supplemental comparison. Note that STRAC-M is trained in three domains and therefore utilized three times the amount of data compared to STRAC-S and FeudalGain. Results of STRAC were taken from [5].

ple efficient learning but also for achieving high final performance. FeudalGain excels in difficult environments, namely 2 and 4, where unreasonable actions are not masked. FeudalGain also performs very well in environment 6 that exhibits a very high noise level of 30%, which shows that information gain is robust to high error rates. Final performance is slightly worse in environment 5, where an "unfriendly" user simulator is used. The results show that policy optimisation can significantly benefit from our reward based on information gain.

6.2. Ablation Study

We conduct an ablation study for FeudalGain to investigate the difference in stability and convergence speed due to our proposed changes. We conduct experiments in environment 3 that exhibits a semantic error rate of 15%. We chose this environment as it is close to human experiment characteristics [33]. Figure 2 depicts our findings.

We first ablate the *pass* action for π_i by only updating π_i with tuples (b_t, a_t, r_t, b_{t+1}) where $a_t \neq$ *pass* to empirically verify that it is needed in order to back-propagate the final reward to actions taken by π_i. The algorithm is not capable of learning without *pass* when only the extrinsic reward is used.

While usage of noisy networks can stabilise the learning for Feudal, the true benefit comes from the addition of information gain that results in fast and smooth convergence after as few as 500 dialogues. The dependence on the seed almost vanishes when introducing information gain, hence more sta-

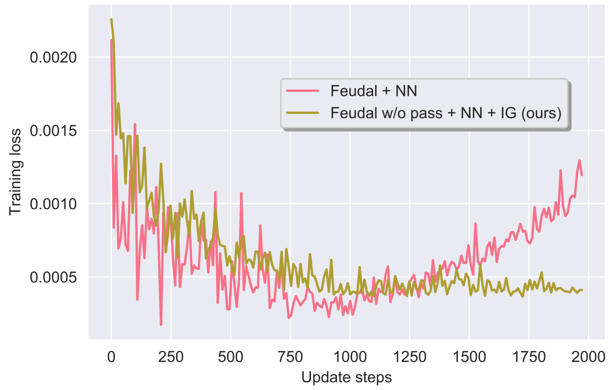


Fig. 4: Training loss for Feudal [28] using noisy networks (NN) and our proposed information gain reward (IG). We approximated the loss in every update step by taking 512 samples from the replay buffer.

ble learning and robustness against randomness in the initialisation is achieved. The usage of a single policy π_{mg} in addition to information gain only led to a small performance difference in the first 200 dialogues. We hence omitted it in Figure 2 for better readability.

We further test the robustness of FeudalGain for increasing amount of semantic error rates by comparing it against Feudal+NN that does not use information gain. Results are depicted in Figure 3, for policies trained for 200 and 4000 dialogues, averaged over 15 seeds. For 200 training dialogues, the difference between FeudalGain and the baseline is consistent across noise levels. For 4000 training dialogues, the baseline catches up a little but does not outperform FeudalGain on any of the noise levels.

Lastly, we want to empirically verify that the usage of extrinsic reward may provide misleading feedback for π_i, resulting in incorrect policy updates. Figure 4 shows the training loss for policy π_i using Feudal with noisy networks and information gain. Substituting r_e for r_i' leads to quick and stable convergence of the algorithm with weaker oscillations.

6.3. Human Evaluation

In order to show that our results transfer from simulation to humans, we compare FeudalGain against Feudal with noisy networks (Feudal + NN) in a human trial, where users directly interact with the two policies. We collected 400 dialogues using each policy. We took the policies after only 200 training dialogues that were closest to the average performance in environment 3. The reward was 11.7 for Feudal + NN and 12.9 for FeudalGain on the simulated user. We chose such a small number of training dialogues to examine the sample efficiency of FeudalGain. At the end of each interaction, we asked users if the dialogue was successful, whether the system asked for information when necessary ("AskIfNec") and what the overall performance was ("Overall"). Table 3 shows that superior performance of FeudalGain in terms of success and number of turns in simulation translates to real users. More interestingly, the rating if information was requested when necessary is much higher, which confirms that our intrinsic reward enables π_i to learn guided information gathering. The overall rating ("Overall") is correlated to "AskIfNec", showing how important guided information gathering is for the overall perception of the system. The reduced standard deviation shows the stability of our approach.

	Success	Turns	AskIfNec	Overall
FeudalGain	0.71/0.45*	6.5/3.4*	3.8/1.4*	3.7/1.5*
Feudal + NN	0.43/0.5	8.1/5.0	3.0/1.5	2.7/1.6

Table 3: Mean/standard deviation for success, number of turns, whether the system asked for information when necessary and overall performance according to human evaluation. *We used the t-test to check statistical significance, where $p < 0.05$.

7. CONCLUSION

We proposed the use of intrinsic reward within the hierarchical Feudal Dialogue Management approach for the information seeking policy. Our new architecture gracefully deals with shortcomings such as artificial *pass* actions and misleading reward signals that lead to sample inefficiency and instability. Our proposed reward encourages the policy to seek useful information from the user and puts more emphasis on the user's needs, which is an integral part of dialogue systems that aid the user in solving any kinds of tasks. We show in experiments with simulated users that incorporating our reward improves sample efficiency, stability, and quality of the resulting policy, and that our algorithm FeudalGain algorithm leads to state-of-the-art results for the PyDial benchmark. We confirm the results in a human trial where volunteers interacted with our policy. Our results warrant a more widespread use of intrinsic reward in task-oriented dialogue systems.

In future work, we like to scale our approach to multiple domains and learn the hierarchical structure automatically.

Acknowledgments

This work is a part of DYMO project which has received funding from the European Research Council (ERC) provided under the Horizon 2020 research and innovation programme (Grant agreement No. STG2018 804636). N. Lubis, C. van Niekerk, M. Heck and S. Feng are funded by an Alexander von Humboldt Sofja Kovalevskaja Award endowed by the Federal Ministry of Education and Research. Computational resources were provided by Google Cloud.

8. REFERENCES

[1] Esther Levin and Roberto Pieraccini, "A stochastic model of computer-human interaction for learning dialogue strategies," in *EUROSPEECH 97*, 1997, pp. 1883–1886.

[2] Nicholas Roy, Joelle Pineau, and Sebastian Thrun, "Spoken dialogue management using probabilistic reasoning," in *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, USA, 2000, ACL '00, p. 93–100, Association for Computational Linguistics.

[3] Jason D. Williams and Steve Young, "Partially observable markov decision processes for spoken dialog systems," *Comput. Speech Lang.*, vol. 21, no. 2, pp. 393–422, Apr. 2007.

[4] Zheng Zhang, Ryuichi Takanobu, Minlie Huang, and Xiaoyan Zhu, "Recent advances and challenges in task-oriented dialog system," *CoRR*, vol. abs/2003.07490, 2020.

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

[6] Thibault Cordier, Tanguy Urvoy, Lina M. Rojas-Barahona, and Fabrice Lefèvre, "Diluted near-optimal expert demonstrations for guiding dialogue stochastic policy optimisation," in *34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada*, 2020.

[7] Nurul Lubis, Christian Geishauser, Michael Heck, Hsien-chin Lin, Marco Moresi, Carel van Niekerk, and Milica Gašić, "LAVA: Latent action spaces via variational auto-encoding for dialogue policy optimization," in *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online), Dec. 2020, pp. 465–479, International Committee on Computational Linguistics.

[8] Richard S. Sutton and Andrew G. Barto, *Reinforcement learning: an introduction*, The MIT Press, second edition, 2018.

[9] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Rémi Munos, Koray Kavukcuoglu, and Nando de Freitas, "Sample efficient actor-critic with experience replay," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017, OpenReview.net.

[10] Gabriel Dulac-Arnold, Daniel J. Mankowitz, and Todd Hester, "Challenges of real-world reinforcement learning," *Reinforcement Learning for Real Life (RL4RealLife) Workshop in the 36th International Conference on Machine Learning, Long Beach, California, USA, 2019*, 2019.

[11] Hao Dong Zihan Ding, "Challenges of reinforcement learning," in *Deep Reinforcement Learning: Fundamentals, Research, and Applications*, Shanghang Zhang Hao Dong, Zihan Ding, Ed., chapter 7, pp. 249–272. Springer Nature, 2020, http://www.deepreinforcementlearningbook.org.

[12] Richard S. Sutton, Doina Precup, and Satinder Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, no. 1, pp. 181–211, 1999.

[13] Peter Dayan and Geoffrey E Hinton, "Feudal reinforcement learning," in *Advances in Neural Information Processing Systems*, S. Hanson, J. Cowan, and C. Giles, Eds. 1993, vol. 5, Morgan-Kaufmann.

[14] Iñigo Casanueva, Paweł Budzianowski, Pei-Hao Su, Stefan Ultes, Lina M. Rojas-Barahona, Bo-Hsiang Tseng, and Milica Gašić, "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)*, New Orleans, Louisiana, June 2018, pp. 714–719, Association for Computational Linguistics.

[15] Iñigo Casanueva, Paweł Budzianowski, Pei-Hao Su, Nikola Mrkšić, Tsung-Hsien Wen, Stefan Ultes, Lina Rojas-Barahona, Steve Young, and Milica Gašić, "A benchmarking environment for reinforcement learning based task oriented dialogue management," vol. abs/1711.11023, 2018.

[16] Aishwarya Padmakumar and Raymond J. Mooney, "Dialog policy learning for joint clarification and active learning queries," in *The AAAI Conference on Artificial Intelligence (AAAI)*, February 2021.

[17] Joseph Polifroni and Marilyn Walker, "Learning database content for spoken dialogue system design," in *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy, May 2006, European Language Resources Association (ELRA).

[18] Paul Fodor, "Domain directed dialogs for decision processes," *CoRR*, vol. abs/cs/0703072, 2007.

[19] Rein Houthooft, Xi Chen, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel, "Vime: Variational information maximizing exploration," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds. 2016, vol. 29, Curran Associates, Inc.

[20] Zachary Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng, "BBQ-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018.

[21] Louis Bagot, Kevin Mets, and S. Latré, "Learning intrinsically motivated options to stimulate policy exploration," in *Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020.*, 2020.

[22] Sang-Woo Lee, Yu-Jung Heo, and Byoung-Tak Zhang, "Answerer in questioner's mind: Information theoretic approach to goal-oriented visual dialog," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. 2018, vol. 31, Curran Associates, Inc.

[23] Pushkar Shukla, Carlos E. L. Elmadjian, Richika Sharan, Vivek Kulkarni, Matthew A. Turk, and William Yang Wang, "What should I ask? using conversationally informative rewards for goal-oriented visual dialog," in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, Anna Korhonen, David R. Traum, and Lluís Màrquez, Eds. 2019, pp. 6442–6451, Association for Computational Linguistics.

[24] Ehsan Abbasnejad, Iman Abbasnejad, Qi Wu, Javen Shi, and Anton van den Hengel, "Gold seeker: Information gain from policy distributions for goal-oriented vision-and-langauge reasoning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. 2020, pp. 13447–13456, IEEE.

[25] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas, "Dueling network architectures for deep reinforcement learning," in *Proceedings of the 33rd International Conference on Machine Learning - Volume 48*. 2016, ICML'16, p. 1995–2003, JMLR.org.

[26] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alexander Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg, "Noisy networks for exploration," in *Proceedings of the International Conference on Representation Learning (ICLR 2018)*, Vancouver (Canada), 2018.

[27] Gellért Weisz, Paweł Budzianowski, Pei-Hao Su, and Milica Gašić, "Sample efficient deep reinforcement learning for dialogue systems with large action spaces," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 11, pp. 2083–2097, 2018.

[28] Iñigo Casanueva, Paweł Budzianowski, Stefan Ultes, Florian Kreyssig, Bo-Hsiang Tseng, Yen-chen Wu, and Milica Gašić, "Feudal dialogue management with jointly learned feature extractors," in *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, Melbourne, Australia, July 2018, pp. 332–337, Association for Computational Linguistics.

[29] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. 2017, ICML'17, p. 2778–2787, JMLR.org.

[30] Solomon Kullback and Richard A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79 – 86, 1951.

[31] Stefan Ultes, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Dongho Kim, Iñigo Casanueva, Paweł Budzianowski, Nikola Mrkšić, Tsung-Hsien Wen, Milica Gašić, and Steve Young, "PyDial: A multi-domain statistical dialogue system toolkit," in *Proceedings of ACL 2017, System Demonstrations*, Vancouver, Canada, July 2017, pp. 73–78, Association for Computational Linguistics.

[32] Kyusong Lee, Tiancheng Zhao, Alan W. Black, and Maxine Eskenazi, "DialCrowd: A toolkit for easy dialog system assessment," in *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, Melbourne, Australia, July 2018, pp. 245–248, Association for Computational Linguistics.

[33] Steve Young, Catherine Breslin, Milica Gašić, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Eli Hancock, *Evaluation of Statistical POMDP-Based Dialogue Systems in Noisy Environments*, pp. 3–14, 04 2016.

Chapter 6

Dynamic Dialogue Policy for Continual Reinforcement Learning

This chapter summarizes our work on dynamic dialogue policies for continual reinforcement learning and gives a verbatim copy of our paper (Geishauser et al., 2022):

Christian Geishauser et al. (Oct. 2022). "Dynamic Dialogue Policy for Continual Reinforcement Learning". In: *Proceedings of the 29th International Conference on Computational Linguistics*. Gyeongju, Republic of Korea: International Committee on Computational Linguistics, pp. 266–284. URL: `https://aclanthology.org/2022.coling-1.21`

6.1 Summary

As mentioned in Section 4.7, the dialogue policy requires continual learning abilities due to the large amount of potential domains it can experience in its lifetime. As the state and action space of the dialogue policy grows over time due to the introduction of new domains, the dialogue policy in particular requires an architecture that can incorporate new state information and actions seamlessly and allows forward transfer to new domains.

In this work we conduct the first study on continual reinforcement learning for dialogue policies. We study the continual learning setup where domains are introduced sequentially and only one domain is present for a certain time-period, which allows us to investigate the effects of forward transfer to new domains and catastrophic forgetting of how to operate in already observed domains. Moreover, we center our attention on architectural considerations to address the continual learning challenges arising from growing state and action spaces.

Our contributions are a continual RL framework for dialogue policies consisting of two baseline architectures (Xu et al., 2020; Zhu et al., 2020), the state-of-the-art continual RL algorithm CLEAR (Rolnick et al., 2019) and continual learning metrics for studying forward transfer and forgetting. Moreover, we propose the dynamic dialogue policy transformer (DDPT), an architecture specifically designed for addressing the challenges of continual RL for dialogue policies. DDPT consists of a Transformer encoder-decoder structure and utilizes a pre-trained language model for embedding state information and actions. The Transformer together with the language model allows us to process a dynamic input and produce a dynamic output as well as incorporate new information and actions seamlessly without any growth in network parameter size. In order to reduce the effective state space during acting, DDPT leverages hard-attention on unnecessary information. Moreover, DDPT is equipped with a domain gate that facilitates forward transfer to new domains.

We conduct experiments with different domain sequences and user simulators and show that DDPT is superior in absolute performance and shows significant forward transfer and robustness against forgetting. We validate the strengths of DDPT in a human trial, which shows that DDPT is on par with an expert model that requires a model for every domain.

6.2 Personal contributions

The implementation, technical results and writing are my contribution. Co-authors assisted in writing and proofreading.

Dynamic Dialogue Policy for Continual Reinforcement Learning

Christian Geishauser, Carel van Niekerk, Hsien-Chin Lin,
Nurul Lubis, Michael Heck, Shutong Feng, Milica Gašić
Heinrich Heine University Düsseldorf, Germany
{geishaus, niekerk, linh, lubis, heckmi, fengs, gasic}@hhu.de

Abstract

Continual learning is one of the key components of human learning and a necessary requirement of artificial intelligence. As dialogue can potentially span infinitely many topics and tasks, a task-oriented dialogue system must have the capability to continually learn, dynamically adapting to new challenges while preserving the knowledge it already acquired. Despite the importance, continual reinforcement learning of the dialogue policy has remained largely unaddressed. The lack of a framework with training protocols, baseline models and suitable metrics, has so far hindered research in this direction. In this work we fill precisely this gap, enabling research in dialogue policy optimisation to go from static to dynamic learning. We provide a continual learning algorithm, baseline architectures and metrics for assessing continual learning models. Moreover, we propose the dynamic dialogue policy transformer (DDPT), a novel dynamic architecture that can integrate new knowledge seamlessly, is capable of handling large state spaces and obtains significant zero-shot performance when being exposed to unseen domains, without any growth in network parameter size. We validate the strengths of DDPT in simulation with two user simulators as well as with humans.

1 Introduction

Task-oriented dialogue systems are characterised by an underlying task or a goal that needs to be achieved during the conversation, such as managing a schedule or finding and booking a restaurant. Modular dialogue systems have a tracking component that maintains information about the dialogue in a belief state, and a planning component that models the underlying policy, i.e., the selection of actions (Levin and Pieraccini, 1997; Roy et al., 2000; Williams and Young, 2007; Zhang et al., 2020b). The spectrum of what a task-oriented dialogue system can understand and talk about is defined by an ontology. The ontology defines domains such as restaurants or hotels, slots within a domain such as the area or price, and values that a slot can take, such as the area being west and the price being expensive. As dialogue systems become more popular and powerful, they should not be restricted by a static ontology. Instead, they should be dynamic and grow as the ontology grows, allowing them to comprehend new information and talk about new topics – just like humans do.

In the literature, this is referred to as continual learning (Biesialska et al., 2020; Khetarpal et al., 2020a; Hadsell et al., 2020). A learner is typically exposed to a sequence of tasks that have to be learned in a sequential order. When faced with a new task, the learner should leverage its past knowledge (*forward transfer*) and be flexible enough to rapidly learn how to solve the new task (*maintain plasticity*). On the other hand, we must ensure that the learner does not forget how to solve previous tasks while learning the new one (prevent *catastrophic forgetting*). Rather, a learner should actually improve its behaviour on previous tasks after learning a new task, if possible (*backward transfer*).

Despite progress in continual learning (Lange et al., 2019; Parisi et al., 2019; Biesialska et al., 2020; Khetarpal et al., 2020a; Hadsell et al., 2020), there is – to the best of our knowledge – no work that addresses continual reinforcement learning (continual RL) of the dialogue policy, even though the policy constitutes a key component of dialogue systems. Research in this direction is hindered by the lack of a framework that provides suitable models, evaluation metrics and training protocols.

In modular task-oriented dialogue systems the input to the dialogue policy can be modelled in many different ways (Lipton et al., 2018; Weisz et al., 2018; Takanobu et al., 2019; Wang et al., 2015; Casanueva et al., 2018; Xu et al., 2020). An appropriate choice of state representation is key

Proceedings of the 29th International Conference on Computational Linguistics, pages 266–284
October 12–17, 2022.

to the success of any form of RL (Madureira and Schlangen, 2020). In continual RL for the dialogue policy, this choice is even more essential. Different dialogue domains typically share structure and behaviour that should be reflected in the state and action representations. The architecture needs to exploit such common structure, to the benefit of any algorithm applied to the model. In this work, we therefore centre our attention on this architecture. We contribute [1]

- the first framework for continual RL to optimise the dialogue policy of a task-oriented dialogue system, two baseline architectures, an implementation of the state-of-the-art continual RL algorithm (Rolnick et al., 2018) and continual learning metrics for evaluation based on Powers et al. (2021), and
- a further, more sophisticated, new continual learning architecture based on the transformer encoder-decoder (Vaswani et al., 2017) and description embeddings, which we call dynamic dialogue policy transformer (DDPT). Our architecture can seamlessly integrate new information, has significant zero-shot performance and can cope with large state spaces that naturally arise from a growing number of domains while maintaining a fixed number of network parameters.

2 Related Work

2.1 Continual Learning in Task-oriented Dialogue Systems

Despite progress in continual learning, task-oriented dialogue systems have been barely touched by the topic. Lee (2017) proposed a task-independent neural architecture with an action selector. The action selector is a ranking model that calculates similarity between state and candidate actions. Other works concentrated on dialogue state tracking (Wu et al., 2019) or natural language generation (Mi et al., 2020; Geng et al., 2021). Geng et al. (2021) proposed a network pruning and expanding strategy for natural language generation. Madotto et al. (2021) introduced an architecture called AdapterCL and trained it in a supervised fashion for intent prediction, state tracking, generation and end-to-end learning. However, that work focused on preventing catastrophic forgetting and did not address the dialogue policy. As opposed to the above-mentioned approaches, we consider continual RL to optimise a dialogue policy.

2.2 Dialogue Policy State Representation

In the absence of works that directly address continual learning for the dialogue policy, it is worth looking at approaches that allow dialogue policy adaptation to new domains and examining them in the context of continual learning requirements.

The first group among these methods introduces new parameters to the model when the domain of operation changes. The approaches directly vectorise the belief state, hence the size of the input vector depends on the domain (as different domains for instance have different numbers of slots) (Su et al., 2016; Lipton et al., 2018; Weisz et al., 2018; Takanobu et al., 2019; Zhu et al., 2020). In the context of continual learning such approaches would likely preserve the plasticity of the underlying RL algorithm but would score poorly on forward and backward transfer.

Another group of methods utilises a hand-coded domain-independent feature set that allows the policy to be transferred to different domains (Wang et al., 2015; Casanueva et al., 2018; Chen et al., 2018; Chen et al., 2020; Lin et al., 2021). This is certainly more promising for continual learning, especially if the requirement is to keep the number of parameters bounded. However, while such models might score well on forward and backward transfer, it is possible that the plasticity of the underlying RL algorithm is degraded. Moreover, developing such features requires manual work and it is unclear if they would be adequate for any domain.

Xu et al. (2020) go a step further in that direction. They propose the usage of embeddings for domains, intents, slots and values in order to allow cross-domain transfer. To deal with the problem of a growing state space with an increased number of domains, they propose a simple averaging mechanism. However, as the number of domains becomes larger, averaging will likely result in information loss. Moreover, their architecture still largely depends on predefined feature categories.

A third option is to exploit similarities between different domains while learning about a new domain. Gašić et al. (2015) use a committee of Gaussian processes together with designed kernel functions in order to define these similarities and therefore allow domain extension and training on new

[1] https://doi.org/10.5281/zenodo.7075192

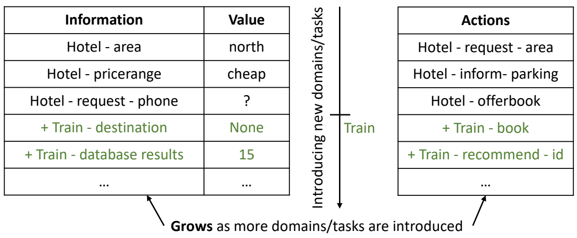


Figure 1: The amount of information that the dialogue agent must comprehend and the possible actions it can take increases as new domains/tasks are introduced.

domains. A similarity-based approach could in principle score well on all three continual learning measures. However, it is desirable to minimise the amount of manual work needed to facilitate continual learning.

2.3 Dialogue Policy Action Prediction

In the realm of domain adaptation, works assume a fixed number of actions that are slot-independent, and focus on the inclusion of slot-dependent actions when the domain changes (Wang et al., 2015; Casanueva et al., 2018; Chen et al., 2018; Chen et al., 2020; Lin et al., 2021). This allows seamless addition of new slots, but the integration of new intents or slot-independent actions requires an expansion of the model.

Works that allow new actions to be added to the action set compare the encoded state and action embeddings with each other (Lee, 2017; Xu et al., 2020; Vlasov et al., 2019), suggesting that exploiting similarities is key not only for state representations but also for action prediction.

With multi-domain dialogues it becomes necessary to be able to produce more than one action in a turn, which is why researchers started to use recurrent neural network (RNN) models to produce a sequence of actions in a single turn (Shu et al., 2019; Zhang et al., 2020a). RNNs are known however to only provide a limited context dependency.

3 Background

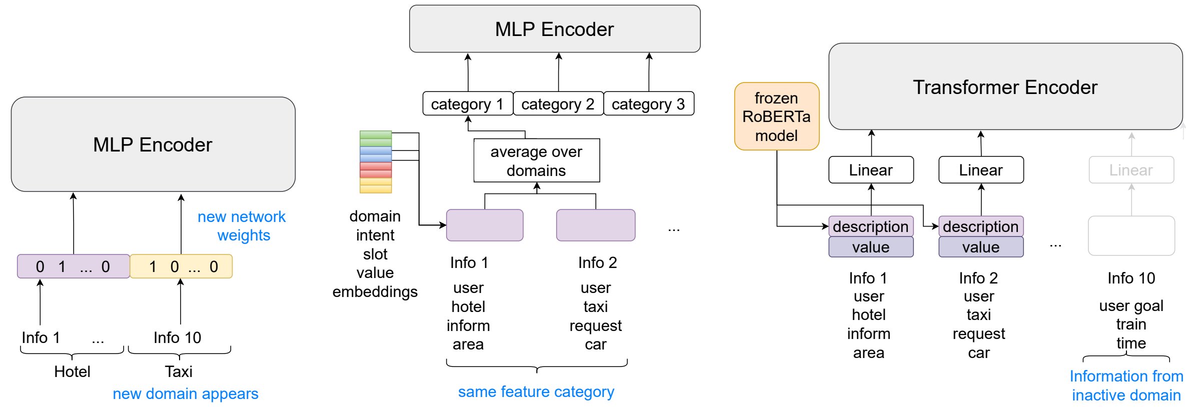
3.1 Continual Reinforcement Learning

In typical RL scenarios, an agent interacts with a stationary MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p, p_0, r \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ constitute the state and action space of the agent, $p(s'|s,a)$ models the probability of transitioning to state s' after executing action a in state s, and $p_0(s)$ is the probability of starting in state s. The reward function r defines the observed reward in every time-step. The goal is to maximise the cumulative sum of rewards in that MDP.

In contrast, continual reinforcement learning focuses on non-stationary or changing environments (Hadsell et al., 2020). Generally speaking, the agent faces a sequence of Markov decision processes $\{\mathcal{M}_z\}_{z=1}^{\infty}$ (Lecarpentier and Rachelson, 2019; Chandak et al., 2020; Khetarpal et al., 2020b) with possibly different transition dynamics, reward functions or even state or action spaces. The variable z is often referred to as a task (or context) (Caccia et al., 2020; Normandin et al., 2021). While the MDP can change from episode to episode, it is often assumed that the agent is exposed to a fixed MDP for a number of episodes and then switches to the next MDP. Once a new task (or MDP) is observed, the old task is either never observed again or only periodically (Rolnick et al., 2018; Powers et al., 2021). The goal is to retain performance on all seen tasks. This requires the model to prevent catastrophic forgetting of old tasks while at the same time adapting to new tasks.

A state-of-the art method for continual RL that uses a replay memory is CLEAR (Rolnick et al., 2018). CLEAR manages the trade-off between preventing catastrophic forgetting and fast adaptation through an on-policy update step as well as an off-policy update step. The on-policy step is supposed to adapt the policy to the recent task by using the most recent dialogues while the off-policy step should lead to retaining performance on old tasks by updating on old experiences from the replay buffer. The off-policy update is further regularized such that policy and critic outputs are close to the historical prediction. More information on CLEAR is provided in the Appendix A.1.

In the context of dialogue, a task usually refers to a domain as defined in Madotto et al. (2021) and we will use these two terms interchangeably. As an example setting, a dialogue system is tasked with fulfilling user goals concerning hotel information and booking and after some amount of time with fulfilling goals related to train bookings. In terms of MDPs, the dialogue system first faces the MDP $\mathcal{M}_{z_1}, z_1 =$ hotel, for some amount of dialogues and afterwards $\mathcal{M}_{z_2}, z_2 =$ train. Once the train domain is introduced, the state and action space grows (as a result of the growing ontology) as depicted exemplarily in Figure 1. As a consequence, the model needs to understand new topics such as



(a) Binary representation (Bin) (b) Semantic features (Sem) (c) Descriptions and values (DDPT)

Figure 2: State representation for different architectures. (a) *Bin* uses a flattened dialogue state with binary features, where the input size grows and new network weights need to be added when facing a new domain. (b) *Sem* uses the idea from Xu et al. (2020), using trainable embeddings for domain, intent, slot and value. The information corresponding to a specific feature category is then averaged over domains in order to be independent on the number of domains. (c) Our proposed DDPT model uses descriptions for every information which are embedded using a pretrained language model. The embedded description together with a value for the information is then fed into a linear layer and a transformer encoder.

the destination of the train and select new actions such as booking a train. In addition, the probability distributions p and p_0 of $\mathcal{M}_{z_2}$ are different compared to $\mathcal{M}_{z_1}$ since the probability that the user talks about hotels should be close to 0 while the probability that the agent's states contain information related to trains is close to 1.0.

3.2 Dialogue Policy in Modular Systems

In modular task-oriented dialogue systems, the decision of a dialogue policy is commonly based on the hidden information state of the dialogue system. This hidden information state, according to Young et al. (2007), should consist of the following information: the predicted user action, the predicted user goal and a representation of the dialogue history. For reactive behaviour by the policy, the user action is important as it includes information related to requests made by the user. The predicted user goal summarises the current goal of the user, including specified constraints. Lastly, the dialogue history representation captures the relevant information mentioned in the dialogue history, such as the latest system action. The state can also include the likelihood of the predicted acts, goal and dialogue history in the form of confidence scores. Moreover, the state often contains information about the database, for instance the number of entities that are available given the current predicted user goal.

Each domain that the system can talk about is either active, meaning that it has already been mentioned by the user, or inactive. The active domains can be derived from the user acts, from the user goal or tracked directly (van Niekerk et al., 2021).

Finally, the policy is supposed to take actions. As in (Shu et al., 2019; Zhang et al., 2020a), each action can be represented as a sequence of tuples $(domain, intent, slot)$. For instance, an action could be that the system requests the desired arrival time of a train or asks for executing a payment.

4 Dynamic Dialogue Policy Transformer

Our goal is to build a model that can talk about a potentially very large number of domains and is able to deal with new domains and domain extensions seamlessly without requiring any architectural changes. In particular, the number of model parameters should remain fixed. This is challenging since new domains require understanding of previously unseen information and the ability to talk about new topics.

Our approach is inspired by the way an employee would explain and act upon a novel task: 1) describe the information that can be used and the actions that can be taken in natural language, 2) restrict the focus to the information that is important for solving the task at hand, 3) when an action needs to be taken, this action is based on the information that was attended to (e.g. for the action to request the area, one would put attention on the information whether the area is already given). We propose an architecture that uses the transformer encoder with information embeddings (Section 4.1 and Figure 2(c)) to fulfill 1) and 2) and the transformer decoder that leverages the domain gate (Section 4.2, 4.3 and Figure 3) to fulfill 3), which we call *dynamic dialogue policy transformer* (DDPT).

4.1 State Representation

Recall from Section 3.2 that the agent is provided with information on various concepts f for domain d_f: the user goal (domain-slot pairs), the user action (intents) and the dialogue history (system intents and database results). We assume that the agent has access to an external dictionary providing a natural language description $descr_f$ of each of these, e.g. “area of the hotel” or “number of hotel database results”, which is common in dialogue state tracking (Rastogi et al., 2020; van Niekerk et al., 2021; Lee et al., 2021). See Appendix A.5 for the full list of descriptions. During a dialogue, the dialogue state or belief tracker assigns numerical values v_f, e.g. confidence scores for user goals or the number of data base results, etc. For every concept f we define the information embedding

$$\boldsymbol{e}_{\text{info}_f} = \text{Lin}\left(\left[\overline{\text{LM}}(descr_f), \text{Lin}(v_f)\right]\right) \in \mathbb{R}^h$$

where $\overline{\text{LM}}$ denotes applying a language model such as RoBERTa (Liu et al., 2019) and averaging of the token embeddings, and Lin denotes a linear layer. $\boldsymbol{e}_{\text{info}_f}$ represents information in a high-dimensional vector space. Intuitively, every information can be thought of as a node in a graph. The list of information embeddings are the input to a transformer encoder (Vaswani et al., 2017). The attention mechanism allows the agent to decide for every information embedding $\boldsymbol{e}_{\text{info}_f}$ on which other embeddings $\boldsymbol{e}_{\text{info}_g}$ it can put its attention. With a growing number of domains that the system can talk about, the number of information embeddings will increase, making it more difficult to handle the growing state space. However, we observe that only information that is related to active domains is important at the current point in time. Therefore, we prohibit the information embeddings from attending to information that is related to inactive domains in order to avoid the issue of growing state spaces. While the actual state space may be extremely large due to hundreds of domains, the effective state space remains small, making it possible to handle a very large number of domains. Our proposed state encoder is depicted in Figure 2(c).

In this way, the state representation meets the

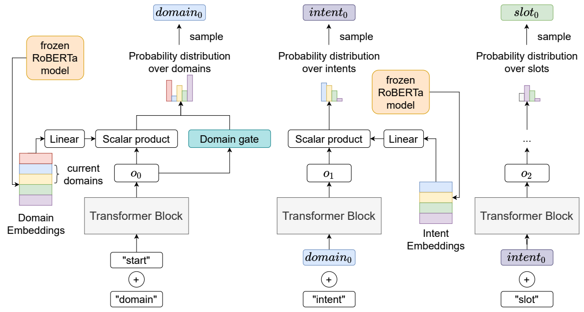


Figure 3: Proposed action prediction in DDPT using a transformer decoder. In every decoding step, a token embedding for domain, intent or slot informs the model what needs to be predicted and the previous output is fed into the decoder. In case of domain prediction, we propose a domain gate that decides whether to choose a domain that the user currently talks about.

following demands: 1) new concepts can be understood and incorporated seamlessly into the state without a growth in network parameters, as long as they are descriptive; 2) the description embeddings from a language model allow forward transfer by exploiting similarities and common structure among tasks; 3) the value v_f allows numerical information such as confidence scores or other measures of model uncertainty to be included; 4) the state space will not be unreasonably large as information for inactive domains is masked.

4.2 Action Prediction

Similar to existing work (Shu et al., 2019; Zhang et al., 2020a) we separately predict domains, intents and slots for action prediction. We define a domain set $\mathcal{D}$, intent set $\mathcal{I}$ and slot set $\mathcal{S}$ as follows. The domain set $\mathcal{D}$ consists of all domains the model has seen so far plus an additional *stop* domain. The intent set $\mathcal{I}$ and slot set $\mathcal{S}$ consist of all intents and slots we can use for actions, respectively. Every domain, intent and slot has an embedding vector, which we obtain by feeding the token of the domain, intent or slot into our pretrained language model. The embedding vectors are then fed into a linear layer that produces vectors of size $\mathbb{R}^h$. We thus obtain domain, intent and slot embeddings $\boldsymbol{b}^d\ \forall d \in \mathcal{D}$, $\boldsymbol{b}^i\ \forall i \in \mathcal{I}$, and $\boldsymbol{b}^s\ \forall s \in \mathcal{S}$.

The policy first chooses a domain. Then, based on the domain, it picks an intent from the list of intents that are possible for that domain. Lastly, it picks an adequate slot from the set of possible slots for that domain and intent. This process repeats until the policy selects the *stop* domain. This will lead to a sequence $(domain_m, intent_m, slot_m)_{m=0}^n$. We leverage a transformer decoder (Vaswani et al., 2017), the aforementioned embeddings for domains, intents and slots and similarity matching to produce the sequence. In every decoding step t the input to the transformer is $\boldsymbol{b}_{t-1} + \boldsymbol{l}_t$, where $\boldsymbol{b}_{t-1}$ is the embedding of the previous prediction and $\boldsymbol{l}_t$ is a token embedding for token *domain*, *intent* or *slot* that indicates what needs to be predicted in turn t. $\boldsymbol{b}_{-1}$ is an embedding of a *start* token.

If we need to predict a domain in step t, we calculate the scalar product between the decoder output vector $\boldsymbol{o}_t$ and the different domain embeddings $\boldsymbol{b}^d$ and apply the softmax function to obtain a probability distribution $\mathrm{softmax}[\boldsymbol{o}_t \odot \boldsymbol{b}^d, d \in \mathcal{D}]$ over domains from which we can sample. Intent and slot prediction is analogous. In order to guarantee exploration during training and variability during evaluation, we sample from the distributions. While it is important to explore domains during training, during evaluation the domain to choose should be clear. We hence take the domain with the highest probability during evaluation.

As in the state representation, the embeddings using a pretrained language model allow understanding of new concepts (such as a new intent) immediately, which facilitates zero-shot performance. We do not fine-tune any embedding that is produced by the language model.

4.3 Domain Gate

If the policy is exposed to a new unseen domain, the most important point to obtain any zero-shot performance is that the policy predicts the correct domain to talk about. If we only use similarity matching of domain embeddings, the policy will likely predict domains it already knows. In dialogue state tracking we often observe that similarity matching approaches predict values they already know when faced with new unseen values, which leads to poor zero-shot generalisation (Rastogi et al., 2018). To circumvent that, we propose the domain gate. Let $\mathcal{D}_{\text{curr}}$ be the set of domains that the user talks about in the current turn. In every decoding step t where a domain needs to be predicted, the domain gate obtains $\boldsymbol{o}_t$ as input and predicts the probability p_{curr} of using a domain from $\mathcal{D}_{\text{curr}}$. When the policy needs to predict a domain in step t, it now uses the probability distribution given by $p_{\text{curr}} \cdot \mathrm{softmax}[\boldsymbol{o}_t \odot \boldsymbol{b}_d, d \in \mathcal{D}_{\text{curr}}] + (1 - p_{\text{curr}}) \cdot \mathrm{softmax}[\boldsymbol{o}_t \odot \boldsymbol{b}_d, d \notin \mathcal{D}_{\text{curr}}]$.

In this process, the policy does not have to predict the new domain immediately but can abstractly first decide whether it wants to use a domain that the user talks about at the moment. The decoding process is depicted in Figure 3.

5 Experimental Setup

5.1 Metrics

We follow the setup recently proposed by Powers et al. (2021), which assumes that our N tasks/domains $z_1, ..., z_N$ are represented sequentially and each task z_i is assigned a budget k_{z_i}. We can cycle through the tasks M times, leading to a sequence of tasks $x_1, ..., x_{N \cdot M}$. The cycling over tasks defines a more realistic setting than only seeing a task once in the agent's lifetime, in particular in dialogue systems where new domains are introduced but rarely removed.

Continual evaluation: We evaluate performance on all tasks periodically during training. We show the performance for every domain separately to have an in-depth evaluation and the average performance over domains for an overall trend whether the approaches continually improve.

Forgetting: We follow the definition proposed by Chaudhry et al. (2018) and Powers et al. (2021). Let $m_{i,k}$ be a metric achieved on task z_i after training on task x_k, such as the average return or the average dialogue success. For seeds s, tasks z_i and x_j, where $i < j$, we define

$$\mathcal{F}_{i,j} = \frac{1}{s} \sum_s \max_{k \in [0, j-1]} \{m_{i,k} - m_{i,j}\}. \quad (1)$$

$\mathcal{F}_{i,j}$ compares the maximum performance achieved on task z_i before training on task x_j to the performance for z_i after training on task x_j. If $\mathcal{F}_{i,j}$ is positive, the agent has become worse at past task z_i after training on task x_j, indicating forgetting. When $\mathcal{F}_{i,j}$ is negative, the agent has become better at task z_i, indicating backward transfer. We define $\mathcal{F}_i$ as the average over the $\mathcal{F}_{i,j}$ and $\mathcal{F}$ as the average over $\mathcal{F}_i$.

(Zero-Shot) Forward transfer: For seeds s, tasks z_i and z_j, where $j < i$, we define

$$\mathcal{Z}_{i,j} = \frac{1}{s} \sum_s m_{i,j}. \quad (2)$$

We do not substract initial performance as in Powers et al. (2021) as we are interested in the absolute performance telling us how well we do on task z_i after training on a task z_j. We define $\mathcal{Z}_i$ as the average over the $\mathcal{Z}_{i,j}$ and $\mathcal{Z}$ as the average over $\mathcal{Z}_i$.

5.2 Baselines

We implemented two baselines in order to compare against our proposed DDPT architecture. We do not include a baseline based on expert-defined domain-independent features (Wang et al., 2015) as this requires a significant amount of hand-coding and suffers from scalability issues.

5.2.1 Baseline State Representations

We will abbreviate the following baselines with **Bin** and **Sem** that indicate their characteristic way of state representation.

Bin: The first baseline uses a flattened dialogue state for the state representation with *binary* values for every information which is the most common way (Takanobu et al., 2019; Zhu et al., 2020; Weisz et al., 2018). If a new domain d appears, the input vector must be enlarged in order to incorporate the information from d and new network parameters need to be initialised. The state encoding can be seen in Figure 2(a). This baseline serves as a representative of methods where new domains necessitate additional parameters.

Sem: The second baseline implements the idea from Xu et al. (2020), which uses trainable embeddings for domains, intents, slots and values that can capture *semantic* meaning and allow cross-domain transfer. Using trainable embeddings, one representation is calculated for every feature in every feature category (such as user-act, user goal, etc.) in every domain. The feature representations in a category are then averaged over domains to obtain a final representation. More information can be found in Appendix A.4. This baseline serves as a representative of methods where feature representations remain fixed.

5.2.2 Action Prediction for Baselines

Unlike DDPT, which uses a transformer for action prediction, the baselines Bin and Sem use an RNN model for action prediction (Shu et al., 2019; Zhang et al., 2020a). This model uses the decoding process explained in Section 4.2 with the exception that the baselines use trainable embeddings for domain, intent and slot (randomly initialised) instead of using embeddings from a pretrained language model as DDPT does. Moreover, they do not use the proposed domain gate.

5.3 Setup

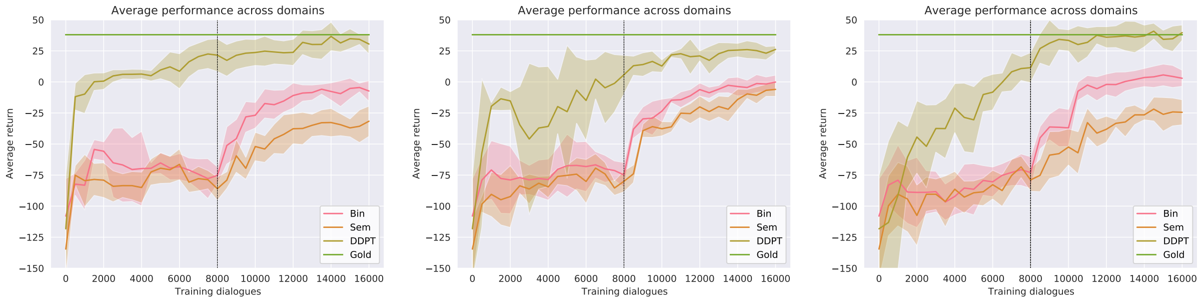
We use ConvLab-2 (Zhu et al., 2020) as the backbone of our implementation. We take five different tasks from the MultiWOZ dataset (Budzianowski et al., 2018) which are hotel, restaurant, train, taxi and attraction. Hotel, restaurant and train are more difficult compared to attraction and taxi as they require the agent to do bookings in addition to providing information about requested slots. We exclude police and hospital from the task list as they are trivial. We use the rule-based dialogue state tracker and the rule-based user simulator provided in ConvLab-2 (Zhu et al., 2020) to conduct our experiments. Typically, the reward provided is -1 in every turn to encourage efficiency, and a reward of 80 or -40 for dialogue success or failure. A dialogue is successful if the system provided the requested information to the user and booked the correct entities (if possible). We stick to the above reward formulation with one exception: Instead of the turn level reward of -1, we propose to use information overload (Roetzel, 2019). The reason is that dialogue policies tend to over-generate actions, especially if they are trained from scratch. While the user simulator ignores the unnecessary actions, real humans do not. We define information overload for an action $(domain_m, intent_m, slot_m)_{m=1}^n$ as $r_{io} = -\rho \cdot n$, where $\rho \in \mathbb{N}$ defines the degree of the penalty. Information overload generalizes the reward of -1 in single action scenarios. We use $\rho = 3$ in the experiments.

We train each of the three architectures using CLEAR (Rolnick et al., 2018). We set the replay buffer capacity to 5000 dialogues and use reservoir sampling (Isele and Cosgun, 2018) when the buffer is full. We assign a budget of 2000 dialogues to restaurant, hotel and train and 1000 to attraction and taxi and cycle through these tasks two times, resulting in 16000 training dialogues in total. Since task ordering is still an open area of research (Jiang et al., 2020), we test three different permutations so that our results do not depend on a specific order. The domain orders we use are 1) *easy-to-hard*: attraction, taxi, train, restaurant, hotel 2) *hard-to-easy*: hotel, restaurant, train, taxi, attraction and 3) *mixed*: restaurant, attraction, hotel, taxi, train.

6 Results

6.1 Continual Evaluation

We show performance in terms of average return for all three task orders in Figure 4(a)-(c). The plots



(a) easy-to-hard (b) hard-to-easy (c) mixed

Figure 4: Training Bin, Sem and DDPT (ours) using CLEAR on three different domain orders, each with 5 different seeds, by interacting with the rule-based user simulator. Each model is evaluated every 500 training dialogues on 100 dialogues per domain. The plots show the average return, where performance is averaged over domains. The vertical line at 8000 dialogues indicates the start of cycle 2. The shaded area represents standard deviation. Gold serves as an upper bound.

show the performance averaged over domains. We refer to Appendix A.8 for in-depth evaluations for each individual domain. The horizontal line **Gold** denotes an upper limit for the models that was obtained by training a Bin model separately on each domain until convergence. We can observe that DDPT outperforms the baselines regardless of task order, almost reaching the upper bound. We will see in Section 6.2 that the baselines suffer more from forgetting compared to DDPT, such that training on a new domain reduces performance on previous domains. We suspect that this contributes to the lower final performance of the baselines. Moreover, we can observe that the final performance of DDPT barely depends on a specific task order. Nevertheless, we can see that training starts off faster in easy-to-hard order, which shows that behaviour learned for attraction transfers well to other domains. Lastly, the second training cycle is necessary for increasing performance of the models. We note that even though it looks like the baselines don't learn at all in the first round, they do learn but tend to forget previous knowledge. This can be observed in detail in Appendix A.8.

6.2 Forward Transfer and Forgetting

We calculated forward and forgetting metrics as explained in Section 5.1. Table 1 shows success rates instead of average return because success is easier to interpret. We can see for every model the summary statistics $\mathcal{F}$ and $\mathcal{Z}$ measuring average forgetting and forward transfer, respectively. To obtain lower bounds we added forward and forgetting of a random model that is initialised randomly again every time it observes a domain.

Table 1 reveals that DDPT outperforms the baselines significantly in terms of absolute numbers and also relative numbers compared to the random performance. As expected, Bin shows almost no zero-shot performance improvement compared to the random model, whereas Sem obtains slight improvement. DDPT shows large forward transfer capabilities and strong robustness against forgetting. We attribute this to the frozen description and action embeddings stemming from the language model and the domain gate. The language model allows us to interpret new information and actions immediately, enabling the model to draw connections between learned tasks and new ones. At the same time, frozen embeddings are robust to forgetting. The domain gate allows the model to choose the domain more abstractly without initial exploration due to the decision between current or non-current domains, which facilitates zero-shot performance. Moreover, the baselines need to make a hard decision between domains (balancing between choosing a domain we learn about at the moment and old domains), whereas the domain decision for DDPT is abstracted through the domain gate, leading to robustness against forgetting. Both baselines perform substantially better than the lower bound, suggesting that these are non-trivial baselines.

6.3 Benefits of Domain Gate

In order to analyse the contribution of the domain gate to the forward capabilities of DDPT, we train a DDPT model without domain gate on the easy-to-hard order, where DDPT showed the highest forward transfer. From Table 2 we can observe that performance drops significantly for all domains if

	Easy-to-hard		Hard-to-easy		Mixed order		Random	
Model	$\mathcal{F}\downarrow$	$\mathcal{Z}\uparrow$	$\mathcal{F}\downarrow$	$\mathcal{Z}\uparrow$	$\mathcal{F}\downarrow$	$\mathcal{Z}\uparrow$	$\mathcal{F}\downarrow$	$\mathcal{Z}\uparrow$
Bin	0.14	0.39	0.14	0.45	0.14	0.38	0.43	0.39
Sem	0.20	0.39	0.17	0.37	0.18	0.29	0.43	0.26
DDPT	**0.01**	**0.73**	**0.02**	**0.68**	**0.03**	**0.57**	0.43	0.34

Table 1: Showing summary statistics in terms of success for forgetting $\mathcal{F}$ (ranging between -1 and 1, the lower the better) and forward transfer $\mathcal{Z}$ (ranging between 0 and 1, the higher the better).

	Taxi	Train	Restaurant	Hotel	$\mathcal{Z}\uparrow$
DDPT	0.90	0.76	0.73	0.53	0.73
DDPT w/o domain gate	0.68	0.19	0.57	0.28	0.43

Table 2: Forward transfer metrics $\mathcal{Z}_i$ in terms of success for different domains i trained on easy-to-hard order with and without domain gate.

the domain gate is not employed, which shows the importance of this mechanism.

6.4 Results on Transformer-based Simulator

In order to strengthen our results and show that they do not depend on the simulator used, we conducted an additional experiment using the transformer-based user simulator TUS (Lin et al., 2021). We only show results for the mixed order, having in mind that results have not been dependent on the domain order used. Figure 5 shows that DDPT again outperforms the baseline.

6.5 Results on Human Trial

We further validate the results by conducting a human trial. We compare Bin, Gold and DDPT, where Bin and DDPT were trained on the mixed domain order. We hire humans through Amazon Mechanical Turk and let them directly interact with our systems, thereby collecting 258, 278 and 296 dialogues for Bin, Gold and DDPT, respectively. After a user finished the dialogue we asked 1) whether the dialogue was successful (Success), 2) whether the system often mentioned something the user did not ask for such as a wrong domain (UnnecInfo) 3), whether the system gave too much information (TooMuchInfo) and 4) about the general performance (Performance). Table 3 shows that the upper bound Gold and DDPT perform equally well ($p > 0.05$) in every metric whereas Bin performs statistically significant worse. The low performance of Bin can be partially attributed to frequently choosing a wrong domain that humans are more sensitive to than a user simulator.

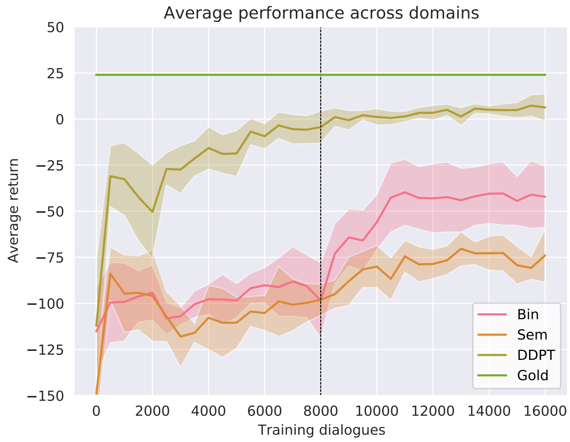


Figure 5: Training Bin, Sem and DDPT (ours) on the mixed domain order with the transformer based user simulator TUS.

	Success $\uparrow$	UnnecInfo $\downarrow$	TooMuchInfo $\downarrow$	Performance $\uparrow$
Bin	0.45	3.98	3.15	2.45
Gold	0.81	2.79	2.71	3.65
DDPT	0.77	2.75	2.56	3.67

Table 3: Human trial results where Bin, Gold and DDPT interacted with real users. There is no statistically significant difference (p > 0.05) between DDPT and Gold, while Bin is statistically significantly worse (p < 0.05) than Gold and DDPT.

Example dialogues are given in Appendix A.6.

7 Conclusion

In this work we provided an algorithm, baseline models and evaluation metrics to enable continual RL for dialogue policy optimisation. Moreover, we proposed a dynamic dialogue policy model called DDPT that builds on information descriptions, a pretrained language model and the transformer encoder-decoder architecture. It integrates new information seamlessly as long as it is descriptive, and obtains significant zero-shot performance on unseen domains while being robust to forgetting. The strengths of DDPT were validated in simulation with two simulators as well as humans. This opens the door for building evolving dialogue systems, that continually expand their knowledge and improve their behaviour throughout their lifetime.

Acknowledgements

This work is a part of DYMO project which has received funding from the European Research Council (ERC) provided under the Horizon 2020 research and innovation programme (Grant agreement No. STG2018 804636). N. Lubis, C. van Niekerk, M. Heck and S. Feng are funded by an Alexander von Humboldt Sofja Kovalevskaja Award endowed by the German Federal Ministry of Education and Research. Computing resources were provided by Google Cloud and HHU ZIM.

References

Magdalena Biesialska, Katarzyna Biesialska, and Marta R. Costa-jussà. 2020. Continual lifelong learning in natural language processing: A survey. *CoRR*, abs/2012.09823.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Massimo Caccia, Pau Rodríguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Caccia, Issam H. Laradji, Irina Rish, Alexandre Lacoste, David Vázquez, and Laurent Charlin. 2020. Online fast adaptation and knowledge accumulation: a new approach to continual learning. *CoRR*, abs/2003.05856.

Iñigo Casanueva, Paweł Budzianowski, Pei-Hao Su, Stefan Ultes, Lina M. Rojas-Barahona, Bo-Hsiang Tseng, and Milica Gašić. 2018. 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, New Orleans, Louisiana. Association for Computational Linguistics.

Yash Chandak, Georgios Theocharous, Shiv Shankar, Martha White, Sridhar Mahadevan, and Philip S. Thomas. 2020. Optimizing for the future in non-stationary mdps. *CoRR*, abs/2005.08158.

Arslan Chaudhry, Puneet Kumar Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. 2018. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XI*, volume 11215 of *Lecture Notes in Computer Science*, pages 556–572. Springer.

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, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

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

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. 2018. IMPALA: scalable distributed Deep-RL with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1406–1415. PMLR.

Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Policy committee for adaptation in multi-domain spoken dialogue systems. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 806–812.

Binzong Geng, Fajie Yuan, Qiancheng Xu, Ying Shen, Ruifeng Xu, and Min Yang. 2021. Continual learning for task-oriented dialogue system with iterative network pruning, expanding and masking. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 517–523, Online. Association for Computational Linguistics.

Raia Hadsell, Dushyant Rao, Andrei Rusu, and Razvan Pascanu. 2020. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24:1028–1040.

David Isele and Akansel Cosgun. 2018. Selective experience replay for lifelong learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. 2020. Prioritized level replay. *CoRR*, abs/2010.03934.

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. 2020a. Towards continual reinforcement learning: A review and perspectives. *CoRR*, abs/2012.13490.

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. 2020b. Towards continual reinforcement learning: A review and perspectives. *CoRR*, abs/2012.13490.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory G. Slabaugh, and Tinne Tuytelaars. 2019. Continual learning: A comparative study on how to defy forgetting in classification tasks. *CoRR*, abs/1909.08383.

Erwan Lecarpentier and Emmanuel Rachelson. 2019. Non-stationary markov decision processes, a worst-case approach using model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. 2021. Dialogue state tracking with a language model using schema-driven prompting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4937–4949, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kyusong Lee, Tiancheng Zhao, Alan W. Black, and Maxine Eskenazi. 2018. DialCrowd: A toolkit for easy dialog system assessment. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 245–248, Melbourne, Australia. Association for Computational Linguistics.

Sungjin Lee. 2017. Toward continual learning for conversational agents. *CoRR*, abs/1712.09943.

Esther Levin and Roberto Pieraccini. 1997. A stochastic model of computer-human interaction for learning dialogue strategies. In *EUROSPEECH 97*, pages 1883–1886.

Hsien-chin Lin, Nurul Lubis, Songbo Hu, Carel van Niekerk, Christian Geishauser, Michael Heck, Shutong Feng, and Milica Gašić. 2021. Domain-independent user simulation with transformers for task-oriented dialogue systems. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 445–456, Singapore and Online. Association for Computational Linguistics.

Zachary Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. 2018. BBQ-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul Crook, Bing Liu, Zhou Yu, Eunjoon Cho, Pascale Fung, and Zhiguang Wang. 2021. Continual learning in task-oriented dialogue systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7452–7467, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Brielen Madureira and David Schlangen. 2020. An overview of natural language state representation for reinforcement learning. *ArXiv*, abs/2007.09774.

Fei Mi, Liangwei Chen, Mengjie Zhao, Minlie Huang, and Boi Faltings. 2020. Continual learning for natural language generation in task-oriented dialog systems. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3461–3474, Online. Association for Computational Linguistics.

Fabrice Normandin, Florian Golemo, Oleksiy Ostapenko, Pau Rodríguez, Matthew D. Riemer, Julio Hurtado, Khimya Khetarpal, Dominic Zhao, Ryan Lindeborg, Timothée Lesort, Laurent Charlin, Irina Rish, and Massimo Caccia. 2021. Sequoia: A software framework to unify continual learning research. *CoRR*, abs/2108.01005.

German Parisi, Ronald Kemker, Jose Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020. Few-shot natural language generation for task-oriented dialog. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 172–182, Online. Association for Computational Linguistics.

Sam Powers, Eliot Xing, Eric Kolve, Roozbeh Mottaghi, and Abhinav Gupta. 2021. CORA: Benchmarks, baselines, and metrics as a platform for continual reinforcement learning agents.

Abhinav Rastogi, Raghav Gupta, and Dilek Hakkani-Tur. 2018. Multi-task learning for joint language understanding and dialogue state tracking. *arXiv preprint arXiv:1811.05408*.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8689–8696.

Peter Gordon Roetzel. 2019. Information overload in the information age: a review of the literature from business administration, business psychology, and related disciplines with a bibliometric approach and framework development. *Business Research*, 12(2):479–522.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. 2018. Experience replay for continual learning. *CoRR*, abs/1811.11682.

Nicholas Roy, Joelle Pineau, and Sebastian Thrun. 2000. Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, page 93–100, USA. Association for Computational Linguistics.

Lei Shu, Hu Xu, Bing Liu, and Piero Molino. 2019. Modeling multi-action policy for task-oriented dialogues. *CoRR*, abs/1908.11546.

Pei-Hao Su, Milica Gašić, Nikola Mrkšić, Lina Maria Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2016. Continuously learning neural dialogue management. *CoRR*, abs/1606.02689.

Ryuichi Takanobu, Hanlin Zhu, and Minlie Huang. 2019. Guided dialog policy learning: Reward estimation for multi-domain task-oriented dialog. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 100–110, Hong Kong, China. Association for Computational Linguistics.

Carel van Niekerk, Andrey Malinin, Christian Geishauser, Michael Heck, Hsien-chin Lin, Nurul Lubis, Shutong Feng, and Milica Gašić. 2021. Uncertainty measures in neural belief tracking and the effects on dialogue policy performance. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7901–7914, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Vladimir Vlasov, Johannes E. M. Mosig, and Alan Nichol. 2019. Dialogue transformers. *CoRR*, abs/1910.00486.

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, Prague, Czech Republic. Association for Computational Linguistics.

Gellért Weisz, Paweł Budzianowski, Pei-Hao Su, and Milica Gašić. 2018. Sample efficient deep reinforcement learning for dialogue systems with large action spaces. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(11):2083–2097.

Jason D. Williams and Steve Young. 2007. Partially observable Markov decision processes for spoken dialog systems. *Comput. Speech Lang.*, 21(2):393–422.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 808–819, Florence, Italy. Association for Computational Linguistics.

Yumo Xu, Chenguang Zhu, Baolin Peng, and Michael Zeng. 2020. Meta dialogue policy learning. *CoRR*, abs/2006.02588.

Steve Young, Jost Schatzmann, Karl Weilhammer, and Hui Ye. 2007. The hidden information state approach to dialog management. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, volume 4, pages IV–149–IV–152.

Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020a. Task-oriented dialog systems that consider multiple appropriate responses under the same context. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9604–9611. AAAI Press.

Zheng Zhang, Ryuichi Takanobu, Minlie Huang, and Xiaoyan Zhu. 2020b. Recent advances and challenges in task-oriented dialog system. *CoRR*, abs/2003.07490.

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, Online. Association for Computational Linguistics.

A Appendix

A.1 Background on CLEAR

A.1.1 VTRACE Algorithm

VTRACE (Espeholt et al., 2018) is an off-policy actor critic algorithm. As such, it optimizes both a policy π_θ and a corresponding critic V_ψ that estimates the state-value function V of π_θ. Actor and critic are both updated using experience from a replay buffer $\mathcal{B}$.

Given a trajectory $\tau = (s_t, a_t, r_t)_{t=k}^{t=k+n}$ generated by a behaviour policy μ, the n-steps vtrace-target for $V(s_k)$ is defined as

$$v_k = V(s_k) + \sum_{t=k}^{k+n-1} \gamma^{t-k} (\prod_{i=k}^{t-1} c_i) \delta_t V,$$

where $\delta_t V = \rho_t (r_t + \gamma V(s_{t+1}) - V(s_t))$ is a temporal difference for V, and $\rho_t = \min(\overline{\rho}, \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)})$ and $c_i = \min(\overline{c}, \frac{\pi(a_i|s_i)}{\mu(a_i|s_i)})$ are truncated importance sampling weights. The scalars $\overline{\rho}$ and $\overline{c}$ are hyperparameters where it is assumed that $\overline{\rho} \geq \overline{c}$.

The critic function is then optimized to minimize the gap between its prediction and the vtrace-target:

$$\mathcal{L}_{critic}(\psi) = \mathbb{E}_{\tau \sim \mathcal{B}}[(v_k - V_\psi(s_k))^2] \quad (3)$$

The actor is optimized using the following off-policy policy gradient:

$$\mathbb{E}_{\tau \sim \mathcal{B}}[\frac{\pi(a_k|s_k)}{\mu(a_k|s_k)} A_k \nabla_\theta \log \pi_\theta(a_k|s_k)] \quad (4)$$

where $A_k = (r_k + \gamma v_{k+1} - V_\psi(s_k))$ is an estimate of the advantage function. To prevent premature convergence, they add an entropy loss $\mathcal{L}_{entropy}(\theta)$ during optimization.

A.1.2 CLEAR

CLEAR is a continual learning algorithm that adapts VTRACE to fulfill the continual learning requirements. The goal is to obtain fast adaptation capabilities as well as preventing catastrophic forgetting. Fast adaptation is tackled by using the most recent trajectories instead of randomly sampling from the buffer $\mathcal{B}$ in Equations 3 and 4.

In order to prevent catastrophic forgetting, they sample non-recent experience from the replay buffer and update policy and critic using Equations 3 and 4. To further regularize these non-recent updates, they introduce regularization losses $\mathcal{L}_{\pi-reg}$ and $\mathcal{L}_{v-reg}$. $\mathcal{L}_{v-reg}$ forces the critic prediction to be close to the historic prediction through a mean-squared error loss. $\mathcal{L}_{\pi-reg}$ regularizes the actor to minimize the KL-divergence between the behaviour policy μ and current policy π_θ:

$$\mathcal{L}_{v-reg}(\psi) = \mathbb{E}_{\tau \sim \mathcal{B}}[(V_\psi(s_k) - V_{replay}(s_k))^2]$$
$$\mathcal{L}_{\pi-reg}(\theta) = \mathbb{E}_{\tau \sim \mathcal{B}}[\sum_a \mu(a_| s_k) \log \frac{\mu(a_| s_k)}{\pi_\theta(a_| s_k)}]$$

An online-offline ratio determines how many recent and non-recent experience is used in an update, thereby trading-off fast adaptation and catastrophic forgetting prevention.

A.2 Training details

For the baselines, the MLP encoder uses a 3-layer MLP with hidden dimension of 128 and RELU as activation function. We use a GRU with 2 layers and input size as well as hidden size of 128 for action decoding. The domain, intent and slot embeddings for action prediction have a size of 64. They are fed through a linear layer that projects it to a vector of size 128 (same size as GRU output) in order to allow computation of the scalar product with the GRU output. The semantic encoding in Sem uses an embedding size of 32 for domain, intent, slot and values. The critic for Bin and Sem has the same architecture as the MLP encoder, with an additional linear layer to project the output to a real valued number.

For the DDPT model, we use an input size and hidden size of 128 in both transformer encoder and decoder. We use two heads for the encoder and decoder, 4 transformer layers for the encoder and 2 for the decoder. The critic for DDPT has the same architecture as the transformer encoder, obtaining the same input as the policy module plus an additional CLS vector (as in RoBERTa). The output of the CLS vector is fed into a linear layer to obtain the critic prediction.

For every model, we use the same training configurations. We use the ADAM optimiser (Kingma and Ba, 2015) with a learning rate of 5e-5 and 1e-4 for policy and critic module, respectively. We sample a batch of 64 episodes for updating the model after every 2 new dialogues. The replay buffer size is set to 5000. For the VTRACE algorithm, the parameters $\bar{\rho}$ and $\bar{c}$ are set to 1.0. For CLEAR we use an online-offline ratio of 0.2, i.e. 20% of the dialogues in a batch are from the most recent dialogues and the remaining 80% from historical dialogues. The regularization losses are weighted by 0.1 and the entropy loss by 0.01.

We used a NVIDIA Tesla T4 provided by the Google Cloud Platform for training the models. The training of one model took 10 to 16 hours depending on the architecture used.

A.3 Masking of illegal actions

To aid the policy in the difficult RL environment, we add a simple masking mechanism that prohibits illegal actions. The action masking includes the following

- If the data base query tells us that entities for a domain are available, the policy is not allowed to say that there are no entities available.
- If there is no entity found with the current constraints, the policy is not allowed to inform on information about entities.
- The *Booking* domain is only usable for hotel and restaurant.

A.4 Baselines

As mentioned in Section 5.2, the second baseline incorporates the idea from Xu et al. (2020), which uses trainable embeddings for domains, intents and slots to allow cross-domain transfer. For every feature category (such as user-act, user goal, etc.) and every domain, it calculates for every feature in that category a representation using trainable domain, intent and slot embeddings. The features in a category are then averaged over domains to obtain a final representation.

For instance, considering the user-act category for a domain d, the user act $(d, i_k, s_k)_{k=0}^{n}$ is first embedded as $\hat{\boldsymbol{s}}_{\text{u-act},d} = \frac{1}{n}\sum_{k=0}^{n}[\boldsymbol{v}_d, \boldsymbol{v}_{i_k}, \boldsymbol{v}_{s_k}]$, where $\boldsymbol{v}_d, \boldsymbol{v}_{i_k}$ and $\boldsymbol{v}_{s_k}$ are trainable embeddings for domain d, intents i_k and slots s_k and afterwards fed through a residual block, leading to $\boldsymbol{s}_{\text{u-act},d} = \hat{\boldsymbol{s}}_{\text{u-act},d} + \text{ReLU}(\boldsymbol{W}_{\text{u-act}}\hat{\boldsymbol{s}}_{\text{u-act},d} + \boldsymbol{b}_{\text{u-act}})$. If there is no user-act for domain d, we use an embedding for *no-user-act* to indicate that. The overall feature representation for the user-act is then given by $\boldsymbol{s}_{\text{u-act}} = \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} \boldsymbol{s}_{\text{u-act},d}$.

The representations for different feature categories are then concatenated and fed into a multi-layer perceptron encoder. The state encoding can be seen in Figure 2(b). We abbreviate this baselines as *Sem* as it uses semantic features.

A.5 Descriptions

Our DDPT model uses descriptions for every possible information. This allows us to seamlessly deal with new information we have not seen before yet by leveraging a pretrained language model. The language model provides us token embeddings for the description, which are averaged in order to obtain the description embedding. The descriptions are built as follows.

- For every domain d and every slot s the user can inform on, the description is given by user goal <d> <s>. The corresponding value is 1, if that slot has been mentioned and 0 else.

- For every atomic user act d i s that was used in the current turn, the description is given by user act <d> <i> <s>. We consider each atomic user act as one information and only provide user acts that were used in the current turn to the model with a corresponding value of 1.

Attraction

Turn 0
User: I'm looking for a night club in the north area.
System: im sorry, we do not have a nightclub attraction like that, could i help you find something else?
Turn 1
User: Is there a night club in the centre?
System: sure, ballare is located in the centre, the postcode is cb23na and the entrance fee is 5 pounds.
Turn 2
User: thank you, bye
System: youre welcome. let us know if you need anything else.

Restaurant

Turn 0
User: I am interested in making a restaurant booking.
System: Did you want italian or not italian food?
Turn 1
User: I would like cheap Chinese food
System: may i suggest charlie chan? its located at regent street city centre and the phone number is 01223361763
Turn 2
User: Can you book it on Thursday for 2 people at around 19.45
System: ok ive got you booked for thursday. your reference number is 00000010.
Turn 3
User: thank you very much
System: Thanks for your participation.

Hotel

Turn 0
User: I need to book a hotel
System: will you be staying at a guesthouse? what is your price range and what amenities are you looking for?
Turn 1
User: I will stay at a guesthouse. I need parking and 3 stars
System: i recommend bridge guest house. it is in the moderate price range. its phone number is 01223247942.
Turn 2
User: What is the address?
System: bridge guest house is located at 151 hills road and has a postcode of cb28rj. does this sound okay?
Turn 3
User: bye
System: Thanks for your participation.

Figure 6: Example dialogues that were collected during the human trial. Users hired through Amazon Mechanical Turk interact with our DDPT model.

- For every atomic system act d i s that was used in the previous turn, the description is given by last system act <d> <i> <s> with a corresponding value of 1.

- For every domain d where a data base query is possible to obtain the number of entities that fulfill the user constraints, the description is given by data base <d> <number of entities> with a corresponding value indicating the number of search results.

- For every domain d where an entity can be booked, the description is given by general <d> <booked> with a binary indicating whether an entity has already been booked.

A.6 Human trial

We conducted a human trial to validate our results in simulation. The website was build using DialCrowd (Lee et al., 2018) and users were hired using Amazon Mechanical Turk. We used SetSUMBT (van Niekerk et al., 2021) as belief tracker and SC-GPT (Peng et al., 2020) as NLG module to accompany the dialogue policies Bin, Gold and DDPT in the dialogue system pipelines. Example dialogues, where DDPT interacted with users hired through Amazon Mechanical Turk, are depicted in Figure 6.

Task	Easy-to-hard Bin	Sem	DDPT	Hard-to-easy Bin	Sem	DDPT	Mixed order Bin	Sem	DDPT
Attraction	/	/	/	0.43	0.35	0.83	0.60	0.33	0.79
Taxi	0.51	0.75	0.90	0.51	0.47	0.85	0.35	0.43	0.77
Train	0.21	0.18	0.76	0.23	0.15	0.28	0.17	0.09	0.34
Restaurant	0.47	0.36	0.73	0.62	0.52	0.74	/	/	/
Hotel	0.36	0.26	0.53	/	/	/	0.39	0.28	0.39
Average	0.39	0.39	0.73	0.45	0.37	0.68	0.38	0.29	0.57
Random	0.39	0.26	0.34	0.39	0.26	0.34	0.39	0.26	0.34

Table 4: Forward transfer table showing for every domain i the metric $\mathcal{Z}_i$ in terms of success rate, where numbers range between 0 and 1. The higher the number, the more forward transfer is achieved.

Task	Easy-to-hard Bin	Sem	DDPT	Hard-to-easy Bin	Sem	DDPT	Mixed order Bin	Sem	DDPT	Random
Attraction	0.28	0.49	0.03	0.08	0.09	0.02	0.29	0.40	0.0	
Taxi	0.13	0.15	0.01	0.01	0.01	0.02	0.01	0.02	0.0	
Train	0.18	0.20	0.02	0.13	0.14	-0.01	0.03	0.03	0.0	
Restaurant	0.06	0.11	-0.01	0.16	0.19	0.0	0.22	0.26	0.09	
Hotel	0.04	0.07	0.0	0.32	0.41	0.07	0.14	0.19	0.03	
Average	0.14	0.20	0.01	0.14	0.17	0.02	0.14	0.18	0.03	0.43

Table 5: Forgetting table showing for every domain i the metric $\mathcal{F}_i$ in terms of success rate, where numbers range between -1 and 1. Negative numbers indicate backward transfer whereas positive numbers indicate forgetting.

Task	Easy-to-hard Bin	Sem	DDPT	Hard-to-easy Bin	Sem	DDPT	Mixed order Bin	Sem	DDPT
Attraction	/	/	/	-88	-124	12	-16	-125	-3
Taxi	-91	-32	23	-65	-117	13	-85	-127	-12
Train	-149	-156	-17	-66	-180	-108	-140	-189	-112
Restaurant	-94	-119	-15	-15	-97	-19	/	/	/
Hotel	-121	-143	-81	/	/	/	-45	-139	-107
Average	-114	-113	-23	-58	-129	-25	-71	-145	-58

Table 6: Forward transfer table showing for every domain i the metric $\mathcal{Z}_i$ in terms of average return. The higher the number, the more forward transfer is achieved.

Task	Easy-to-hard Bin	Sem	DDPT	Hard-to-easy Bin	Sem	DDPT	Mixed order Bin	Sem	DDPT
Attraction	99	151	6	34	36	2	93	126	1
Taxi	73	89	4	16	23	4	18	29	1
Train	68	68	1	43	49	-2	10	10	-1
Restaurant	35	38	-1	59	71	2	78	91	26
Hotel	12	21	-1	89	112	18	51	59	7
Average	58	73	2	48	58	5	50	63	7

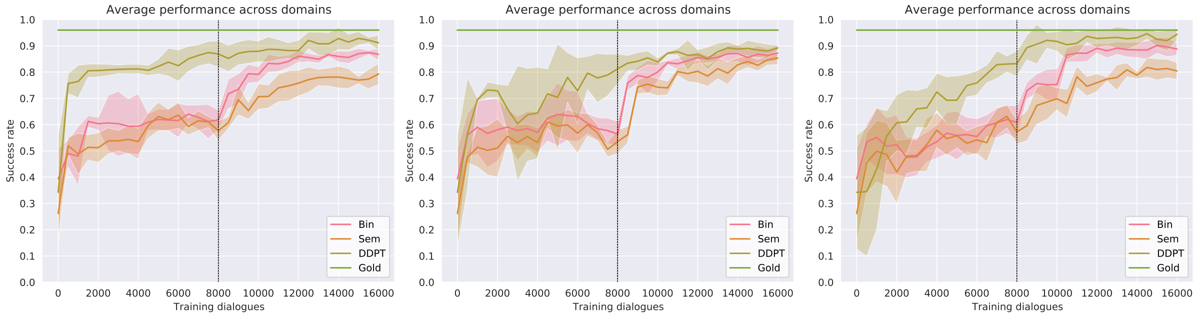
Table 7: Forgetting table showing for every domain i the metric $\mathcal{F}_i$ in terms of average return. Negative numbers indicate backward transfer whereas positive numbers indicate forgetting.

A.7 Forward Transfer and Forgetting

We provide the forward and forgetting tables in terms of success rate and average return in Tables 4, 5, 6, 7.

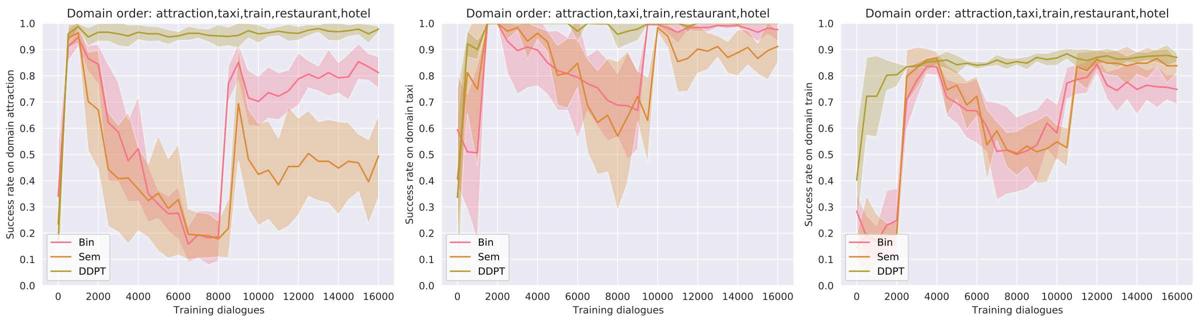
A.8 Continual Evaluation

Here, we provide in-depth results for all experiments. Each graph shows the performance of a single domain during training. Moreover, we provide the average performance over domains in terms of success rate in Figure 7 to complement Figure 4.

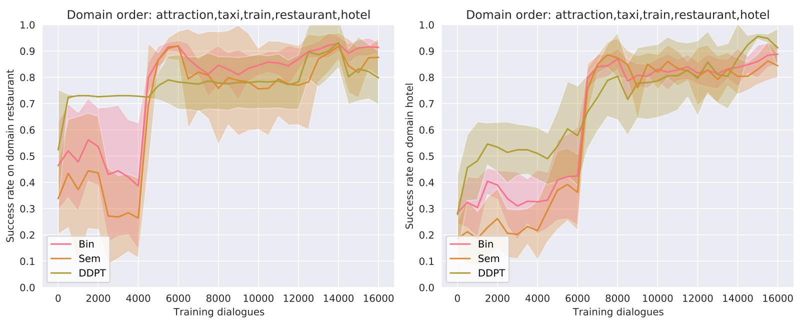


(a) easy-to-hard order (b) hard-to-easy order (c) mixed domain order

Figure 7: Training the three architectures Bin, Sem and DDPT using CLEAR on three different domain orders, each with 5 different seeds. Each model is evaluated every 500 training dialogues on 100 dialogues per domain. The plots show the success rate, where performance is averaged over domains. The vertical line at 8000 dialogues indicates the start of cycle 2.

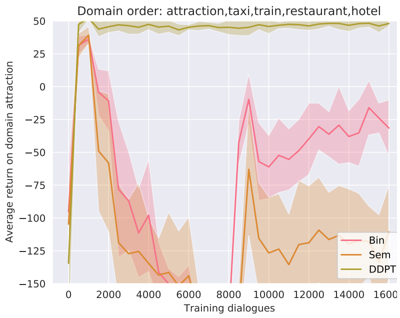


(a) Success rate on attraction domain (b) Success rate on taxi domain (c) Success rate on train domain

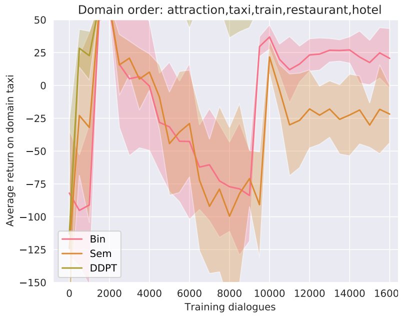


(d) Success rate on restaurant domain (e) Success rate on hotel domain

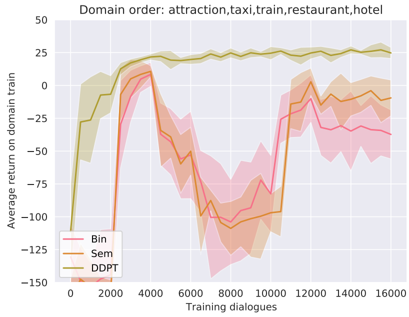
Figure 8: Success rate for each individual domain, where algorithms are trained in the order easy-to-hard.



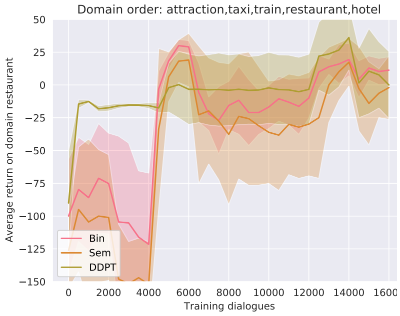
(a) Average return on attraction domain



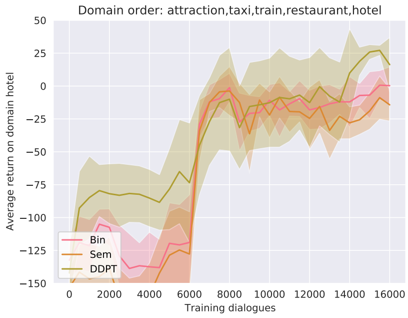
(b) Average return on taxi domain



(c) Average return on train domain

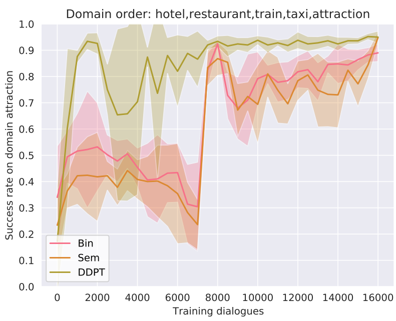


(d) Average return on restaurant domain

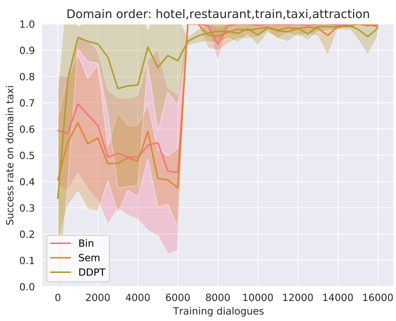


(e) Average return on hotel domain

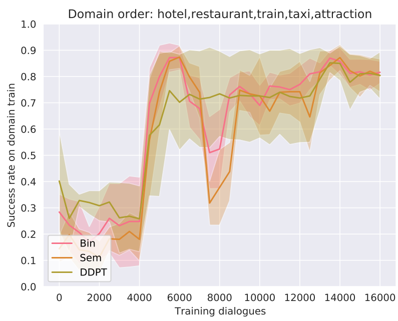
Figure 9: Average return for each individual domain, where algorithms are trained in the order easy-to-hard.



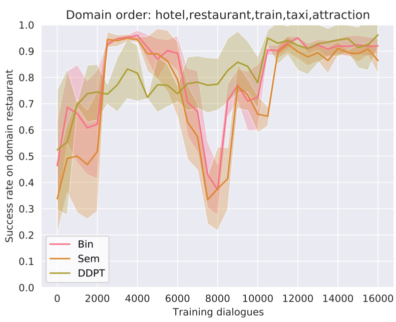
(a) Success rate on attraction domain



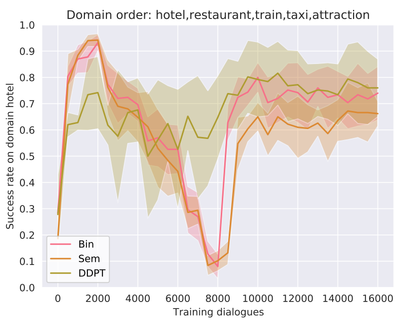
(b) Success rate on taxi domain



(c) Success rate on train domain

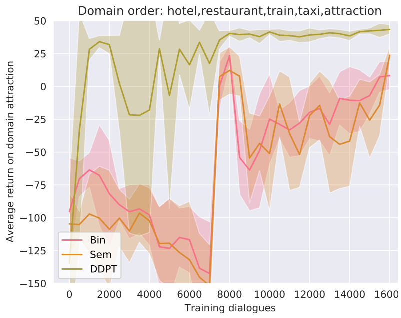


(d) Success rate on restaurant domain

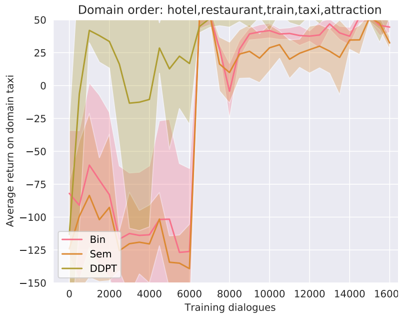


(e) Success rate on hotel domain

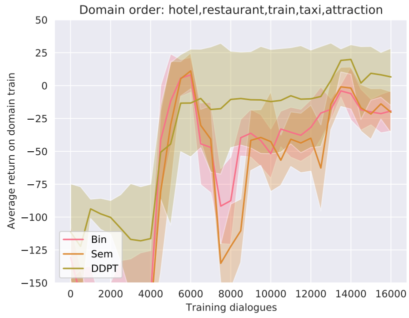
Figure 10: Success rate for each individual domain, where algorithms are trained in the order hard-to-easy.



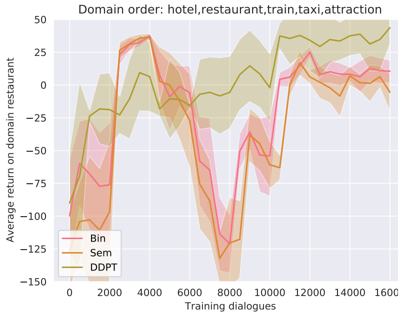
(a) Average return on attraction domain



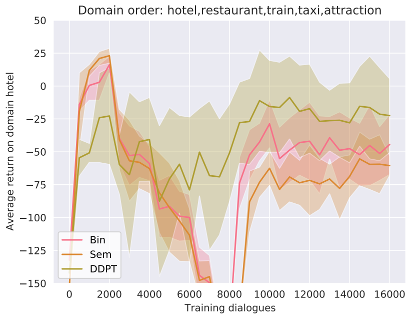
(b) Average return on taxi domain



(c) Average return on train domain

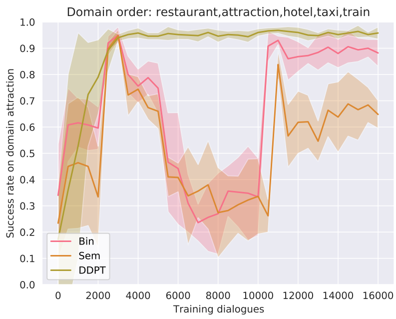


(d) Average return on restaurant domain

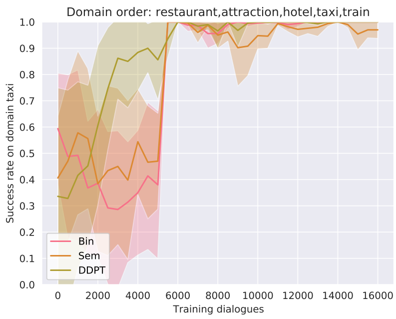


(e) Average return on hotel domain

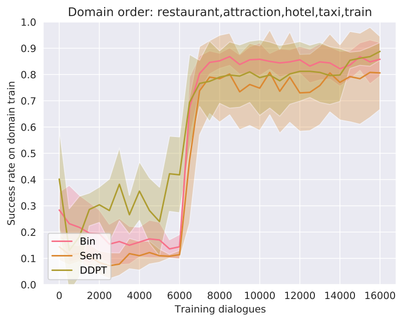
Figure 11: Average return for each individual domain, where algorithms are trained in the order hard-to-easy.



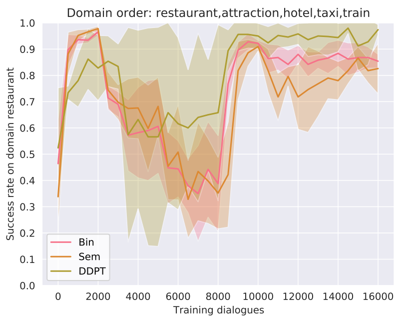
(a) Success rate on attraction domain



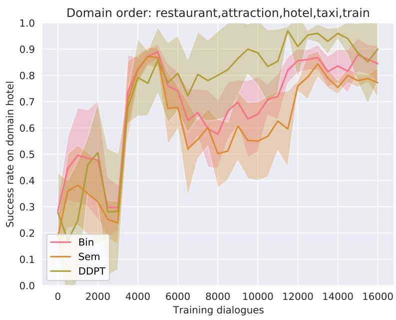
(b) Success rate on taxi domain



(c) Success rate on train domain

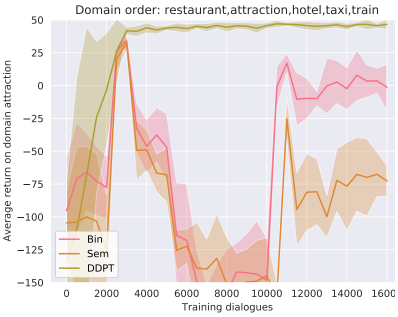


(d) Success rate on restaurant domain

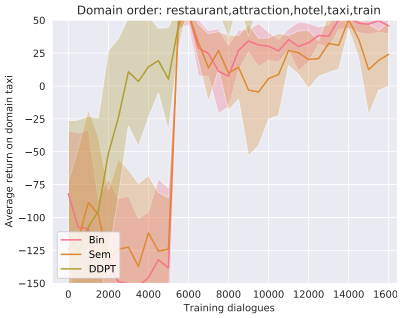


(e) Success rate on hotel domain

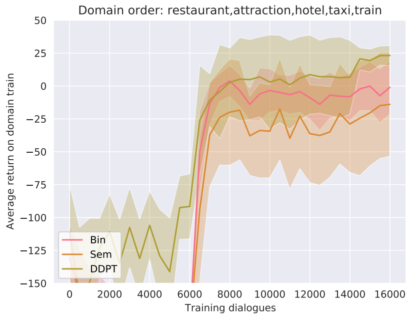
Figure 12: Success rate for each individual domain, where algorithms are trained in the order mixed.



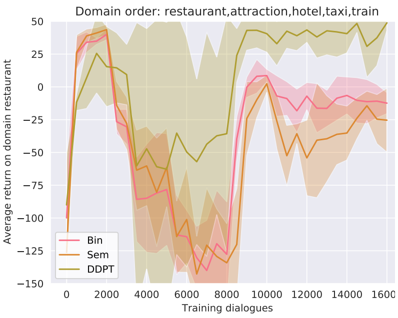
(a) Average return on attraction domain



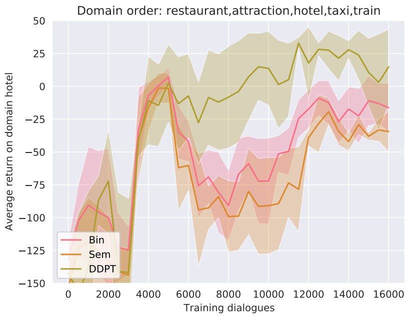
(b) Average return on taxi domain



(c) Average return on train domain



(d) Average return on restaurant domain



(e) Average return on hotel domain

Figure 13: Average return for each individual domain, where algorithms are trained in the order mixed.

Chapter 7

Learning with an Open Horizon in Ever-Changing Dialogue Circumstances

This chapter summarizes our work on learning with an open horizon in ever-changing dialogue circumstances and gives a verbatim copy of our paper (Geishauser et al., 2024):

Christian Geishauser et al. (2024). "Learning With an Open Horizon in Ever-Changing Dialogue Circumstances". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 32, pp. 2352–2366. DOI: 10.1109/TASLP.2024.3385289

7.1 Summary

The study of continual learning in dialogue has so far focused on the setup where individual domains are observed sequentially and only for a period of time. Once a new domain is introduced, the previously observed domain is not present anymore. This configuration is well-suited for conducting an examination of forward transfer to unseen domains and forgetting of previously observed domains. Nevertheless, it is arguable whether this defines a realistic setup for dialogue due to the following reasons. Firstly, while the introduction of a new domain might lead to a lower occurance for old domains for a period of time, it should not lead to their entire absence. On the contrary, the new domain should be combined with previous ones to form multi-domain dialogues. Secondly, user demands for specific domains change over time, for instance due to seasonal changes, which is not modeled in the current setup. Lastly, current setups lack the modeling of multiple user behaviors that are inevitably encountered during the lifetime of a dialogue agent.

In this work, we propose realistic environments for continual RL of dialogue policies (RECORD), a more realistic continual learning setup for dialogue, which takes into account 1) multiple user behaviors, 2) multi-domain dialogues, 3) reoccuring domains, and 4) changing user demands over time. RECORD generalizes the previous setup and is both flexible and controllable. In order to learn in realistic, ever-changing dialogue environments, we propose the usage of lifetime return for optimizing the dialogue policy. Lifetime return calculates the sum of rewards until the end of the agent's lifetime and thus provides a balance between the present and the (possibly different) future. This creates a more robust learning of the agent in changing environments. In order to address the challenge of hyperparameter adaptation in changing environments, we propose the usage of meta-gradient RL. In this regard, we are the first to leverage meta-gradient RL for dialogue policy optimization for learning the hyperparameters of the underlying algorithm.

We conduct extensive experiments with single and multiple user simulators, two algorithms and various continual learning setups. We show that the usage of lifetime return facilitates continual learning in every setup. The additional utilization of meta-gradient RL leads to the best results, showing the benefits of our proposals.

7.2 Personal contributions

The implementation, technical results and writing are my contribution. Co-authors assisted in writing and proofreading.

Learning With an Open Horizon in Ever-Changing Dialogue Circumstances

Christian Geishauser, Carel van Niekerk, Nurul Lubis, Hsien-chin Lin, Michael Heck, Shutong Feng, Benjamin Ruppik, Renato Vukovic, and Milica Gašić, *Senior Member, IEEE*

***Abstract*—Task-orienteddialogue systems aid users in achieving their goals for specific tasks, e.g., booking a hotel room or managing a schedule. The systems experience various changes during their lifetime such as new tasks emerging or varying user behaviours and task requests, which requires the ability of continually learning throughout their lifetime. Current dialogue systems either perform no continual learning or do it in an unrealistic way that mostly focuses on avoiding catastrophic forgetting. Unlike current dialogue systems, humans learn in such a way that it benefits their present and future, while adapting their behaviour to current circumstances. In order to equip dialogue systems with the capability of learning for the future, we propose the usage of lifetime return in the reinforcement learning (RL) objective of dialogue policies. Moreover, we enable dynamic adaptation of hyperparameters of the underlying RL algorithm used for training the dialogue policy by employing meta-gradient reinforcement learning. We furthermore propose a more general and challenging continual learning environment in order to approximate how dialogue systems can learn in the ever-changing real world. Extensive experiments demonstrate that lifetime return and meta-gradient RL lead to more robust and improved results in continuously changing circumstances. The results warrant further development of dialogue systems that evolve throughout their lifetime.**

***Index Terms*—Task-oriented dialogue, dialogue policy, deep reinforcement learning, continual learning.**

I. Introduction

TASK-ORIENTED dialogue systems aid users in achieving their goal during a conversation. The goal is commonly comprised of certain tasks (typically called domains), such as managing a schedule or planning a journey. Modular dialogue systems have a tracking component that maintains information about the dialogue in a dialogue state, and a decision-making component that models the underlying policy, i.e., the selection of actions [1], [2], [3], [4]. The array of concepts that a task-oriented dialogue system can understand and talk about is defined by an ontology. The ontology is comprised of domains, e.g. *restaurants* or *hotels*, domain-specific slots, e.g. the *area* or *price*, and values that a slot can take, e.g *west*, *east*, *north*, *south*, and *center* for the area slot.

A dialogue system that interacts with the real world faces unique challenges during its lifetime. The sheer amount of tasks a dialogue system can potentially handle will result in new tasks emerging over time. Moreover, a dialogue system might need to fulfill multiple tasks within a dialogue, such as booking a flight and finding a hotel, which increases the difficulty of achieving the user goal. The system will experience a multitude of user behaviours throughout its lifetime. In addition, user needs will change over time, for instance due to seasonal changes where hotels are more frequently demanded in the vacation season. Continual learning defines the adequate learning platform for modelling these dialogue challenges as it is concerned with non-stationary, ever-changing environments [5], [6]. Nevertheless, as shown in Fig. 1(a), current setups in continual learning for dialogue operate with a single type of user behaviour and rely on the assumption that the dialogue system is exposed to a single, fixed domain for a period of time. Once another domain is introduced, the previous domain is never seen again [7], [8], [9]. Moreover, the performance is measured on all domains seen even though the dialogue system does not encounter previous domains anymore. Consequently, they mainly focus on the past instead of considering for the future. This unrealistic setup focuses mainly on catastrophic forgetting but ignores important challenges such as encountering multiple user behaviours, user goals with multiple domains or changing user demands that are inevitably encountered in lifelong dialogue policy learning. If dialogue systems are supposed to be deployed for real-world applications, where they continue learning over time, the simulation of these challenges is essential.

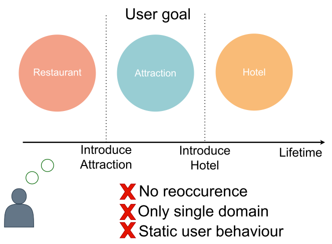
Dialogue policies for continual learning are best optimised using continual reinforcement learning. The learning objective is to maximise episodic return, i.e. the cumulative sum of rewards obtained during an *episode*, i.e. dialogue in our case [10], [11]. Nevertheless, episodic return, which considers the reward only

Manuscript received 14 June 2023; revised 20 November 2023; accepted 27 March 2024. Date of publication 4 April 2024; date of current version 22 April 2024. The work of Christian Geishauser, Hsien-chin Lin, Benjamin Ruppik, and Renato Vukovic was supported by the European Research Council (ERC) through the Horizon 2020 Research and Innovation Programme under Grant Agreement STG2018804636. The work of Michael Heck, Nurul Lubis, Shutong Feng, and Carel van Niekerk was supported by the Alexander von Humboldt Foundation in the framework of the Sofja Kovalevskaja Award endowed by the Federal Ministry of Education and Research. Computational infrastructure and support were provided by the Centre for Information and Media Technology at Heinrich Heine University Düsseldorf and the Google Cloud Platform. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Jiajun Zhang. *(Corresponding author: Christian Geishauser.)*

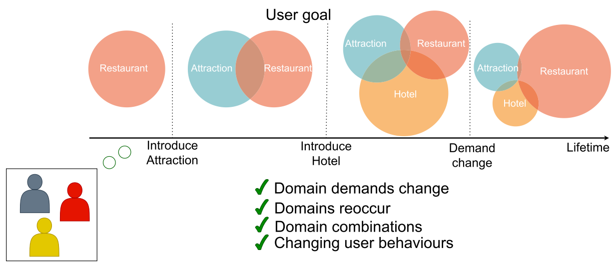
This work involved human subjects or animals in its research. Approved by Prof. Dr. Martin Heil and Prof. Dr. Bettina Pause on 26.11.2019 and 01.11.2021 respectively. This approval was made on behalf of the Heinrich Heine University, Düsseldorf ethics committee.

The authors are with the Heinrich Heine University Duesseldorf, 40225 Duesseldorf, Germany (e-mail: geishaus@hhu.de; niekerk@hhu.de; lubis@hhu.de; lin@hhu.de; heckmi@hhu.de; shutong.feng@hhu.de; benjamin.ruppik@hhu.de; renato.vukovic@hhu.de; gasic@hhu.de).

Digital Object Identifier 10.1109/TASLP.2024.3385289



(a) Previously proposed continual learning setup in dialogue



(b) An instance of our proposed RECORD setup

Fig. 1. In previously proposed setups for continual dialogue policy learning, dialogues were conducted with a single type of user behaviour and only with single domain that does not reoccur once a new domain is introduced. In our proposal RECORD, we simulate multiple user behaviours, multi-domain dialogue goals and reoccuring domains. We moreover model user demand changes where the likelihood for requested domains varies.

until the end of the dialogue, is biased on the present and neglects the changes that might happen in the future. In order to take the future into account, we propose to incorporate lifetime return [12] into the learning objective of the dialogue policy, which we refer to as **learning with an open horizon**. Lifetime return formally calculates the sum of rewards until the end of the *lifetime* and thus provides a balance between the present and the (possibly different) future.

Moreover, if the environment is changing, the optimal hyperparameters of the underlying algorithm can depend on the current circumstances faced and are likely to change over time [13], [14]. In order to address this problem, we propose the usage of meta-gradient RL [15] to continually learn the hyperparameters during the lifetime of the dialogue policy. We contribute the following:[1]

- We propose a sophisticated environment for continual learning of dialogue policies, called RECORD, including 1) multiple user behaviours, 2) multi-domain dialogues, 3) reoccuring domains, and 4) changing user demands.
- We are the first to use lifetime return for continual dialogue policy learning and the first to explore the optimisation of both episodic and lifetime return.
- We are the first to use meta-gradient reinforcement learning for dialogue policy optimisation.

In the age of large language models such as ChatGPT, that are constantly fine-tuned through RL to continue improving and keep their knowledge updated, our proposed methods for adapting to environmental changes are ever more important.

II. Background

A. Continual Reinforcement Learning

The formal framework in RL [16] is given by a Markov decision process (MDP) $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, r, p, p_0, \gamma\}$. Here, $\mathcal{S}$ denotes the (continuous) state space, $\mathcal{A}$ is the action space, r is the reward function, $p(s'|s,a)$ is the transition probability that models the probability of transitioning to state s' after executing action a in state s, and $p_0(s)$ is the probability of starting in state s. γ denotes the discount factor. A MDP defines an *episodic* environment if a terminal state is reached after finitely many steps (which holds for dialogue), and *non-episodic* otherwise. Instead of γ being a real-valued number as it is common in the RL definition, [17] uses transition-based discounting to unify the episodic and non-episodic return calculation: let $(s_t, a_t, r_t, s_{t+1}, \ldots)$ be an infinite sequence of states, actions, rewards and next states where some of these states can be terminal (i.e. the episode ends). For a given time-step t, we define the discounted return

$$R_t = \sum_{i=0}^{\infty} \left(\prod_{j=0}^{i-1} \gamma(s_{t+j}, a_{t+j}, s_{t+j+1}) \right) \cdot r_{t+i} \qquad (1)$$

where γ is a transition-based discount function that we henceforth abbreviate with $\gamma_j := \gamma(s_j, a_j, s_{j+1})$ if it is clear from the context. For episodic environments, we can obtain the episodic return by defining $\gamma_{t+j} = 0$ if s_{t+j} is a terminal state and otherwise a constant value $\gamma_{t+j} = \gamma_c$ between 0 and 1. This effectively cuts off the influence of the future from time-step $t+j+1$ onward. The non-episodic case is retrieved from (1) by defining the transition-based discounting γ as a strictly positive number for every state. The goal of the agent is to maximise the discounted return in expectation in every state s, which is called the state-value $V(s)$. The related Q-function for a state-action pair (s,a) is the expected discounted return when executing action a in state s. Lastly, the policy $\pi(a \mid s)$ defines a probability distribution over actions for every state s and is used for action selection.

In contrast to a typical RL use case that operates in a fixed MDP, in continual RL the agent experiences non-stationary or changing environments [5], [18]. In its broadest sense, the agent faces a sequence of Markov decision processes $\{\mathcal{M}_w\}_{w=1}^{\infty}$ with possibly different transition dynamics, reward functions or even state or action spaces [18], [19], [20]. Each such MDP $\mathcal{M}_w$ can thus model a different circumstance. A typical assumption on the MDP sequence is local stationarity [21], i.e. the agent collects episodes using an MDP $\mathcal{M}_w$ for some period of time before the next MDP emerges [10], [11], [22].

A common goal of continual learning is to maximise performance on all MDPs, which requires prevention of catastrophic forgetting if previous MDPs are not seen anymore [11], [23]. The evaluation of the agent is performed on all MDPs, no matter whether the MDPs occur again. As mentioned in [18], it is questionable why an agent should retain performance on tasks that are never observed again. Moreover, this requires separate evaluation phases during training that are not always available [18]. Thus, an alternative form of performance measure

[1] [Online]. Available: https://gitlab.cs.uni-duesseldorf.de/general/dsml/record-publichttps://gitlab.cs.uni-duesseldorf.de/general/dsml/record-public.

that we also employ in this work is given through the cumulative performance over the agent's lifetime [21], [24] and the average performance within a time-window [25].

B. Actor-Critic RL

RL methods can be distinguished into value-based and policy-based methods [16]. Policy-based methods parameterise the policy π_θ directly using parameters θ, whereas value-based methods learn a value-function. Actor-critic methods are a combination of the two methods and learn both an actor π_θ and a critic $V_\psi(s)$ (or $Q_\psi(s,a)$). The critic approximates the state-function $V(s)$ by minimising the mean-squared error between its prediction and a target. The target could be the discounted return as in (1), the vtrace-target [26], or the one-step look ahead $r + V_\psi(s')$ for instance. Let the advantage function be defined as

$$A(s,a) = Q(s,a) - V(s) = \mathbb{E}_\pi[r + V(s') - V(s)], \quad (2)$$

where s' denotes the state reached after executing action a in state s. We emphasise here that the advantage can be expressed in terms of the value-function V. Policy updates are based on the policy gradient theorem [16] and update the policy parameters in the direction

$$\mathbb{E}_\pi[A(s,a)\nabla_\theta \log \pi_\theta(a \mid s)]. \quad (3)$$

The expectation with respect to π in (3) means that we require generating episodes using π to approximate the expectation, which is called on-policy. In contrast, off-policy methods build an approximation using episodes generated by a different policy μ, often called behaviour policy, which can also be a previous version of π. Off-policy methods can hence reuse generated data multiple times, which makes them generally more sample-efficient in comparison to on-policy algorithms. In general, for actor-critic methods the advantage function guides the policy optimisation and judges which actions should be reinforced and which should be suppressed. Instead of the advantage function, one could also use the return R_t or Q-function [27].

C. Meta-Gradient Reinforcement Learning

Meta-gradient RL is a general framework for continually adapting differentiable hyperparameters η of the RL algorithm [15], [28]. Consider a loss function $\mathcal{L}^{\text{inner}}(\theta, \eta)$ that depends on model parameters θ to be learned and differentiable hyperparameters η (e.g. the learning rate or weights for regularisation losses). The differentiable hyperparameters η that are optimised during meta-gradient RL are called metaparameters and are a subset of the hyperparameters.

In every minimisation step for $\mathcal{L}^{\text{inner}}$, which is called an inner loop update, we optimise parameters θ_t by updating according to $\theta_{t+1}(\eta) = \theta_t - \nabla_{\theta_t}\mathcal{L}^{\text{inner}}(\theta_t, \eta)$. We note that the acquired parameters θ_{t+1} have a dependence on η. After m inner loop updates with fixed η, we obtain parameters $\theta_{t+n}(\eta)$. For a specified outer loss function $\mathcal{L}^{\text{outer}}$, meta-gradient RL then performs the outer loop update (or meta update) by $\eta' = \eta - \nabla_\eta \mathcal{L}^{\text{outer}}(\theta_{t+n}(\eta))$ and the process repeats.

D. Task-Oriented Dialogue Policy

We view dialogue as a sequence of turns between a user and a dialogue system. The sequential nature of dialogue motivates the optimisation through RL, where the corresponding MDP properties are explained as follows. In every turn of the conversation, the user produces a **user action** (or utterance on text level), which is composed of a list of atomic actions. Examples for atomic actions are given by *Restaurant-inform-price-cheap* or *Hotel-request-address-?*, which means that the user informs the system that the restaurant should be cheap or that the user requires the address of a hotel, respectively. The **dialogue state** is commonly based on the hidden information state of the dialogue system [29]. This hidden information state consists of the *current user goal*, the *user action*, and the *dialogue history*. Here, the goal is composed of a list of domains and each domain comes with a list of constraints, e.g. the hotel has to be cheap and in the centre and the restaurant has to serve Italian food. Since the current user goal already summarises the history to some extent, the dialogue history information in the state typically only includes the last chosen dialogue system action [30], [31]. In addition, the state contains information about the database results, for instance the number of entities that are available given the current user goal. We note that the dialogue state can moreover incorporate uncertainty about the predicted user goal or user act, which is then referred to as dialogue belief state [32].

Given the state, the dialogue policy selects a **system action**, which is given by a list of atomic actions. For instance, the atomic actions *Restaurant-inform-address* or *Train-book* mean that the system informs the address of a restaurant or books a train for the user. The large number of atomic actions and their combinations produce a large action space for dialogue policies that moreover grows in continual learning [9], [33]. The dialogue ends if the user regards it as completed or a maximum number of turns T_{max} has been reached (where typically $T_{\text{max}} = 40$ [34]). The **reward function** captures the two important objectives of a task-oriented dialogue system: dialogue success and efficiency. To encourage efficiency, the policy receives a small negative reward (typically -1) in every dialogue turn. Once the dialogue is finished, the policy obtains a large positive or negative reward (in comparison to the efficiency reward) depending on dialogue success or failure, encouraging the policy to first favor success optimisation over efficiency. The rewards for dialogue success and failure are typically set to $2 \cdot T_{\text{max}}$ and $-T_{\text{max}}$, respectively, such that the highest and lowest reachable returns will be roughly $2 \cdot T_{\text{max}}$ and $-T_{\text{max}} + T_{\text{max}} \cdot (-1) = -2 \cdot T_{\text{max}}$ [34], [35].

Lastly, the **starting state probability** and **transition probabilities** are governed by the user behaviour, which determines how the user reacts to the system, and the user goal distribution. For instance, the likelihood of starting a conversation with restaurant or hotel related information depends on the user goal distribution and user behaviour. Moreover, regarding the user behaviour, some users tend to provide more information than others in a single turn, or are less patient and more quickly jump to the next domains. An example of user action, state, system action and reward is provided in Appendix Fig. 9.

III. Learning With Open Horizon in Ever-Changing Environments

A. RECORD: Realistic Environments for Continual RL of Dialogue Policies

In task-oriented dialogue, the system aids users in achieving their goals, where the goal is composed of a list of domains and each domain comes with a list of constraints. The system learns from a sequence of dialogues during its lifetime, where each dialogue comes with a certain user goal, which thus gives rise to a sequence of goals. Previous works in continual learning for dialogue focus on goal sequences which can be subdivided into smaller goal sequences that only contain single-domain goals with a fixed domain. The fixed domain changes once another domain is introduced.

In the following, we strive for a more challenging and general dialogue learning scenario that is both flexible and controllable, where 1) domains can reoccur anytime in goals of a sequence, 2) multiple domains can be present in goals of a sequence, 3) the probability of a domain occurring in a goal varies throughout the lifetime, thereby modelling user demand changes, and 4) different user behaviours are encountered for different dialogues. The difference between our proposed continual learning setup and the previously proposed setup is shown in Fig. 1.

The dialogue agent faces a sequence of MDPs in continual RL, where we assume local stationarity [21], i.e. the agent is faced with a fixed MDP for a certain amount of dialogues until a new MDP is created. We model MDP changes with a timeline, which specifies the number of dialogues that should be conducted before a new MDP is introduced. A change in MDP can be caused by two different events:

1) a new domain is introduced to the system: the emerging MDP has an expanded state and action space compared to before since the agent needs to understand new concepts for the new domain (e.g. if the hotel has wifi) as well as select new actions (e.g. inform the address of the hotel). Moreover, the transition probabilities change since the probability for obtaining states that consider the new domain is positive.
2) The user request probability for specific domains, i.e. the user demand, changes: the emerging MDP then has a different transition probability as the likelihood of encountering a specific domain has changed.

We now explain how user goals are constructed in our framework for an MDP $\mathcal{M}$ during the agent's lifetime, see also Algorithm 1. Firstly, we check whether the MDP has emerged as a consequence of 1) a new introduced domain d_{new} or 2) a user demand change. In the first case (event 1), we determine if the new domain should be included in the user goal by sampling from a Bernoulli distribution Bernoulli(p_{new}), where p_{new} models the probability that d_{new} is part of the goal. If d_{new} should be part of the goal, we initialise the goal with d_{new} (Algorithm 1, line 5), and otherwise keep the goal as an empty list. If $p_{\text{new}} < 1$, there exists a chance that the newly introduced domain is not included in the goal. Consequently, a previously introduced domain is required to be in the goal, resulting in the **reoccurance of the domain** (Algorithm 1, lines 9-11). The higher we choose the probability p_{new} of a new domain being part of the goal, the more frequently the policy has to adapt to a new domain. A domain is considered new until the MDP changes again.

Algorithm 1: Construct User Goal.

Require: d_{new}, p_{new}, p_{num}, σ
- d_{new}: newly introduced domain or None
- p_{new}: probability that d_{new} is in the goal
- p_{num}: distribution for how many domains should be in the user goal
- σ: decides on user demand deviation

1: goal $\leftarrow$ []
2: **if** d_{new}**then** is not None $\triangleright$ domain d_{new} was introduced
3: use_$d_{\text{new}} \sim$ Bernoulli(p_{new})
4: **if** use_d_{new} **then**
5: goal $\leftarrow [d_{\text{new}}]$
6: **if** MDP changed **then**
7: $z_i \sim \mathcal{N}(0, \sigma)$ for all introduced domains d_i
8: num_domains $\sim p_{\text{num}}$
9: **while** len(goal) $\neq$ num_domains **do**
10: $d_j \sim p_{\text{demand}} = \text{softmax}(z_i \mid z_i \neq d_{\text{new}}, z_i \notin \text{goal})$
11: goal.append(d_j)
12: **return** goal

Secondly, regardless whether the MDP emerged as a consequence of event 1) or 2), we determine the number n of domains that the user goal should consist of by sampling from a pre-defined distribution p_{num} over $\mathbb{N}$. If $p_{\text{num}}(1) < 1$, there exists a chance that the goal comprises more than a single domain, resulting in **multiple domains per goal**. As goals with many domains are more difficult to solve, specifying a high probability for larger n in p_{num} allows us to model the difficulty of the environment. To obtain n domains for the user goal (which is so far either empty or only consists of d_{new}), we continue sampling from a distribution p_{demand} over introduced domains. We note that even in the case $p_{\text{new}} = 1$, a previously introduced domain d_{prev} can reoccur in multi-domain dialogues if $p_{\text{num}}(1) < 1$ and $p_{\text{demand}}(d_{\text{prev}}) > 0$.

While the distributions p_{new} and p_{num} are pre-defined and fixed in our setup, we use the distribution p_{demand} in order to model **changing user demands over time**. The distribution p_{demand} is altered every time the MDP change is caused by a change in user demands according to our timeline (i.e. event 2). We model different values for p_{demand} as follows. The initial choice for p_{demand} is such that all introduced domains are equally likely to occur in a goal as we can not argue a priori why one domain will be more frequently demanded than another. Nevertheless, some domains might be on average more frequently demanded by users than others for some period of time (for instance due to seasonal changes). For every introduced domain d_i, we model the demand for d_i by a noise variable Z_i with distribution $\mathcal{N}(0, \sigma)$, where σ is the variability of deviation. Whenever the MDP changes, we sample from $\mathcal{N}(0, \sigma)$ to obtain values z_i. The demand probability p_{demand} over introduced domains is then given by softmax(z_i) and kept fixed until another

MDP change occurs. This has the intuitive interpretation that demand for d_i is lower for $z_i < 0$ and higher for $z_i > 0$ while on average we obtain the uniform distribution (i.e. domains are equally likely) as the mean is set to 0. Finally, once the user goal is constructed, we uniformly sample a user from our given set of possible users that interacts with the dialogue agent, thereby modelling **different user behaviours**.

We note that setting $p_{\text{new}} = p_{\text{num}}(1) = 1$ and allowing only the introduction of new domains, we retain the previous setup in continual dialogue learning (see Fig. 1(a)). Due to its generality, our framework can be also used to study related problems such as how to deal with composite goals [36] or multiple user behaviours. We abbreviate our endeavour towards **r**ealistic **e**nvironments for **co**ntinual **r**einforcement learning of **d**ialogue policies as **RECORD**. We emphasise here that by realistic environment, we do not necessarily mean we simulate exactly what may be happening in the real world, but aim to simulate the most important challenges (see Section I) that need to be overcome by a continually learning agent.

Algorithm 2: Continual Policy Learning with Open Horizon.

Require: policy π_θ, episodic critic V_ψ^{epi}, lifetime critic V_ψ^{life}, meta-frequency m, meta parameters η, episodes_until_update, actor-critic algorithm $\mathbb{A}$.

num_updates $\leftarrow 0$
num_episodes $\leftarrow 0$
while lifetime not ended **do**
  Create episode using π_θ and save in buffer
  $\text{num\_episodes} += 1$
  if num_episodes $\bmod$ episodes_until_update $= 0$ **then**
    Update V_ψ^{epi} towards expected episodic return
    Update V_ψ^{life} towards expected lifetime return
    Update π_θ to minimize episodic and lifetime return using critics V_ψ^{epi} and V_ψ^{life}
    num_updates $+= 1$
  if num_updates $\bmod m = 0$ **then**
    Update meta-parameters η for lifetime return

B. Optimising for Lifetime

Continual RL for episodic tasks such as dialogue typically aims at maximising the episodic return, i.e. the cumulative sum of rewards until the end of the *episode* (or dialogue in our case) [10], [11]. In a changing environment, this produces biased updates that are overly focused on the current experience distribution rather than the expected future distribution [18]. Nevertheless, lifelong environment changes as modelled in RECORD require the model to take these future variations into account. To deal with this additional challenge, we propose the usage of lifetime return in addition to episodic return for dialogue policy optimisation.

For the episodic return, the transition-based discounting becomes 0 in a terminal state s_{terminal} (for a time-step t_{terminal}). The infinite sum in (1) thus becomes a finite sum

$$R_t^{\text{epi}} = \sum_{i=0}^{t_{\text{terminal}}} \gamma_c^i \cdot r_{t+i} \tag{4}$$

which only looks ahead until the end of the *episode* for a fixed discount factor γ_c.

In the non-episodic case, i.e. without terminal states, the discounting is a strictly positive constant $\gamma_c > 0$ for every state and we can seek to optimise the infinite horizon return in every time point t [18]. In contrast to R_t^{epi}, the lifetime return takes into account all future states until the end of the agent's *lifetime*. In order to leverage lifetime return for our episodic environment, we artificially cast it to be non-episodic by defining the transition based discount factor to be $\gamma_c > 0$ for terminal states as well. We abbreviate the resulting lifetime return in time step t by

$$R_t^{\text{life}} = \sum_{i=0}^{\infty} \gamma_c^i \cdot r_{t+i} \tag{5}$$

and refer to the optimisation of it as learning *with an open horizon*. As our agent faces a never-ending life where circumstances change over time, the lifetime objective formally takes these changes into account, whereas episodic return only considers the present distribution.

To optimise for both the present and future, we thus propose to maximise the returns R_t^{epi} and R_t^{life}. In practice when we apply it to actor-critic RL, we learn two critics V_ψ^{epi} and V_ψ^{life} that approximate the expected episodic return R_t^{epi} and lifetime return R_t^{life}, respectively. For the sake of clarity, in the following we use the standard policy gradient as given in (3), but the additions can be applied to any actor-critic algorithm. If A_ψ^{epi} and A_ψ^{life} denote the advantage estimates for V_ψ^{epi} and V_ψ^{life}, we update π in the direction of

$$\mathbb{E}_\pi[(A_\psi^{\text{epi}} + A_\psi^{\text{life}})\nabla_\theta \log \pi_\theta(a|s)], \tag{6}$$

which will reinforce or suppress actions based on both episodic and lifetime advantage.

C. Hyperparameter Learning for Lifetime

Continual learning algorithms often have additional loss terms that affect learning, such as a policy entropy loss $\sum_a \pi_\theta(a|s) \cdot \log \pi_\theta(a|s)$ to enhance exploration or regularisation losses (e.g. KL-divergence $\text{KL}[\mu||\pi]$ with the past policy μ) to prevent catastrophic forgetting [10]. These loss terms are weighted by hyperparameters to balance their impact with the actual objective of maximising return. The hyperparameters can depend on the current circumstances faced, which makes it difficult to set them optimally beforehand. To mitigate that problem, we propose the usage of meta-gradient RL for learning hyperparameters of the underlying dialogue policy RL algorithm. We optimise hyperparameters such that the lifetime return is maximised (as an alternative, one could also optimise hyperparameters for episodic and lifetime return). If we meta-learn the weight for a regularisation loss to imitate past behaviour, this has the intuitive interpretation that we only imitate past behaviour to the extent that it benefits future lifetime performance. Similarly for the entropy loss, we only require high entropy if the exploration is needed for obtaining a better lifetime performance.

In practice, we perform meta-gradient RL by updating the policy m times using (6) with fixed η and then update the meta-parameters in the direction of

$$\mathbb{E}_\pi[A_\psi^{\text{life}} \nabla_\eta \log \pi_\eta(a \mid s)]. \quad (7)$$

We depict a high-level pseudo code of the overall optimisation in Algorithm 2.

IV. Experimental Setup

In this section our aim is to explore how the addition of lifetime return and meta-gradient RL aids learning in ever-changing environments given by our proposed RECORD environment.

A. RECORD Environment

We conduct the experiments using the recently introduced ConvLab-3 framework [34], which provides a rule-based simulator [37] as well as the transformer-based user simulator TUS [38] for the MultiWOZ environment [39]. The rule-based simulator can have either high or low initiative, which decides on how many actions the user simulator takes in a turn. We run experiments with three user setups: 1) single rule-based user of high initiative as it is the default in ConvLab-3, 2) single TUS, and 3) multiple user behaviours: the two rule-based user simulators with high and low initiative and TUS. We introduce 5 domains of MultiWOZ in the order *restaurant*, *attraction*, *hotel*, *taxi* and *train*. We omit the domains *hospital* and *police* as in [9] since they are considered trivial. The user and dialogue policy use semantic actions for interaction. All algorithms were run on 5 different seeds and the results are averaged.

Our continual learning setup is constructed as follows. We set $p_{\text{new}} = 0.8$, which means that on average 80% of goals contain the new domain. The high number will require the policies to adapt as fast as possible to the new domain. We use $p_{\text{num}}(1) = 0.3, p_{\text{num}}(2) = 0.5, p_{\text{num}}(3) = 0.2$, i.e. on average 50% of the dialogue goals consist of two domains and 20% of three domains, which produces a mix between simple and complex goals. In the first part of its lifetime, the dialogue agent only faces MDP changes caused by the introduction of new domains. We introduce a new domain after 2000, 5000, 9000 and 11000 dialogues, which gives the policy time to adapt to the new domain before yet another domain is introduced. We use a value of $\sigma = 0$ during that stage, which means that p_{demand} is a uniform distribution. The domains are introduced in a fixed domain order as it has been shown by [9] that, while the learning might be quicker, the final performance is independent of the domain order chosen. In the second phase of its lifetime, starting after 15000 dialogues, a new MDP emerges every 1000 dialogues due to user demand changes, where we set $\sigma = 0.5$. The lifetime ends once 25000 dialogues are reached. The high number of 25000 dialogues in total allows us to observe a variety of user demands.

B. Model and Algorithms

To test the benefits of lifetime and meta-gradient RL, we use PPO [40] and CLEAR [10] as base algorithms. PPO is an on-policy actor-critic algorithm that has been widely adopted in the RL literature as well as real applications (e.g. training InstructGPT [41]). While not directly built for continual learning, it can adapt to changes and serves as a strong baseline algorithm. In contrast, CLEAR builds upon the off-policy algorithm V-trace [26], which increases its sample efficiency. CLEAR is specifically built for continual learning and has an additional regularisation loss that prevents forgetting by regularising the policy towards past predictions. In addition, both PPO and CLEAR have an entropy loss that facilitates exploration. When using meta-gradient RL, we meta-learn the entropy weight β_{ent} for both PPO and CLEAR and additionally the regularisation weight β_{reg} for CLEAR. A summary is shown in Table I and in-depth descriptions can be found in Appendix. We use four different setups:

1) **epi:** optimise for episodic return, i.e. use A_ψ^{epi}.
2) **life:** optimise for lifetime return, i.e. use A_ψ^{life}.
3) **epi+life:** optimise for episodic and lifetime return, i.e. use $A_\psi^{\text{epi}} + A_\psi^{\text{life}}$.
4) **epi+life+meta:** use $A_\psi^{\text{epi}} + A_\psi^{\text{life}}$ and meta-learn hyperparameters.

TABLE I
Two Algorithms Which We Equip With Lifetime Return Objective and Meta-Gradient RL

Algorithm	type	Meta-parameters
PPO	on-policy	$\eta = \{\beta_{\text{ent}}\}$
CLEAR	off-policy	$\eta = \{\beta_{\text{ent}}, \beta_{\text{reg}}\}$

Regarding the model architecture, we employ the state-of-the-art model for continual dialogue policy learning called DDPT [9]. DDPT has been shown to have strong forward transfer capabilities to new domains, prevents forgetting, and can adapt to new domains without architectural changes. We note that optimising DDPT with episodic return (setup 1) coincides with the method used in [9]. We chose DDPT, which is specifically built for continual dialogue policy learning, instead of the AdapterCL architecture proposed in [8] for the following reasons. AdapterCL was proposed in the continual learning setup where only a single domain occurs in dialogues for a period of time. Each new domain leads to random initialisation of a corresponding adapter and freezing of all other adapters. This mechanism focuses on preventing forgetting but impedes forward and backward transfer to other domains that we regard as essential in continual learning. Moreover, the AdapterCL architecture grows in size with the introduction of new domains and it is unclear how a variable number of adapters can be used in a single dialogue turn, which is required in multi-domain dialogues (e.g. if the system wants to talk about hotels and restaurants in one turn). However, since our proposal is of algorithmic nature, we believe that our method leads to improvements regardless of the specific architecture chosen.

C. Performance Measure

The objective of the dialogue policy is to provide information to the user as well as book entities if required. An entity can

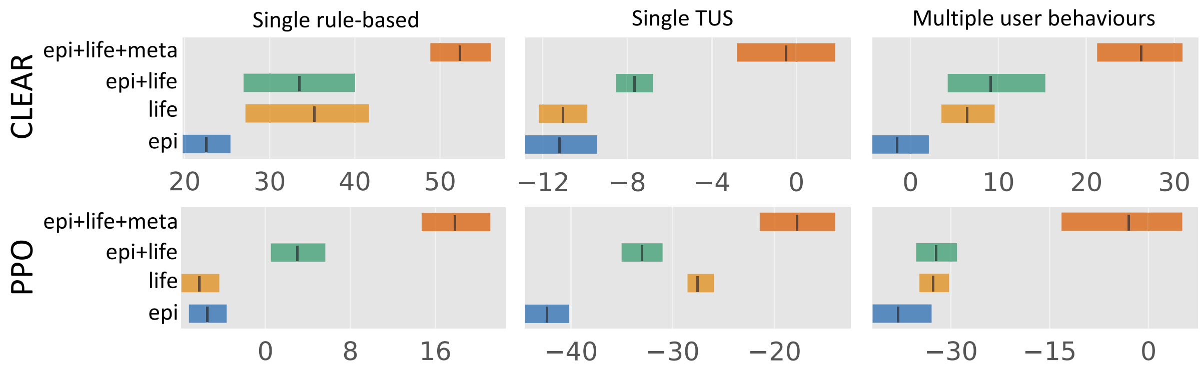


Fig. 2. Average lifetime return for different simulator setups and algorithms. We equipped the algorithms CLEAR and PPO with lifetime return optimisation (epi+life) as well as meta-learning hyperparameters (epi+life+meta). For comparison, we add an ablated version that uses lifetime return without episodic return (life). The results show the mean over 5 seeds with 95% confidence intervals.

be for instance a hotel, restaurant, train, or taxi. A dialogue is considered successful if the user has 1) obtained all information required given the correct constraints and, 2) the correct entities are booked (if necessary). The constraints of the user can be for instance given by [*Restaurant:* [*area: north*, *food-type: Italian*, *day: Saturday*]]. The general performance of a dialogue policy is measured by success rate and return. We do not report other measurements such as booking rate or complete rate since they do not fully evaluate whether the user goal was achieved. The maximum number of turns is set to $T_{\max} = 40$ [34]. The reward in every turn is given by -1 to maximise dialogue efficiency, and a final reward of $2 \cdot T_{\max} = 80$ or $-T_{\max} = -40$ is given in case of dialogue success or failure, respectively. Our main measurement will be lifetime performance, i.e. the averaged sum of return or success over the lifetime, as we want to evaluate which algorithm performs best over the entire lifetime. We use mean with 95% confidence intervals when averaging measures across seeds [42]. In addition, we also showcase the moving average to analyse how algorithms perform locally during changes of the environment as done in [25]. The time-window we choose is 500 dialogues.

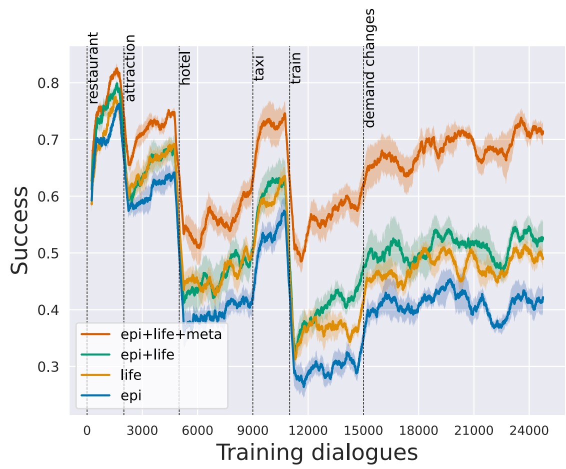


Fig. 3. Moving average (and standard error) with time-window of 500 dialogues while interacting with multiple simulators and CLEAR as base algorithm. New domains will be introduced into the agent's life at the vertical lines. After 15000 dialogues, the user demand distribution p_{demand} changes every 1000 dialogues.

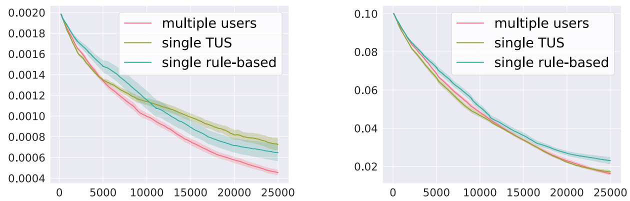
V. Experiment Analysis

A. Algorithm Performance

Lifetime performance: Fig. 2 depicts the results for the single and multiple user behaviour setup. We can see that optimising for the present and future (epi+life) compares favourably to the baseline, achieving better performance in all settings. The substitution of episodic return by lifetime return (life) does improve performance in four out of six setups, only performing similarly to the baseline for PPO and the rule-based simulator and CLEAR and the transformer-based simulator. In addition, the combination of episodic return and lifetime return (epi+life) performs equally well or better than only using lifetime return, except for PPO and the transformer-based simulator. This shows that the introduction of lifetime return optimisation makes learning more robust in changing environments as the infinite horizon is taken into account. Moreover, we can observe that the combination of returns and meta-learning (epi+life+meta) leads to the best performance for both algorithms and all three simulation setups. This emphasises the importance of dynamic adjustments during lifelong learning. The results together show the substantial benefit of our proposals in a challenging continual dialogue policy learning scenario. Lastly, CLEAR generally performs better than PPO since it is specifically built for continual learning. We will therefore focus on CLEAR for the remaining experiments.

Local performance: Lifetime performance gives us a general measure of the overall performance. To obtain better insights into the local learning process, we depict moving average success rate in Fig. 3. The vertical lines with a domain next to it indicate when this domain is introduced. We can observe a drop in performance for all algorithms once a new domain is introduced, where the drop is less severe for the meta-learned algorithm due to the dynamic adaptations. The models do not experience a drop when the taxi domain is introduced as DDPT has strong zero-shot capabilities for this domain [9]. All algorithms adapt to the changed environment as can be seen from the rising trends. The user demands start changing after 15000 dialogues and we can observe a stronger upwards trend for the meta-learned algorithm until the end of the lifetime. The algorithms using

Fig. 4. Moving average success (and standard error) with time-window of 500 dialogues while interacting with the high initiative rule-based simulator and CLEAR as base algorithm. The dialogue agents face changing user demands with $\sigma = 0.5$ every 1000 dialogues and all domains are introduced from the beginning.



(a) β_{ent} (b) β_{reg}

Fig. 5. Meta-learned parameters β_{ent} and β_{reg} during the lifetime of the meta-learned agent CLEAR with epi+life+meta.

lifetime return but no meta-learning perform similarly to the baseline at the beginning when only restaurant is present, since this defines a stationary environment. Once the MDP changes (starting with the introduction of attraction), lifetime return maximisation shows its benefit for changing environments. We can also observe that the local performance is the highest for restaurant and lower once more domains are introduced. One reason for that is the fact that once a second domain is introduced, the agent additionally faces multi-domain dialogues that are more difficult to fulfill. Our proposed RECORD framework is the only simulation environment that takes multi-domain dialogues during continual learning into account.

User demand changes: Fig. 3 shows that incorporating lifetime objective and meta-learning outperforms the baseline during user demand changes. However, models that perform better until 15000 conducted dialogues have a head start when being exposed to changing user demands. In order to strictly analyse how well these models deal with changing user demands, we conducted additional experiments where all domains are already introduced from the beginning and only the user demands change every 1000 dialogues with $\sigma = 0.5$ as before and $p_{\text{num}}(1) = 0.5, p_{\text{num}}(2) = 0.3, p_{\text{num}}(3) = 0.2$ to obtain a slightly easier setup. We chose an easier setup in terms of the probability distribution p_{num} because the introduction of all domains from the beginning in combination with changing demands is very challenging. Fig. 4 depicts the moving average success. We can see that the usage of lifetime return outperforms the baseline by taking future changes into account. In addition, the usage of lifetime return and meta-learning produces the best results due to additional adaptation during the learning process. This shows that taking future changes into account is beneficial when being exposed to changing user demands even when all models start from the same initial performance. We show more experimental results with higher values of σ in Appendix.

The results overall show that lifetime return is beneficial for the continual learning environment of dialogue, while requiring only minimal algorithmic changes, which motivates the broader usage of lifetime return in general. Moreover, the substantial performance improvements when meta-learning hyperparameters emphasise the strength of self-adaptive algorithms in continual dialogue policy learning and encourage future applications.

B. How Do Meta-Parameters Adapt?

In the following we want to analyse how the meta-learned entropy weight β_{ent} and regularisation weight β_{reg} for the CLEAR policy evolve during the learning process. Fig. 5 shows that both parameters are decreasing over the course of learning. The entropy weight decreases as the requirement to explore decreases over the course of the lifetime. The regularisation weight decreases almost linearly as the agent collects more experience during its lifetime and starts stabilising at the end. This shows that regularisation on past behaviour is beneficial for learning and decreases as more knowledge has been collected. This is reasonable as domains reoccur and past knowledge hence benefits the future. The adaptability is critical for the decision of how much prevention of forgetting is necessary, depending on the future circumstances.

We note that while it might be possible to characterise the change in hyperparameters in retrospect, for example with an exponential function as in Fig. 5, it is not clear whether this function is also helpful in prospect. Namely, as we have no knowledge of future circumstances, it is not clear that this function would be applicable to any other setup in the future. This emphasizes the flexibility and strength of meta-learning albeit for a higher computational cost.

C. Human Trial

To verify that our proposal of lifetime return and meta-learning for continual dialogue policy learning outperforms the baseline also in interaction with humans, we ran a human trial using Amazon Mechanical Turk [43], where human volunteers directly interacted with the systems. We used CLEAR as base algorithm and compared the baseline (epi) with our advancements (epi+life) and (epi+life+meta), which were trained in the multiple user behaviour setup. After each conducted dialogue the user was asked to rate the performance of the systems with a number between 1 and 5. We collected 75 dialogues for each system. The results are given in Table II, showing clearly that our models outperform the baseline.

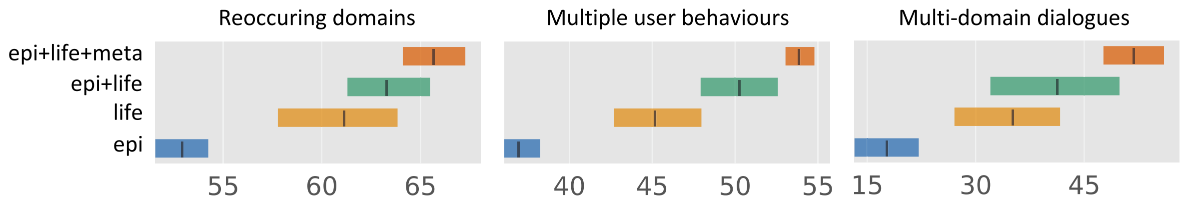


Fig. 6. Average lifetime return for different RECORD ablations. We use the previous setup in continual learning where single domain dialogues are considered and only the newest introduced domain is present until another domain change occurs. We then add either reoccuring domains, multiple user behaviours, or multi-domain dialogues to this setup. The setups that use only one simulator use the high initiative rule-based simulator.

TABLE II
HUMAN TRIAL RESULTS FOR THE BASELINE (EPI) AND OUR PROPOSED METHODS (EPI+LIFE) AND (EPI+LIFE+META) WITH CLEAR AS BASE ALGORITHM

Algorithm	How did the system perform?
epi	3.42
epi+life	3.79
epi+life+meta	3.94

Hired humans interacted with the policies and were asked to rate the performance afterwards, given by a number between 1 and 5. Our proposals are significantly better than the baseline ($p < 0.05$).

D. RECORD Ablations

RECORD allows us to model reoccuring domains, multiple user behaviours, multi-domain dialogues, and user demand changes. In this section, we want to analyse the scenarios that emerge through adding one of the above proposals to the previous setup in continual learning, where the newly introduced domain is the only domain in the user goal until another domain change occurs [8], [9]. We create three scenarios by adding one of the following to the previous setup:

- **reocurring domains** realised using the rule-based simulator with high initiative and $p_{\text{new}} = 0.8, p_{\text{num}}(1) = 1.0$.
- **multiple user behaviours** realised with three user simulators and $p_{\text{new}} = p_{\text{num}}(1) = 1.0$.
- **multi-domain dialogues** realised using the rule-based simulator with high initiative and $p_{\text{new}} = 1.0, p_{\text{num}}(1) = 0.3, p_{\text{num}}(2) = 0.5, p_{\text{num}}(3) = 0.2$.

We omit user demand changes in this experiment as this is already covered in Section V-A. The results are shown in Fig. 6, confirming our previous results that lifetime return aids in learning and additional meta-learning leads to the best performance. The modelling of reoccuring domains in isolation leads to the highest returns in the three setups since single domain dialogues with the rule-based simulator is the easiest setup, where the reoccurance of domains result in a higher proportion of already existing domains in the user goal that are potentially easier to solve for the model. The modelling of multi-domain dialogues leads to the lowest returns out of the three scenarios due to the increased difficulty of achieving multiple domain goals within a single dialogue.

VI. Related Work

A. Continual Learning Setup in Dialogue

In terms of simulating reoccuring tasks, our framework shows similarities with other works in supervised continual learning [24], [44]. However, these frameworks do not explicitly model circumstances that are specific to dialogue, such as how to combine single tasks to form multi-domain dialogues or changing user demands over time. [20] modelled changing user demands by sine curves of different amplitude and frequency. In contrast, we leverage normal distributions based on an uninformed prior over tasks, which gives more diverse and non-periodic changes.

Previous setups in continual learning for task-oriented dialogue have focused on learning a set of domains in a sequential order, where each domain is only observed for a certain period of time [7], [8], [9], [45]. The goal then is to maximise performance on all seen domains. While suitable for studying the effects of catastrophic forgetting, this setup constitutes an unrealistic scenario for modelling how a dialogue system would learn in the actual world: it is unlikely that a dialogue system provides aid for domains only for a certain time period until it starts learning a new domain. Even if that is true, it is contradictory that the dialogue system should retain performance on previous domains if these are never seen again. Moreover, the setup does not model the possibility of multi-domain dialogues [39] that are required in the real world, e.g. finding a flight, train and hotel for a journey in a single conversation. There has been no learning environment that simulates the challenges of a dialogue system in a real-world continual learning scenario.

B. Lifetime Return

RL algorithms in episodic environments aim to maximise the episodic return, i.e. the discounted sum of rewards until the end of the episode. Lifetime return looks further into the future until the end of the lifetime, taking an infinite future horizon into account [12]. [46] meta-learn an intrinsic reward function with the goal of maximising lifetime return of the RL agent using multiple lifetimes in parallel. Lifetime return has been previously only used for proof-of-concept environments with less than 10 actions to showcase its use in training an intrinsic reward function but not for more complex tasks such as multi-domain dialogue [12], [46]. We are the first to use lifetime return for dialogue, which defines an environment with large state and action spaces [33]. Moreover, we are the first to explore the optimisation of both episodic and lifetime return.

C. Meta-Gradient RL

Meta-gradient RL has been used for meta-learning various components such as hyperparameters [13], [15], [28] or update targets [15]. To the best of our knowledge, our work is the first to apply it in the domain of dialogue policy learning. [13] study meta-gradient reinforcement learning in a non-stationary grid-world, where the grid-world exhibits a small state and action space. The design of non-stationarity is targeted at showing the merits of meta-gradient reinforcement learning, where the optimal policy is required to be re-learned after every MDP change. In contrast, we apply meta-gradient reinforcement learning to continual learning, where previously learned tasks are supposed to facilitate learning of a new task and re-learning of a task would imply catastrophic forgetting that is prohibitive in continual learning. Moreover, we apply meta-gradient reinforcement learning to a setup with much larger state and action spaces as well as optimise the meta-parameters towards lifetime return maximisation.

VII. Discussion

Our RECORD framework strives for a more realistic approximation of how a dialogue system could interact in a real world. Modelling the circumstances in the real world exactly is a difficult endeavour due to its multitude of unpredictable situations such as crises, pandemics, sudden novel inventions, etc. We thus emphasize that our goal is not to simulate exactly what may be happening in the real world, but aim to abstract and simulate the (most important) challenges that need to be overcome by a continually learning agent. In this respect, we consider RECORD as a strong foundation to build upon. An interesting extension of RECORD is to model the fact that reward might not be always accessible in every dialogue, opening up research in continual reward learning for dialogue.

Regarding our experimental setup, our study on continual RL in ever-changing dialogue circumstances was conducted with five domains of the MultiWOZ environment [39]. We chose this environment due to its unique combination of multi-domain dialogues, multiple user simulators, a database, and a success evaluation component required for our research [34]. In addition, the relatively small number of domains in MultiWOZ allowed us to conduct a large variety of scenarios with many different RECORD settings. Nevertheless, considering the extensive array of potential domains in dialogue, the logical progression involves expanding our investigation to encompass a broader spectrum of domains. To this end, the domains of MultiWOZ can be extended by using additional domains of other datasets such as SGD [47]. Only recently, a database, success evaluator as well as a rule-based user simulator has been built for the SGD dataset [48]. We regard a large scale continual learning study using our RECORD framework as an exciting next step.

VIII. Conclusion

In this work, we address the problem of dialogue systems learning in a world that is continuously changing. To provide an experimental environment that models the most important challenges during the lifetime of a dialogue policy, we proposed a flexible and controllable learning environment called RECORD that models multiple user behaviours, multi-domain dialogues, introduction of new domains and changing user demands. Instead of solely optimising the policy for episodic return, which only addresses the present, we proposed the usage of lifetime return as an additional objective for taking the future into account, i.e. learning with an open horizon. Additionally, we employed meta-gradient RL to enable dynamic hyperparameter adaptation for continual dialogue policy learning. Extensive experiments with single and multiple user simulators and two algorithms show that our proposed algorithmic advancements of lifetime return and meta-gradient RL results in more robust continual RL in a setup that models important challenges in real world scenarios. It moreover encourages widespread usage of lifetime return and meta-gradient RL for continual learning scenarios. For future work, we investigate different learning strategies to improve performance in our RECORD framework such as hierarchical lifetime learning and episodic memory.

Appendix
Open Horizon CLEAR

CLEAR [10] is a continual RL algorithm that builds upon the off-policy actor-critic algorithm V-trace [26]. It hence learns both a policy π_θ parameterised with θ and a critic V_ψ parameterised with ψ, where the critic approximates the state-value function for π. Given a trajectory $\tau = (s_t, a_t, r_t)_{t=k}^{t=k+n}$ generated by a behaviour policy μ, the n-steps V-trace target v_k and advantage A is defined by

$$v_k = V_\psi(s_k) + \sum_{t=k}^{k+n-1} \left(\prod_{j=k}^{t} \gamma_{k+j}\right)\left(\prod_{i=k}^{t-1} c_i\right)\delta_t \quad (8)$$

$$A(s_k, a_k) = r_k + \gamma_k v_{k+1} - V_\psi(s_k), \quad (9)$$

where we used transition-based discounting for unifying the episodic and non-episodic case as introduced in (1). Here, $\delta_t = \rho_t(r_t + \gamma_t V_\psi(s_{t+1}) - V_\psi(s_t))$ is a temporal difference term. Moreover, $\rho_t = \min(\overline{\rho}, \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)})$ and $c_t = \min(\overline{c}, \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)})$ are truncated importance sampling weights, where $\overline{\rho}$ and $\overline{c}$ are hyperparameters.

To view the environment as both episodic and non-episodic (Section III-B), we require a critic V_ψ^{epi} that estimates episodic return and a critic V_ψ^{life} that estimates the lifetime return. Let v_k^{epi} and A^{epi} denote v_k and A in (8) and 9 with $\gamma_{k+j} = 0$ if s_{k+j} is a terminal state, i.e. the episodic case. Let v_k^{life} and A^{life} denote v_k and A in (8) and 9 with $\gamma_{k+j} = \gamma_c > 0$ for all states s_{k+j}, i.e. the non-episodic case. Moreover, let $V_{\text{old}}^{\text{epi}}(s_k)$ be the prediction of V_ψ^{epi} at the time of creating the state s_k. We define the following losses:

$$\mathcal{L}_v^{\text{epi}} = (V_\psi^{\text{epi}}(s_k) - v_k^{\text{epi}})^2 \quad (10)$$

TABLE III
DIFFERENT LOSSES OF OPEN HORIZON CLEAR AND WHETHER THEY USE RECENT OR OLD EXPERIENCE FROM THE BUFFER

	$\mathcal{L}_v^{\text{epi}}$	$\mathcal{L}_v^{\text{life}}$	$\mathcal{L}_v^{\text{reg}}$	$\mathcal{L}_\pi^{\text{epi}}$	$\mathcal{L}_\pi^{\text{life}}$	$\mathcal{L}_\pi^{\text{ent}}$	$\mathcal{L}_\pi^{\text{reg}}$
$\mathcal{B}_{\text{old}}$	✓	x	✓	✓	x	✓	✓
$\mathcal{B}_{\text{rec}}$	✓	✓	x	✓	✓	✓	x

$$\mathcal{L}_v^{\text{life}} = (V_\psi^{\text{life}}(s_k) - v_k^{\text{life}})^2 \tag{11}$$

$$\mathcal{L}_v^{\text{reg}} = (V_\psi^{\text{epi}}(s_k) - V_{\text{old}}^{\text{epi}}(s_k))^2 \tag{12}$$

The episodic critic V_ψ^{epi} is optimised by minimising the sum $\mathcal{L}_v^{\text{epi}} + \mathcal{L}_v^{\text{reg}}$, whereas the lifetime critic is optimised by minimising $\mathcal{L}_v^{\text{life}}$. The regularisation loss $\mathcal{L}_v^{\text{reg}}$ is needed to prevent catastrophic forgetting by regularising towards the old critic predictions.

For the policy updates, we define the following losses:

$$\mathcal{L}_\pi^{\text{epi}} = -\rho_k \cdot A^{\text{epi}}(s_k, a_k) \cdot \log \pi_\theta(a_k \mid s_k) \tag{13}$$

$$\mathcal{L}_\pi^{\text{life}} = -\rho_k \cdot A^{\text{life}}(s_k, a_k) \cdot \log \pi_\theta(a_k \mid s_k) \tag{14}$$

$$\mathcal{L}_\pi^{\text{ent}} = \sum_a \pi_\theta(a \mid s_k) \cdot \log \pi_\theta(a \mid s_k) \tag{15}$$

$$\mathcal{L}_\pi^{\text{reg}} = \sum_a \mu(a \mid s_k) \cdot \log \frac{\mu(a \mid s_k)}{\pi_\theta(a \mid s_k)} \tag{16}$$

The loss $\mathcal{L}_\pi^{\text{epi}}$ is used to optimise the episodic return whereas $\mathcal{L}_\pi^{\text{life}}$ is used to optimise the lifetime return. The entropy loss $\mathcal{L}_\pi^{\text{ent}}$ is used to maximise the entropy and thus prevent premature convergence. $\mathcal{L}_\pi^{\text{reg}}$ forces π to imitate the past behaviour of μ to prevent catastrophic forgetting. For meta-gradient RL we define the inner and the outer loss as

$$\mathcal{L}_\pi^{\text{inner}}(\theta, \beta_{\text{ent}}, \beta_{\text{reg}}) = \mathcal{L}_\pi^{\text{epi}} + \mathcal{L}_\pi^{\text{life}} + \beta_{\text{ent}}\mathcal{L}_\pi^{\text{ent}} + \beta_{\text{reg}}\mathcal{L}_\pi^{\text{reg}} \tag{17}$$

$$\mathcal{L}_\pi^{\text{outer}}(\theta) = \mathcal{L}_\pi^{\text{life}} \tag{18}$$

where $\eta = \{\beta_{\text{ent}}, \beta_{\text{reg}}\}$ are the hyperparameters that we optimise during the meta-learning step. We perform m inner loop updates using $\mathcal{L}_\pi^{\text{inner}}$ before a meta-learning update using $\mathcal{L}_\pi^{\text{outer}}$ is performed.

In practice, CLEAR stores the generated episodes during interaction in a replay buffer $\mathcal{B}$. In our case, in every update step, we retrieve the n_{rec} most recent episodes $\mathcal{B}_{\text{rec}}$ from $\mathcal{B}$ as well as sample n_{old} episodes from $\mathcal{B} \setminus \mathcal{B}_{\text{rec}}$. The retrieved trajectories are then used for calculating the defined losses, where some losses only use experience from $\mathcal{B}_{\text{rec}}$ or $\mathcal{B}_{\text{old}}$ as shown in Table III. We only use recent experience for the update of V_ψ^{life} as we want to approximate how much lifetime return we can expect from the current time step onwards. The full algorithm is depicted in Algorithm 3.

Note we can retrieve the original CLEAR algorithm by removing the lifetime critic V_ψ^{life} as well as the loss $\mathcal{L}_\pi^{\text{life}}$ in (17) and performing only the inner loop updates.

OPEN HORIZON PPO

PPO is an on-policy actor-critic algorithm and hence learns both a policy π_θ and a critic V_ψ. Given a trajectory $\tau = (s_t, a_t, r_t)_{t=k}^{t=k+n}$ generated by π, we define the one-step target v_k and generalized advantage A by

$$v_k = r_k + \gamma_k V_\psi(s_{k+1}) \tag{19}$$

$$A(s_k, a_k) = \sum_{t=k}^{k+n-1} (\gamma_t \lambda)^{t-k} \delta_t \tag{20}$$

where $\delta_t = r_t + \gamma_t V_\psi(s_{t+1}) - V_\psi(s_t)$ is the temporal difference term. We used transition-based discounting for unifying the episodic and non-episodic case as introduced in (1). To view the environment as both episodic and non-episodic (Section III-B), we require a critic V_ψ^{epi} that estimates episodic return and a critic V_ψ^{life} that estimates the lifetime return. Let v_k^{epi} and A^{epi} denote v_k and A in (19) and 20 with $\gamma_{k+j} = 0$ if s_{k+j} is a terminal state, i.e. the episodic case. Let v_k^{life} and A^{epi} denote v_k and A in (19) and 20 with $\gamma_{k+j} = \gamma_c > 0$ for all states s_{k+j}, i.e. the non-episodic case. We define the following losses:

$$\mathcal{L}_v^{\text{epi}} = (V_\psi^{\text{epi}}(s_k) - v_k^{\text{epi}})^2 \tag{21}$$

$$\mathcal{L}_v^{\text{life}} = (V_\psi^{\text{life}}(s_k) - v_k^{\text{life}})^2 \tag{22}$$

The episodic critic V_ψ^{epi} then minimises $\mathcal{L}_v^{\text{epi}}$, whereas the lifetime critic minimises $\mathcal{L}_v^{\text{life}}$.

Regarding the policy updates, let $h_\theta(s,a) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}$, where θ_{old} is the vector of policy parameters before we perform a policy update. We define the losses

$$\mathcal{L}_\pi^{\text{epi}} = \min(h_\theta(s_k, a_k) \cdot A^{\text{epi}}(s_k, a_k)),$$
$$\text{clip}(h_\theta(s_k, a_k), 1-\epsilon, 1+\epsilon) \cdot A^{\text{epi}}(s_k, a_k)) \tag{23}$$

$$\mathcal{L}_\pi^{\text{life}} = \min(h_\theta(s_k, a_k) \cdot A^{\text{life}}(s_k, a_k)),$$
$$\text{clip}(h_\theta(s_k, a_k), 1-\epsilon, 1+\epsilon) \cdot A^{\text{life}}(s_k, a_k)) \tag{24}$$

$$\mathcal{L}_\pi^{\text{ent}} = \sum_a \pi_\theta(a \mid s_k) \cdot \log \pi_\theta(a \mid s_k), \tag{25}$$

where ϵ denotes a hyperparameter and $\text{clip}(h, \text{lower}, \text{upper})$ clips the real value h if it exceeds the lower or upper bound. The loss $\mathcal{L}_\pi^{\text{epi}}$ is used to optimise the episodic return whereas $\mathcal{L}_\pi^{\text{life}}$ is used to optimise the lifetime return. The entropy loss $\mathcal{L}_\pi^{\text{ent}}$ is used to maximise the entropy and thus prevent premature convergence. For meta-gradient RL we define the inner and outer loss as

$$\mathcal{L}_\pi^{\text{inner}}(\theta, \beta_{\text{ent}}) = \mathcal{L}_\pi^{\text{epi}} + \mathcal{L}_\pi^{\text{life}} + \beta_{\text{ent}}\mathcal{L}_\pi^{\text{ent}} \tag{26}$$

$$\mathcal{L}_\pi^{\text{outer}}(\theta) = \mathcal{L}_\pi^{\text{life}} \tag{27}$$

where $\eta = \{\beta_{\text{ent}}\}$ is the hyperparameter that we optimise during the meta-learning step. We perform m inner loop updates using $\mathcal{L}_\pi^{\text{inner}}$ before a meta-learning update using $\mathcal{L}_\pi^{\text{outer}}$ is performed.

In practice, our PPO implementation generates n_{episodes} episodes using policy $\pi_{\theta_{\text{old}}}$ and updates the critics and policy for K epochs with the generated data. The full algorithm is depicted in Algorithm 4.

Algorithm 3: Open Horizon CLEAR.

Require: policy π_θ, episodic critic V_ψ^{epi}, lifetime critic V_ψ^{life}, buffer $\mathcal{B}$, meta-frequency m, meta-parameters $\eta = \{\beta_{\text{ent}}, \beta_{\text{reg}}\}$, dialogues_until_update, n_{rec}, n_{old}, user set U

num_updates $\leftarrow 0$
num_dialogues $\leftarrow 0$
while lifetime not ended **do**
  goal $\leftarrow$ Algorithm 1 $\triangleright$ get a user goal
  user $\sim \mathcal{U}(U)$ $\triangleright$ uniformly sample a user
  $\tau = (s_0, a_0, r_0, \ldots, s_T) \sim (\pi, \text{user}, \text{goal})$ $\triangleright$ create an episode
  $\mathcal{B}$.append(τ) $\triangleright$ save experience in buffer
  num_dialogues$+= 1$
  if num_dialogues mod dialogues_until_update $= 0$ **then**
    $\mathcal{B}_{\text{rec}} \leftarrow \mathcal{B}[-n_{\text{rec}} :]$ $\triangleright$ get most recent dialogues from $\mathcal{B}$
    $\mathcal{B}_{\text{old}} \leftarrow \text{RandomSample}(\mathcal{B} \setminus \mathcal{B}_{\text{rec}}, n_{\text{old}})$ $\triangleright$ sample n_{old} dialogues from $\mathcal{B} \setminus \mathcal{B}_{\text{rec}}$
    Update V_ψ^{epi} to minimise $\mathcal{L}_v^{\text{epi}} + \mathcal{L}_v^{\text{reg}}$ with (10) and 12 using $\mathcal{B}_{\text{rec}}$ and $\mathcal{B}_{\text{old}}$
    Update V_ψ^{life} to minimise $\mathcal{L}_v^{\text{life}}$ with (11) using $\mathcal{B}_{\text{rec}}$
    Update π_θ to minimise $\mathcal{L}_\pi^{\text{inner}}$ with (17) using $\mathcal{B}_{\text{rec}}$ and $\mathcal{B}_{\text{old}}$ $\triangleright$ Optimise for episodic and lifetime return
    num_updates $+= 1$
  if num_updates mod $m = 0$ **then** $\triangleright$ perform a meta update
    Update $\beta_{\text{reg}}, \beta_{\text{ent}}$ to minimise $\mathcal{L}_\pi^{\text{outer}}$ with (18) using $\mathcal{B}_{\text{rec}}$ $\triangleright$ Optimise meta parameters for lifetime return

Algorithm 4: Open Horizon PPO.

Require: policy π_θ, episodic critic V_ψ^{epi}, lifetime critic V_ψ^{life}, meta-frequency m, meta-parameters $\eta = \{\beta_{\text{ent}}\}$, n_{episodes}, user set U

while lifetime not ended **do**
  $\mathcal{B} \leftarrow []$
  for dialogue $= 1, 2, \ldots, n_{\text{episodes}}$ **do**
    goal $\leftarrow$ Algorithm 1 $\triangleright$ get a user goal
    user $\sim \mathcal{U}(U)$ $\triangleright$ uniformly sample a user
    $\tau = (s_0, a_0, r_0, \ldots, s_T) \sim (\pi_{\text{old}}, \text{user}, \text{goal})$ $\triangleright$ create an episode
    $\mathcal{B}$.append(τ)
  for $k = 1, \ldots, K$ $\triangleright$ Run multiple epochs **do**
    Update V_ψ^{epi} to minimise $\mathcal{L}_v^{\text{epi}}$ using (21)
    Update V_ψ^{life} to minimise $\mathcal{L}_v^{\text{life}}$ using (22)
    Update π_θ to minimise $\mathcal{L}_\pi^{\text{inner}}$ using (26) $\triangleright$ Optimise for episodic and lifetime return
    if k mod $m = 0$ **then** $\triangleright$ perform a meta update
      Update β_{ent} to minimise $\mathcal{L}_\pi^{\text{outer}}$ using (27) $\triangleright$ Optimise meta parameter for lifetime return
    $\theta_{\text{old}} \leftarrow \theta$
    $\mathcal{B} \leftarrow []$

Note we can retrieve the original PPO algorithm by removing the lifetime critic V_ψ^{life} as well as the loss $\mathcal{L}_\pi^{\text{life}}$ in (26) and performing only the inner loop updates.

Detailed Algorithm Specification

For the configurations of the DDPT model and CLEAR algorithm, we use the same specification as given in [9]: for the policy we use an input size and hidden size of 128 in both transformer encoder and decoder. We use two heads for the encoder and decoder, 4 transformer layers for the encoder and 2 for the decoder. The critics for lifetime and episodic return have the same architecture as the transformer encoder, obtaining the same input as the policy module plus an additional CLS vector (as in RoBERTa). The output of the CLS vector is fed into a linear layer to obtain the critic prediction.

For CLEAR, we use a learning rate of 0.001 for meta-learning the hyperparameters and $\beta_{\text{reg}} = 0.1$ and $\beta_{\text{ent}} = 0.01$ for the regularization and entropy loss weights. When meta-learning, we perform $m = 4$ update steps using the inner loss function before doing one meta-update using the outer loss. We generate two dialogues before performing an update. We sample $n_{\text{rec}} = n_{\text{old}} = 32$ when retrieving experience from the buffer. The replay buffer size is set to 5000 and uses first-in first-out once the buffer reached its capacity. For the V-trace algorithm, the parameters $\bar{\rho}$ and $\bar{c}$ are set to 1.0.

(a) $\sigma = 1.0$ (b) $\sigma = 2.0$

Fig. 7. Moving average success (and standard error) with time-window of 500 dialogues while interacting with the rule-based simulator and CLEAR as base algorithm. The dialogues agents face changing user demands every 1000 dialogues and all domains are introduced from the beginning.

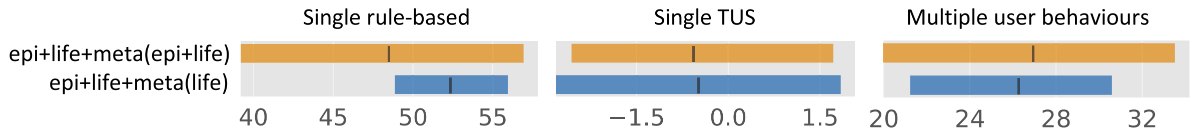


Fig. 8. Lifetime return for different simulator setups. We compare two different outer losses for optimizing the metaparameters.

For PPO we use learning rate of 0.1 for meta-learning the hyperparameters and $\beta_{\text{ent}} = 0.01$ for the entropy loss weights. The number of epochs is set to $K = 5$. When meta-learning, we perform $m = 5$ update steps using the inner loss function before doing one meta-update using the outer loss. For one PPO epoch we generate $n_{\text{episodes}} = 100$ dialogues. We use $\lambda = 0.0$ and $\epsilon = 0.2$ for the remaining PPO parameters.

We use the ADAM optimiser [49] with a learning rate of 5e-5 and 1e-4 for policy and critic modules, respectively. We used a single NVIDIA A100 GPU for our experiments. The experiments lasted around 10-15 h per lifetime, depending on the algorithm and user simulator used. Our code was implemented using PyTorch==1.10.2.

Results with Different User Demand Setups

We here show results with different levels of user demand changes given through different values of σ. We show average success in a time-window of 500 for policies interacting with the rule-based simulator of high initiative. Fig. 7 shows performance for $\sigma = 1.0$ and $\sigma = 2.0$. We can observe that our proposals outperform the baseline even in these highly varying circumstances.

Meta-Learning Ablation

In Section III-B we proposed to use the lifetime return objective for meta-learning hyperparameters of the algorithm, which meant using A_{ψ}^{life} in (7). We ran additional experiments using

$$\mathcal{L}_{\pi}^{\text{outer}} = \mathcal{L}_{\pi}^{\text{epi}} + \mathcal{L}_{\pi}^{\text{life}} \tag{28}$$

for optimising the meta-parameters in CLEAR, which we termed epi+life+meta(epi+life). The losses $\mathcal{L}_{\pi}^{\text{epi}}$ and $\mathcal{L}_{\pi}^{\text{life}}$ are given in (13) and 14. We denote the usage of the outer loss as in (18) as epi+life+meta(life). Fig. 8 shows the results, where we can observe no statistical difference between the two methods. The experimental setup is the same as explained in Section IV.

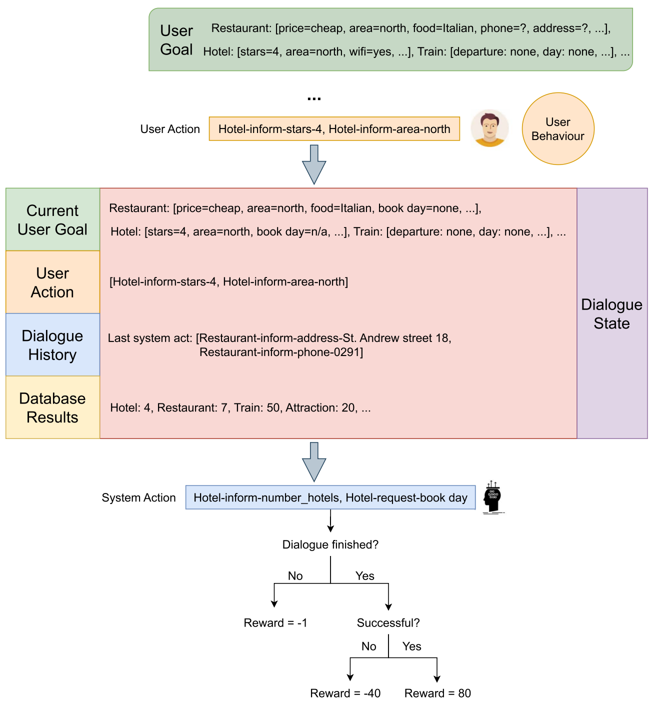


Fig. 9. Example for user action, dialogue state information, system action and reward calculation. The user action selection depends on the user behaviour and the user goal.

References

[1] E. Levin and R. Pieraccini, "A stochastic model of computer-human interaction for learning dialogue strategies," in *Proc. EuroSpeech*, 1997, pp. 1883–1886.

[2] N. Roy, J. Pineau, and S. Thrun, "Spoken dialogue management using probabilistic reasoning," in *Proc. 38th Annu. Meeting Assoc. Comput. Linguistics*, 2000, pp. 93–100, doi: 10.3115/1075218.1075231.

[3] J. D. Williams and S. Young, "Partially observable Markov decision processes for spoken dialog systems," *Comput. Speech Lang.*, vol. 21, no. 2, pp. 393–422, Apr. 2007, doi: 101016/j.csl.2006.06.008.

[4] Z. Zhang, R. Takanobu, M. Huang, and X. Zhu, "Recent advances and challenges in task-oriented dialog system," *Sci. China Technol. Scis.*, vol. 63, no. 10, pp. 2011–2027, 2003.

[5] R. Hadsell, D. Rao, A. Rusu, and R. Pascanu, "Embracing change: Continual learning in deep neural networks," *Trends Cogn. Sci.*, vol. 24, pp. 1028–1040, Dec. 2020.

[6] M. Biesialska, K. Biesialska, and M. R. Costa-jussà, "Continual lifelong learning in natural language processing: A survey," in *Proc. 28th Int. Conf. Comput. Linguistics*, 2020, pp. 6523–6541.

[7] F. Mi, L. Chen, M. Zhao, M. Huang, and B. Faltings, "Continual learning for natural language generation in task-oriented dialog systems," in *Proc. Findings Assoc. Comput. Linguistics: EMNLP*, 2020, pp. 3461–3474, [Online]. Available: https://aclanthology.org/2020.findings-emnlp.310

[8] A. Madotto et al., "Continual learning in task-oriented dialogue systems," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2021, pp. 7452–7467, [Online]. Available: https://aclanthology.org/2021.emnlp-main.590

[9] C. Geishauser et al., "Dynamic dialogue policy for continual reinforcement learning," in *Proc. 29th Int. Conf. Comput. Linguistics*, 2022, pp. 266–284, [Online]. Available: https://aclanthology.org/2022.coling-1.21

[10] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Paper.pdf

[11] S. Powers, E. Xing, E. Kolve, R. Mottaghi, and A. Gupta, "CORA: Benchmarks, baselines, and metrics as a platform for continual reinforcement learning agents," in *Proc. Conf. Lifelong Learn. Agents*, 2022, pp. 705–743.

[12] S. Singh, R. L. Lewis, A. G. Barto, and J. Sorg, "Intrinsically motivated reinforcement learning: An evolutionary perspective," *IEEE Trans. Auton. Ment. Develop.*, vol. 2, no. 2, pp. 70–82, Jun. 2010.

[13] J. Luketina, S. Flennerhag, Y. Schroecker, D. Abel, T. Zahavy, and S. Singh, "Meta-gradients in non-stationary environments," in *Proc. Workshop Agent Learn. Open-Endedness*, 2022, pp. 886–901. [Online]. Available: https://openreview.net/forum?id=SlzBXwZIZ9

[14] J. Parker-Holder et al., "Automated reinforcement learning (autorl): A survey and open problems," *J. Artif. Intell. Res.*, vol. 74, 517–568, 2022, [Online]. Available: https://arxiv.org/abs/2201.03916

[15] Z. Xu, H. P. v. Hasselt, and D. Silver, "Meta-gradient reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/2715518c875999308842e3455eda2fe3-Paper.pdf

[16] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[17] M. White, "Unifying task specification in reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 3742–3750, [Online]. Available: http://proceedings.mlr.press/v70/white17a.html

[18] K. Khetarpal, M. Riemer, I. Rish, and D. Precup, "Towards continual reinforcement learning: A review and perspectives," *J. Artif. Intell. Res.*, vol. 75, pp. 1401–1476, Dec. 2022.

[19] E. Lecarpentier and E. Rachelson, "Non-stationary Markov decision processes, a worst-case approach using model-based reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/859b00aec8885efc83d1541b52a1220d-Paper.pdf

[20] Y. Chandak, G. Theocharous, S. Shankar, M. White, S. Mahadevan, and P. S. Thomas, "Optimizing for the future in non-stationary mdps," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 1414–1425.

[21] S. Padakandla, P. K. J, and S. Bhatnagar, "Reinforcement learning in non-stationary environments," *Appl. Intell.*, vol. 50, no. 11, pp. 3590–3606, 2020.

[22] M. Wołczyk, M. Zajac, R. Pascanu, L. Kuciński, and P. Miłoś, "Continual world: A robotic benchmark for continual reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 28496–28510, [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/ef8446f35513a8d6aa2308357a268a7e-Paper.pdf

[23] D. Lopez-Paz and M. A. Ranzato, "Gradient episodic memory for continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/f87522788a2be2d171666752f97ddebb-Paper.pdf

[24] M. Caccia et al., "Online fast adaptation and knowledge accumulation (osaka): A new approach to continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 16532–16545. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/c0a271bc0ecb776a094786474322cb82-Paper.pdf

[25] E. A. Platanios, A. Saparov, and T. M. Mitchell, "Jelly bean world: A testbed for never-ending learning," in *Proc. 8th Int. Conf. Learn. Representations*, 2020, OpenReview.net. [Online]. Available: https://openreview.net/forum?id=Byx_YAVYPH

[26] L. Espeholt et al., "IMPALA: Scalable distributed Deep-RL with importance weighted actor-learner architectures," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 1406–1415, [Online]. Available: http://proceedings.mlr.press/v80/espeholt18a.html

[27] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. 4th Int. Conf. Learn. Representations*, 2016.

[28] T. Zahavy et al., "A self-tuning actor-critic algorithm," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 20913–20924, [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/f02208a057804ee16ac72ff4d3cec53b-Paper.pdf

[29] S. Young, J. Schatzmann, K. Weilhammer, and H. Ye, "The hidden information state approach to dialog management," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2007, vol. 4, pp. IV–149–IV–152.

[30] Q. Zhu et al., "ConvLab-2: An open-source toolkit for building, evaluating, and diagnosing dialogue systems," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics: Syst. Demonstrations*, 2020, pp. 142–149, [Online]. Available: https://aclanthology.org/2020.acl-demos.19

[31] Y. Xu, C. Zhu, B. Peng, and M. Zeng, "Meta dialogue policy learning," 2006, *arXiv:2006.02588*.

[32] S. Young, M. Gašić, B. Thomson, and J. D. Williams, "POMDP-based statistical spoken dialog systems: A review," *Proc. IEEE*, vol. 101, no. 5, pp. 1160–1179, May 2013.

[33] W.-C. Kwan, H. Wang, H. Wang, and K.-F. Wong, "A survey on recent advances and challenges in reinforcement learning methods for task-oriented dialogue policy learning," *Mach. Intell. Res.*, vol. 20, no. 3, pp. 318–334, 2023.

[34] Q. Zhu et al., "Convlab-3: A flexible dialogue system toolkit based on a unified data format," in *Proc. Conf. Empirical Methods Natural Lang. Process. Syst. Demonstrations*, 2022, pp. 106–123.

[35] B. Peng et al., "Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 2231–2240, [Online]. Available: https://www.aclweb.org/anthology/D17-1237

[36] D. Tang, X. Li, J. Gao, C. Wang, L. Li, and T. Jebara, "Subgoal discovery for hierarchical dialogue policy learning," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 2298–2309, [Online]. Available: https://aclanthology.org/D18-1253

[37] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young, "Agenda-based user simulation for bootstrapping a POMDP dialogue system," in *Proc. Hum. Lang. Technol.: Conf. North Amer. Chapter Assoc. Comput. Linguistics; Companion Volume, Short Papers*, 2007, pp. 149–152, [Online]. Available: https://aclanthology.org/N07-2038

[38] H.-C. Lin et al., "Domain-independent user simulation with transformers for task-oriented dialogue systems," in *Proc. 22nd Annu. Meeting Special Int. Group Discourse Dialogue*, 2021, pp. 445–456, [Online]. Available: https://aclanthology.org/2021.sigdial-1.47

[39] P. Budzianowski et al., "Multi-WOZ - A large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 5016–5026, [Online]. Available: https://aclanthology.org/D18-1547

[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[41] L. Ouyang et al., "Training language models to follow instructions with human feedback," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 27730–27744.

[42] R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare, "Deep reinforcement learning at the edge of the statistical precipice," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 29304–29320. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/f514cec81cb148559cf475e7426eed5e-Paper.pdf

[43] K. Crowston, "Amazon mechanical turk: A research tool for organizations and information systems scholars," in *Proc. Shaping Future ICT Res. Methods Approaches*, 2012, pp. 210–221.

[44] H. Koh, D. Kim, J.-W. Ha, and J. Choi, "Online continual learning on class incremental blurry task configuration with anytime inference," in *Proc. Int. Conf. Learn. Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=nrGGfMbY_qK
[45] B. Geng, F. Yuan, Q. Xu, Y. Shen, R. Xu, and M. Yang, "Continual learning for task-oriented dialogue system with iterative network pruning, expanding and masking," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process. (Volume 2: Short Papers)*, 2021, pp. 517–523, [Online]. Available: https://aclanthology.org/2021.acl-short.66
[46] Z. Zheng et al., "What can learned intrinsic rewards capture?," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 11436–11446, [Online]. Available: https://proceedings.mlr.press/v119/zheng20b.html
[47] A. Rastogi, X. Zang, S. Sunkara, R. Gupta, and P. Khaitan, "Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 5, pp. 8689–8696. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6394
[48] W.-C. Kwan et al., "JoTR: A. joint transformer and reinforcement learning framework for dialog policy learning," 2023, *arXiv:2309.00230*.
[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015.

Christian Geishauser received the B.Sc. and M.Sc. degrees in mathematics from the Faculty of Mathematics at Ludwig Maximilians University, Munich, Germany, in 2015 and 2018, respectively. He is currently working toward the Ph.D. degree with the Dialog Systems and Machine Learning Group led by Prof. Milica Gašić at Heinrich Heine University Düsseldorf, Düsseldorf, Germany. His research interests include continual reinforcement learning, reward learning and meta learning applied to task-oriented dialogue systems.

Carel van Niekerk received the B.Sc. degree in actuarial and financial mathematics and the second B.Sc. (Hons.) degree in mathematical statistics and M.Sc. degree in mathematical statistics (with distinction) from the University of Pretoria, South Africa, in 2015, 2016, and 2018, respectively. Thereafter, he applied his knowledge as an AI Consultant at NGA Risksecure, from 2018 to 2019. He is currently engaged in the Towards Intelligent Dialogue Systems Project as a Ph.D. Candidate within the Dialogue Systems and Machine Learning Group, Heinrich Heine University, Dusseldorf, Germany, under the supervision of Prof. Milica Gašić. He also co-authored the 2020 SIGDial Best Paper, TripPy. His current research interests primarily involve the development of intelligent dialogue systems, with a specific focus on managing ambiguity and uncertainty in human conversation.

Nurul Lubis received the B.Eng. degree (*cum laude*) from the Bandung Institute of Technology, Bandung, Indonesia, in 2014, and the M.Eng. and Dr.Eng. degrees from the Nara Institute of Science and Technology (NAIST), Nara, Japan, in 2017 and 2019, respectively. She was the recipient of the NAIST Best Student Award in 2019, Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT) Scholarship from 2014 to 2019. She was a Research Intern with Honda Research Institute Japan, Company. Ltd., Saitama, Japan, and is currently a Postdoctoral Researcher with the Heinrich Heine University Düsseldorf, Düsseldorf, Germany. Her research interests include emotion in spoken language, affective dialogue systems, and dialogue policy optimisation with reinforcement learning and variational methods.

Hsien-chin Lin was born in Taipei, Taiwan, in 1993. He received the B.S.E. degree majoring in electrical engineering and the M.S. degree majoring in communication engineering from National Taiwan University, Taipei, Taiwan, in 2015 and 2017, respectively. He is currently working toward the Ph.D. degree with the Dialogue Systems and Machine Learning Group, Heinrich-Heine-Universität Düsseldorf, Düsseldorf, Germany. His research interests include natural language generation, user simulation, and dialogue system.

Michael Heck received the Dipl.-Inf. degree from the Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, in 2012, and the D.Eng. degree from Nara Institute of Science and Technology (NAIST), Nara, Japan, in 2018. Between 2013 and 2015, he was a Research Assistant with the Interactive Systems Labs at KIT, where he was involved in several international programs including Quaero, Babel and EU-BRIDGE. He was a Research Intern with IBM Research, Tokyo, Japan, in 2017. He was a Technical Staff at RIKEN Center for Advanced Intelligence Project, Nara, in 2019. He is currently a Postdoctoral Researcher with Heinrich Heine University Düsseldorf, Düsseldorf, Germany. He was the recipient of the Baden-Württemberg Scholarship and was supported by the NAIST Global Initiatives Program in 2012. His research interests include unsupervised learning, task-oriented dialogue systems, and representation learning. He was the recipient of the SIGDial Best Paper Award in 2020.

Shutong Feng received the B.A. and M.Eng. degrees from the Department of Engineering, University of Cambridge, Cambridge, U.K., in 2019. He is currently working toward the Ph.D. degree with Dialog Systems and Machine Learning Group, Heinrich Heine University Düsseldorf, Düsseldorf, Germany, under the supervision of Prof. Milica Gašić and Dr. Nurul Lubis. His research interests focuses on emotions in task-oriented dialogue systems.

Benjamin Ruppik received the B.Sc. and M.Sc. degrees in mathematics from the Faculty of Mathematics and Natural Sciences, University of Bonn, Bonn, Germany, in 2015 and 2018, respectively, and the Ph.D. degree in low-dimensional topology from the Max-Planck Institute for Mathematics, Bonn, Germany, in 2022. He is currently a Postdoc with the Dialog Systems and Machine Learning lab led by Prof. Milica Gašić at Heinrich-Heine-University Düsseldorf, Düsseldorf, Germany. His research interests include topological deep learning for natural language processing and task-oriented dialogue systems.

Renato Vukovic received the B.Sc. and M.Sc. degrees from the Faculty of Mathematics and Natural Sciences, Heinrich Heine University Düsseldorf, Düsseldorf, Germany, in 2019 and 2022, respectively. He is currently working toward the Ph.D. degree with the Dialog Systems and Machine Learning Group, Heinrich Heine University Düsseldor. He was a Research Assistant with the Dialog Systems and Machine Learning Group as a Graduate Student. His research interests include ontology construction and knowledge management for task-oriented dialogue systems.

Milica Gašić (Senior Member, IEEE) received the B.Sc. degree in computer science and mathematics from the University of Belgrade, Belgrade, Serbia, in 2006, the M.Phil. degree in computer speech, text and internet technology and the Ph.D. degree in engineering from the University of Cambridge, Cambridge, U.K. in 2007 and 2011, respectively. She is curently a Professor iwth Heinrich-Heine-University Düsseldorf, Germany, where she holds a Chair for Dialog Systems and Machine Learning. Dr. Gasic works in the intersection of machine learning and natural language processing with a particular interest in conversational AI. She has coauthored around 70 journal articles and peer reviewed conference papers. She is currently the Vice President of SIGDIAL and a Member of the Scientific Advisory Board of DFKI.

Chapter 8

Conclusions and Future Work

In this chapter, we give a summary of the contributions of this thesis and point out possible directions to continue. The objective of the research presented here was to provide more sample efficient learning as well as investigate continual reinforcement learning for dialogue policies.

8.1 Results

In Chapter 5, we addressed the problem of sample inefficiency in dialogue policy learning due to the sparse reward given by task success or failure. We proposed an intrinsic reward function called *information gain* that aids learning through more informative feedback, which encourages the policy to gather information about the user goal. Our empirical study showed that information gain leads to faster learning and better final policies. It moreover leads to policies that asks the user for information in cases where it is necessary, which also gives a better user experience.

In Chapter 6, we introduced the first study of continual reinforcement learning for dialogue policies. We provided metrics that measure forward transfer to new domains and catastrophic forgetting of already observed domains. Moreover, we proposed a novel architecture called *dynamic dialogue policy transformer* (DDPT) which is specifically designed for continual RL. DDPT addresses the continual RL challenges in dialogue that arise from the introduction of new domains, namely growing state and action spaces and forward transfer. DDPT achieves this by leveraging an encoder-decoder Transformer for processing dynamic input and output, a pre-trained language model for state information and action embeddings, effective state space reduction by hard-attention masking and a novel domain gate for fast forward transfer to new domains. Our empirical study revealed that DDPT achieves significant forward transfer to new domains and prevents catastrophic forgetting of already observed domains. This is obtained without any growth in neural network parameter size and thus fulfills important points in the continual learning desiderata.

In Chapter 7, we challenged the common continual learning setup for dialogue where domains are only seen for a certain period of time. We proposed *realistic environments for continual RL of dialogue policies* (RECORD), a generalized, controllable, and flexible continual learning framework that aims to reflect the most important challenges in realistic continual learning for dialogue policies. RECORD models multi-domain dialogues, changing user demands over time, multiple user behaviors, and reoccuring domains. In order to learn in challenging continual learning environments, we proposed the usage lifetime return for more robust learning and meta-gradient RL for optimizing the hyperparameters of the underlying RL algorithm. We conducted a multitude of experiments with different user behaviors and continual learning setups that proved the flexibility and controllability of RECORD and superiority of leveraging lifetime return and meta-gradient RL.

Our results warrant a more widespread usage of the proposed ideas. The underlying idea of information gain to reward behavior that gathers information or resolve uncertainty is general and applicable to any task where information needs to be retrieved from the interlocutor. Similarly, the usage of lifetime return as an additional objective for maximization can be utilized for any reinforcement learning in episodic environments to potentially increase the stability and improve the performance. Lastly, since our ideas in DDPT were inspired by the way an operator would explain the task, we hope that they transfer to different setups as well (we give an example for language models in the following section).

8.2 Future Directions

The field of continual RL for dialogue policy optimization is still in its infancy and far from being solved. Due to the large number of challenges that a task-oriented dialogue system encounters in its lifetime, there are many possible future directions.

Large Scale Continual Learning Our study on continual RL was conducted with five domains of the MultiWOZ environment (Budzianowski et al., 2018). We chose this environment due to its unique combination of multi-domain dialogues, a database, and a success evaluation component required for our research. This was sufficient for the first investigations of continual RL for dialogue policies. However, considering the extensive array of potential domains, the logical progression involves expanding our investigation to encompass a broader spectrum of domains. To this end, the domains of MultiWOZ can be extended by using additional domains of other datasets such as SGD or Task-Master (Byrne et al., 2021; Rastogi et al., 2020). The challenge in this expansion lies in the creation of suitable databases and success evaluation mechanisms for these supplementary datasets. We regard this as a short-term future direction.

Learning to Adapt Continual learning requires constant adaptation to new circumstances. In our study, we used CLEAR (Rolnick et al., 2019) that leverages recent experience for adaptation of the neural network parameters and meta-gradient RL (Xu et al., 2018) for adaptation of the hyperparameters. An interesting next step is to investigate further methods for fast adaptation in our proposed RECORD framework. To this end, promising methods are given by exploration (Amin et al., 2021; Ladosz et al., 2022), where explorative actions can be chosen in the right situations to find better solutions, which is especially important if a new domain is introduced. Another line of work that aids fast adaptation is episodic memory (Fortunato et al., 2019; Pritzel et al., 2017) which is inspired by the hippocampus in the brain (Lengyel and Dayan, 2007). Episodic memory is an additional module that can be updated quickly, in contrast to the dialogue policy that has to be updated based on gradient descent.

Reward Learning Within our RECORD framework, we make the assumption that rewards are consistently provided by the environment, namely the user. Once a dialogue system is deployed in the real world, it might be undesirable to request feedback from the user after every conversation. Consequently, an expansion of the RECORD framework could encompass the consideration of scenarios where rewards are not observed during certain conversations. This extension paves the way for exploring methods for intrinsic reward learning. Intuitively, a system that has gained enough experience should be able to build its own intrinsic reward function. Possible methods for investigation include meta-learning intrinsic rewards (Zheng et al., 2020, 2018) or imitation learning on sub-optimal demonstrations (Zhu et al., 2022c). In that realm, our proposal information gain (Chapter 5) can be potentially utilized as additional regularization objective for the learnable reward function in order to enhance training stability as information gain directly aids in learning how to solve the task at hand.

Large Language Models Large language models (LLMs) such as ChatGPT or Llama (Touvron et al., 2023) have disrupted the field of dialogue systems due to its human-likeness and general purpose capabilities. The DDPT architecture in Chapter 6 was inspired by the way an operator would explain and act upon a novel task, namely, relying on descriptions and focusing on the necessary information. This idea is directly applicable to LLMs: we filter out unnecessary information of the state as in DDPT, incorporate the descriptions of every remaining information into the LLM input (instead of embedding it via a language model) and provide descriptions for every action. We can furthermore simulate the domain gate of DDPT by asking the LLM to first specify the domain to talk about, which subsequently reduces the amount of action descriptions we need to provide. This shows in particular that the ideas presented for DDPT are more generally applicable and furthermore defines a natural next step for investigation. ChatGPT has been used only recently as a dialogue policy to generate semantic actions (Kwan et al., 2023) and achieved a success rate of around 70% on a rule-based simulator for MultiWOZ (Zhu et al., 2020), showing its potential to dialogue policy applications.

The utilization of LLMs becomes particularly interesting within the framework of continual RL for dialogue policies. While DDPT demonstrated substantial knowledge transfer to novel, but related, domains, achieving this becomes challenging when new domains significantly differ from what has already been encountered. Given the versatile capabilities of LLMs, they can assist in selecting actions during the early stages after a novel domain has been introduced, where the dialogue policy performance is not yet sufficient. When leveraging off-policy learning, such as in the CLEAR algorithm, the dialogue policy can directly learn from the experiences generated by the LLM, eventually surpassing the LLM's initial zero-shot performance. This is an interesting symbiosis between LLMs with general-purpose capabilities and specialized task-specific experts that can be efficiently trained.

8.2.1 The Future of Task-oriented Dialogue

Dialogue systems require interpretability, controllability and flexibility. LLMs excel at a wide range of tasks with impressive zero-shot capabilities, where their natural language interface and open-domain knowledge enable great flexibility. The components in a task-oriented dialogue system, on the other hand, are limited to their pre-defined, yet expandable, ontology. This reduced flexibility comes with the advantage of being interpretable and controllable, as for instance specific actions can be forbidden depending on the situation at hand. In addition, these components are typically smaller and hence meet memory requirements as well as being more efficient during training and testing. Moreover, current studies showed that LLMs can not outperform expert components in a task-oriented dialogue system that were trained on task-specific data such as dialogue state trackers or dialogue policies (Heck et al., 2023; Hudeček and Dusek, 2023; Kwan et al., 2023). There will be most likely a symbiosis between the two approaches. LLMs are a great option in cases where the modular system is uncertain about how to act (due to a new domain for instance) or when the user provides an out-of-ontology query. Moreover, LLMs can act as synthetic data generators for more lightweight models (Li et al., 2023; Rosenbaum et al., 2022). The modular dialogue system, on the other hand, can excel at fine-grained tasks due to being an efficiently trained task-specific expert.

Regardless of the future composition of a dialogue system, the unresolved issue remains regarding the definition of an optimal reward function for training such a system. It is hypothesized that reward is enough to learn behavior that exhibits most if not all of intelligent abilities, such as learning, perception, social intelligence, language, etc. (Silver et al., 2021). Humans, for instance, may implicitly optimize their behavior for the survival of the species and reproduction, which is nevertheless challenging to optimize in practice. It is thus more plausible that humans optimize a different, practical reward function leading to similar results. The crucial question arises as to whether there exists a comparable reward for dialogue systems to optimize and what form it might take. This reward function must be in some form interpretable, ensuring the predictability of the resulting agent's behavior, which is essential for safety considerations. Furthermore, the necessity of a single reward function in practice versus the requirement for multiple rewards remains unclear. As previously explained, existing dialogue policies are trained based on task success, a sparse and stringent signal that assesses decisions within individual dialogues. Currently, large language models (LLMs) are trained using human preferences (Ziegler et al., 2019), yielding impressive outcomes (Ouyang et al., 2022). However, the reward models derived from these preferences often exhibit imperfections, giving rise to significant challenges, including human bias, misgeneralization, and potential reward hacking, where the model learns undesirable behavior (Casper et al., 2023). The definition of an optimal reward function, which potentially changes throughout continual learning, is still an open research area in task-oriented dialogue and beyond.

In the realm of continual learning, humans are able to robustly adapt to new circumstances often with only a few trials, while deep RL needs thousands of interactions for achieving the same task (Barreto et al., 2020). The challenge remains unresolved as to how this human-like adaptability can be instilled in artificial intelligence, maintaining plasticity in continual learning and determining the essential learning components. Furthermore, numerous studies on continual learning predominantly address the issue of catastrophic forgetting, which is essential to prevent. Nevertheless, forgetting over time might be a future requirement that offers potential advantages such as releasing capacity for upcoming tasks or facilitating the removal of undesired behavior or outdated knowledge. Related to this is the problem of how specific knowledge, facts, or behavior can be changed in a neural network without affecting the remaining parts (Jang et al., 2022; Sinitsin et al., 2020). The future bears many open problems and challenges for task-oriented dialogue systems in order to achieve human-like abilities. Nevertheless, the recent advancements in reinforcement learning and large language models promise a thriving future to come.

Appendix A

Supplementary Proofs

A.1 Deep Learning

Theorem 2. *For $i \in \mathbb{Z}$, let the positional encoding $\boldsymbol{p}^{(i)} \in \mathbb{R}^d$ be defined as*

$$\boldsymbol{p}^{(i)} = [\sin(w_0 i), \cos(w_0 i), ..., \sin(w_k i), \cos(w_k i), ..., \sin(w_{d/2-1} i), \cos(w_{d/2-1} i)], \tag{A.1}$$

$$w_k = \frac{1}{10000^{2k/d}} \tag{A.2}$$

The norm of the positional encoding $p^{(i)} \in \mathbb{R}^d$ is given by

$$||p^{(i)}||_2 = \sqrt{\frac{d}{2}}. \tag{A.3}$$

Moreover, the distance between two positional encodings is symmetric, i.e. for every $n \in \mathbb{N}$, it holds that

$$||p^{(i+n)} - p^{(i)}||_2 = ||p^{(i-n)} - p^{(i)}||_2. \tag{A.4}$$

Proof. Since $\sin^2(x) + \cos^2(x) = 1$, we have

$$||\boldsymbol{p}^{(i)}||_2 = \sqrt{\sin^2(w_0 \cdot i) + \cos^2(w_0 \cdot i) + ... + \sin^2(w_{\frac{d}{2}-1} \cdot i) + \cos^2(w_{\frac{d}{2}-1} \cdot i)} \tag{A.5}$$

$$= \sqrt{\frac{d}{2}}. \tag{A.6}$$

In order to prove the symmetry, we use the addition theorem $\cos(x-y) = \cos(x)\cos(y) + \sin(x)\sin(y)$ and the symmetry property $\cos(x) = \cos(-x)$. We calculate

$$||p^{(i+n)} - p^{(i)}||_2^2 \tag{A.7}$$

$$= ||p^{(i)}||_2^2 + ||p^{(i+n)}||_2^2 - 2\sum_{k=0}^{d/2-1} [\sin(w_k i)\sin(w_k(i+n)) + \cos(w_k i)\cos(w_k(i+n))] \tag{A.8}$$

$$= d - 2\sum_{k=0}^{d/2-1} [\cos(w_k i - w_k(i+n))] \tag{A.9}$$

$$= d - 2\sum_{k=0}^{d/2-1} [\cos(-w_k i + w_k(i+n))] \tag{A.10}$$

$$= d - 2 \sum_{k=0}^{d/2-1} [\cos(w_k i - w_k(i-n))] \tag{A.11}$$

$$= d - 2 \sum_{k=0}^{d/2-1} [\sin(w_k i)\sin(w_k(i-n)) + \cos(w_k i)\cos(w_k(i-n))] \tag{A.12}$$

$$= ||p^{(i-n)} - p^{(i)}||_2^2 \tag{A.13}$$

□

Theorem 3. *For any $n, i \in \mathbb{Z}$, there exists a matrix $\boldsymbol{M}^{(n)}$, which only depends on n but not on i, such that*

$$\boldsymbol{p}^{(i+n)} = \boldsymbol{M}^{(n)}\boldsymbol{p}^{(i)}. \tag{A.14}$$

Proof. According to the addition theorems of sin and cos, we obtain for every w_k

$$\sin(w_k(i+n)) = \sin(w_k i)\cdot\cos(w_k n) + \sin(w_k n)\cdot\cos(w_k i), \tag{A.15}$$
$$\cos(w_k(i+n)) = \cos(w_k i)\cdot\cos(w_k n) - \sin(w_k n)\cdot\sin(w_k i), \tag{A.16}$$

which can be rewritten in matrix notation as

$$\begin{bmatrix}\sin(w_k(i+n))\\ \cos(w_k(i+n))\end{bmatrix} = \begin{bmatrix}\cos(w_k n) & \sin(w_k n)\\ -\sin(w_k n) & \cos(w_k n)\end{bmatrix}\begin{bmatrix}\sin(w_k i)\\ \cos(w_k i)\end{bmatrix} \tag{A.17}$$

If we define

$$\boldsymbol{\Phi}_k^{(n)} = \begin{bmatrix}\cos(w_k n) & \sin(w_k n)\\ -\sin(w_k n) & \cos(w_k n)\end{bmatrix} \tag{A.18}$$

the identity in Equation A.14 follows for the matrix

$$\boldsymbol{M}^{(n)} = \begin{bmatrix}\boldsymbol{\Phi}_1^{(n)} & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\Phi}_2^{(n)} & \cdots & \mathbf{0}\\ \cdots & \cdots & \cdots & \cdots\\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\Phi}_{\frac{d}{2}}^{(n)}\end{bmatrix} \tag{A.19}$$

□

A.2 Reinforcement Learning

In this section, we give a proof for the policy improvement theorem (Section 3.4.1) and the policy gradient theorem (see Theorem 1) using the Performance Difference Lemma. For states $s_0, s \in \mathcal{S}$ and action $a \in \mathcal{A}$, we first define the discounted occupancy measure as

$$d^{\pi}(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t p_{\pi}(S_t = s), \tag{A.20}$$

$$d^{\pi}(s,a) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t p_{\pi}(S_t = s, A_t = a) = d^{\pi}(s)\cdot\pi(a|s), \tag{A.21}$$

$$d^{\pi}_{s_0}(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t p_{\pi}(S_t = s|S_0 = s_0), \tag{A.22}$$

$$d^{\pi}_{s_0}(s,a) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t p_{\pi}(S_t = s, A_t = a|S_0 = s_0) = d^{\pi}_{s_0}(s)\cdot\pi(a|s), \tag{A.23}$$

where $p_{\pi}(S_t = s, A_t = a)$ is the probability that the pair (s,a) occurs in time-step t when following policy π. From the definitions it follows immediately that

$$d^{\pi}(s,a) = \sum_{s_0} p_0(s_0)\cdot d^{\pi}_{s_0}(s,a). \tag{A.24}$$

Theorem 4 (Performance Difference Lemma). *Let π and π' be two policies and $s_0 \in \mathcal{S}$. We then have*

$$V^{\pi}(s_0) - V^{\pi'}(s_0) = \frac{1}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}}[A^{\pi'}(s,a)] \tag{A.25}$$

Proof.

$$\begin{aligned}
V^{\pi}(s_0) - V^{\pi'}(s_0) &= \mathbb{E}_{\pi}[\sum_{t=0}^{\infty}\gamma^t r_t|S_0 = s] - V^{\pi'}(s_0) && \text{(A.26)}\\
&= \mathbb{E}_{\pi}[\sum_{t=0}^{\infty}\gamma^t (r_t + \gamma V^{\pi'}(S_{t+1}) - \gamma V^{\pi'}(S_{t+1})|S_0 = s] - V^{\pi'}(s_0) && \text{(A.27)}\\
&= \mathbb{E}_{\pi}[\sum_{t=0}^{\infty}\gamma^t (r_t + \gamma V^{\pi'}(S_{t+1}) - V^{\pi'}(S_t)|S_0 = s] && \text{(A.28)}\\
&= \mathbb{E}_{\pi}[\sum_{t=0}^{\infty}\gamma^t (\mathbb{E}[r_t + \gamma V^{\pi'}(S_{t+1})] - V^{\pi'}(S_t)|S_0 = s] && \text{(A.29)}\\
&= \mathbb{E}_{\pi}[\sum_{t=0}^{\infty}\gamma^t (Q^{\pi'}(S_t, A_t) - V^{\pi'}(S_t)|S_0 = s] && \text{(A.30)}\\
&= \mathbb{E}_{\pi}[\sum_{t=0}^{\infty}\gamma^t \cdot A^{\pi'}(S_t, A_t)|S_0 = s] && \text{(A.31)}\\
&= \sum_{\tau} p_{\pi}(\tau|S_0 = s)\sum_{t=0}^{\infty}\gamma^t \cdot A^{\pi'}(s_t, a_t) && \text{(A.32)}\\
&= \sum_{t=0}^{\infty}\sum_{\tau}\sum_{s,a} p_{\pi}(\tau, S_t = s, A_t = a|S_0 = s)\gamma^t \cdot A^{\pi'}(s,a) && \text{(A.33)}\\
&= \sum_{t=0}^{\infty}\sum_{s,a}\sum_{\tau} p_{\pi}(\tau, S_t = s, A_t = a|S_0 = s)\gamma^t \cdot A^{\pi'}(s,a) && \text{(A.34)}\\
&= \sum_{s,a}\sum_{t=0}^{\infty}\gamma^t p_{\pi}(S_t = s, A_t = a|S_0 = s)\cdot A^{\pi'}(s,a) && \text{(A.35)}\\
&= \frac{1}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}}[A^{\pi'}(s,a)] && \text{(A.36)}
\end{aligned}$$

□

Corollary 4.1. *Let $J(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R_t]$. For policies π and π', the following holds:*

$$J(\pi) - J(\pi') = \frac{1}{1-\gamma}\mathbb{E}_{s,a\sim d^\pi}[A^{\pi'}(s,a)] \tag{A.37}$$

Proof. The corollary follows from Theorem 4 since

$$J(\pi) - J(\pi') = \sum_{s_0} p_0(s_0) \cdot [V^\pi(s_0) - V^{\pi'}(s_0)] \tag{A.38}$$

$$= \sum_{s_0} p_0(s_0) \cdot \frac{1}{1-\gamma}\mathbb{E}_{s,a\sim d^\pi_{s_0}}[A^{\pi'}(s,a)] \tag{A.39}$$

$$= \frac{1}{1-\gamma}\mathbb{E}_{s,a\sim d^\pi}[A^{\pi'}(s,a)] \tag{A.40}$$

□

Theorem 5 (Policy Improvement Theorem)**.** *Given arbitrary policies π and π' such that*

$$\sum_a \pi'(a|s)Q^\pi(s,a) \geq V^\pi(s) \quad \forall s \in \mathcal{S}, \tag{A.41}$$

then π' is at least as good as π, i.e.

$$V^{\pi'} \geq V^\pi(s), \quad \forall s \in \mathcal{S}. \tag{A.42}$$

Proof. We first note that

$$\sum_a \pi'(a|s)A^\pi(s,a) = \sum_a \pi'(a|s)Q^\pi(s,a) - \sum_a \pi'(a|s)V^\pi(s) \tag{A.43}$$

$$= \sum_a \pi'(a|s)Q^\pi(s,a) - V^\pi(s) \geq 0 \tag{A.44}$$

by our assumption. Using Theorem 4, we can conclude that for every $s_0 \in \mathcal{S}$

$$V^{\pi'}(s_0) - V^\pi(s_0) = \frac{1}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi'}_{s_0}}[A^\pi(s,a)] \tag{A.45}$$

$$= \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi'}_{s_0}}[\sum_a \pi'(a|s)A^\pi(s,a)] \geq 0. \tag{A.46}$$

□

Corollary 5.1 (Policy Improvement Theorem)**.** *Let π be a policy and define*

$$\pi'(s) = \arg\max_a Q^\pi(s,a). \tag{A.47}$$

The policy $\pi'(s)$ is then at least as good as π, i.e.

$$V^{\pi'} \geq V^\pi(s), \quad \forall s \in \mathcal{S}. \tag{A.48}$$

Proof. We only need to prove the assumption of Theorem 5. We have for all $s \in \mathcal{S}$

$$\sum_a \pi'(a|s)Q^\pi(s,a) = Q^\pi(s, \arg\max_a Q^\pi(s,a)) = \max_a Q^\pi(s,a) \geq V^\pi(s) \tag{A.49}$$

since $V^\pi(s) = \sum_a \pi(a|s)Q^\pi(s,a) \leq \sum_a \pi(a|s) \max_a Q^\pi(s,a) = \max_a Q^\pi(s,a)$. □

Theorem 6 (Policy Gradient Theorem)**.** *Let π_θ be a policy parameterized by parameters θ. Let $J(\theta) = J(\pi_\theta) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t R_t]$. The gradient of $J(\theta)$ is given by*

$$(1-\gamma)\nabla_\theta J(\theta) = \mathbb{E}_{s,a\sim d^\pi}[\nabla_\theta \ln \pi_\theta \cdot A^{\pi_\theta}(s,a)] \tag{A.50}$$

Proof. By the definition of the gradient, we have

$$\begin{aligned}
(1-\gamma)\cdot\nabla_\theta J(\theta) &= (1-\gamma)\cdot \lim_{||h||\to 0} \frac{J(\theta+h)-J(\theta)}{||h||} && \text{(A.51)}\\
&= \lim_{||h||\to 0} \frac{1}{||h||}\mathbb{E}_{s,a\sim d^{\pi_{\theta+h}}}[A^{\pi_\theta}(s,a)] && \text{(A.52)}\\
&= \lim_{||h||\to 0} \frac{1}{||h||}\sum_{s,a} d^{\pi_{\theta+h}}(s)\cdot \pi_{\theta+h}(a|s)A^{\pi_\theta}(s,a) && \text{(A.53)}\\
&= \lim_{||h||\to 0} \frac{1}{||h||}\sum_{s,a} d^{\pi_{\theta+h}}(s)\cdot [\pi_{\theta+h}(a|s)-\pi_\theta(a|s)+\pi_\theta(a|s)]A^{\pi_\theta}(s,a) && \text{(A.54)}\\
&= \lim_{||h||\to 0} \sum_s d^{\pi_{\theta+h}}(s)\cdot \sum_a \frac{\pi_{\theta+h}(a|s)-\pi_\theta(a|s)}{||h||}A^{\pi_\theta}(s,a) + \pi_\theta(a|s)A^{\pi_\theta}(s,a) && \text{(A.55)}\\
&= \lim_{||h||\to 0} \sum_s d^{\pi_{\theta+h}}(s)\cdot \sum_a \frac{\pi_{\theta+h}(a|s)-\pi_\theta(a|s)}{||h||}A^{\pi_\theta}(s,a) && \text{(A.56)}\\
&= \sum_s \lim_{||h||\to 0} d^{\pi_{\theta+h}}(s)\cdot \sum_a \lim_{||h||\to 0} \frac{\pi_{\theta+h}(a|s)-\pi_\theta(a|s)}{||h||}A^{\pi_\theta}(s,a) && \text{(A.57)}\\
&= \sum_s d^{\pi_\theta}(s)\cdot \sum_a \nabla_\theta \pi_\theta \cdot A^{\pi_\theta}(s,a) && \text{(A.58)}\\
&= \sum_s d^{\pi_\theta}(s)\cdot \sum_a \pi_\theta(a|s)\nabla_\theta \ln \pi_\theta \cdot A^{\pi_\theta}(s,a) && \text{(A.59)}\\
&= \mathbb{E}_{s,a\sim d^{\pi_\theta}}[\nabla_\theta \ln \pi_\theta \cdot A^{\pi_\theta}(s,a)] && \text{(A.60)}
\end{aligned}$$

In Equation A.52 we used Corollary 4.1. Equation A.56 followed since

$$\sum_a \pi_\theta(a|s)A^{\pi_\theta}(s,a) = \sum_a \pi_\theta(a|s)[Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)] = V^{\pi_\theta}(s) - V^{\pi_\theta}(s) = 0. \tag{A.61}$$

Lastly, Equation A.58 followed since $d^{\pi_\theta}(s)$ is continuous in θ and hence

$$\lim_{||h||\to 0} d^{\pi_{\theta+h}}(s) = d^{\pi_\theta}(s). \tag{A.62}$$

□

Bibliography

Aljundi, Rahaf, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars (2018). "Memory Aware Synapses: Learning What (Not) to Forget". In: *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part III*. Munich, Germany: Springer-Verlag, 144–161. ISBN: 978-3-030-01218-2. DOI: 10.1007/978-3-030-01219-9_9. URL: https://doi.org/10.1007/978-3-030-01219-9_9.

Aljundi, Rahaf, Min Lin, Baptiste Goujaud, and Yoshua Bengio (2019). "Gradient based sample selection for online continual learning". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, pp. 11816–11825. URL: https://proceedings.neurips.cc/paper/2019/hash/e562cd9c0768d5464b64cf61da7fc6bb-Abstract.html.

Amin, Susan, Maziar Gomrokchi, Harsh Satija, Herke van Hoof, and Doina Precup (2021). "A Survey of Exploration Methods in Reinforcement Learning". In: *CoRR* abs/2109.00157. arXiv: 2109.00157. URL: https://arxiv.org/abs/2109.00157.

Atkinson, Craig, Brendan McCane, Lech Szymanski, and Anthony Robins (2021). "Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting". In: *Neurocomputing* 428, pp. 291–307. DOI: 10.1016/j.neucom.2020.11.050. URL: https://doi.org/10.1016%2Fj.neucom.2020.11.050.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton (2016). *Layer Normalization*. arXiv: 1607.06450 [stat.ML].

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1409.0473.

Barnett, Susan and Stephen Ceci (Aug. 2002). "When and Where Do We Apply What We Learn? A Taxonomy for Far Transfer". In: *Psychological bulletin* 128, pp. 612–37. DOI: 10.1037/0033-2909.128.4.612.

Barreto, André, Shaobo Hou, Diana Borsa, David Silver, and Doina Precup (2020). "Fast reinforcement learning with generalized policy updates". In: *Proceedings of the National Academy of Sciences* 117.48, pp. 30079–30087. DOI: 10.1073/pnas.1907370117. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1907370117. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1907370117.

Beck, Jacob, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson (2023). *A Survey of Meta-Reinforcement Learning*. arXiv: 2301.08028 [cs.LG].

Bellman, Richard (1958). "Dynamic Programming and Stochastic Control Processes". In: *Inf. Control.* 1.3, pp. 228–239. DOI: 10.1016/S0019-9958(58)80003-0. URL: https://doi.org/10.1016/S0019-9958(58)80003-0.

Bertsekas, Dimitri P. and John N. Tsitsiklis (1996). *Neuro-dynamic programming.* Vol. 3. Optimization and neural computation series. Athena Scientific, pp. I–XIII, 1–491. ISBN: 1886529108.

Bertsekas, Dimitri P. and John N. Tsitsiklis (1999). "Gradient Convergence in Gradient Methods with Errors". In: *SIAM J. on Optimization* 10.3, 627–642. ISSN: 1052-6234. DOI: 10.1137/S1052623497331063. URL: https://doi.org/10.1137/S1052623497331063.

Bremner, Andrew J., David J. Lewkowicz, and Charles Spence (June 2012). *Multisensory Development*. Oxford University Press. ISBN: 9780199586059. DOI: 10.1093/acprof:oso/9780199586059.001.0001. URL: https://doi.org/10.1093/acprof:oso/9780199586059.001.0001.

Budzianowski, Paweł, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić (2018). "MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 5016–5026. DOI: 10.18653/v1/D18-1547. URL: https://aclanthology.org/D18-1547.

Byrne, Bill, Karthik Krishnamoorthi, Saravanan Ganesh, and Mihir Kale (Aug. 2021). "TicketTalk: Toward human-level performance with end-to-end, transaction-based dialog systems". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, pp. 671–680. DOI: 10.18653/v1/2021.acl-long.55. URL: https://aclanthology.org/2021.acl-long.55.

Casanueva, Iñigo, Paweł Budzianowski, Pei-Hao Su, Stefan Ultes, Lina M. Rojas-Barahona, Bo-Hsiang Tseng, and Milica Gašić (June 2018). "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)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 714–719. DOI: 10.18653/v1/N18-2112. URL: https://www.aclweb.org/anthology/N18-2112.

Casper, Stephen et al. (2023). *Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback*. arXiv: 2307.15217 [cs.AI].

Chandak, Yash, Georgios Theocharous, Shiv Shankar, Martha White, Sridhar Mahadevan, and Philip Thomas (2020). "Optimizing for the Future in Non-Stationary MDPs". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 1414–1425. URL: https://proceedings.mlr.press/v119/chandak20a.html.

Chaudhry, Arslan, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet Kumar Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato (2019). "Continual Learning with Tiny Episodic Memories". In: *CoRR* abs/1902.10486. arXiv: 1902.10486. URL: http://arxiv.org/abs/1902.10486.

Chen, Lu, Bowen Tan, Sishan Long, and Kai Yu (Aug. 2018). "Structured Dialogue Policy with Graph Neural Networks". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, pp. 1257–1268. URL: https://www.aclweb.org/anthology/C18-1107.

Chen, Zhi, Lu Chen, Xiaoyuan Liu, and Kai Yu (2020). "Distributed Structured Actor-Critic Reinforcement Learning for Universal Dialogue Management". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28, pp. 2400–2411. DOI: 10.1109/TASLP.2020.3013392.

Cho, Kyunghyun, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio (Oct. 2014a). "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches". In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar: Association for Computational Linguistics, pp. 103–111. DOI: 10.3115/v1/W14-4012. URL: https://aclanthology.org/W14-4012.

Cho, Kyunghyun, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (Oct. 2014b). "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. URL: https://aclanthology.org/D14-1179.

Cover, Thomas M. and Joy A. Thomas (2006). *Elements of Information Theory 2nd Edition (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience. ISBN: 0471241954.

Daniels, Zachary Alan et al. (2022). "Model-Free Generative Replay for Lifelong Reinforcement Learning: Application to Starcraft-2". In: *Proceedings of The 1st Conference on Lifelong Learning Agents*. Ed. by Sarath Chandar, Razvan Pascanu, and Doina Precup. Vol. 199. Proceedings of Machine Learning Research. PMLR, pp. 1120–1145. URL: https://proceedings.mlr.press/v199/daniels22a.html.

De Lange, Matthias, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars (2022). "A Continual Learning Survey: Defying Forgetting in Classification Tasks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.7, pp. 3366–3385. DOI: 10.1109/TPAMI.2021.3057446.

Degris, Thomas, Martha White, and Richard S. Sutton (2012). "Off-Policy Actor-Critic". In: *Proceedings of the 29th International Coference on Machine Learning*. ICML'12. Edinburgh, Scotland: Omnipress, 179–186. ISBN: 9781450312851.

Duchi, John, Elad Hazan, and Yoram Singer (2011). "Adaptive subgradient methods for online learning and stochastic optimization". In: *Journal of Machine Learning Research* 12.Jul, pp. 2121–2159.

Espeholt, Lasse et al. (2018). "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 1406–1415. URL: http://proceedings.mlr.press/v80/espeholt18a.html.

Fortunato, Meire, Melissa Tan, Ryan Faulkner, Steven Hansen, Adrià Puigdomènech Badia, Gavin Buttimore, Charlie Deck, Joel Z. Leibo, and Charles Blundell (2019). "Generalization of Reinforcement Learners with Working and Episodic Memory". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, pp. 12448–12457. URL: https://proceedings.neurips.cc/paper/2019/hash/02ed812220b0705fabb868ddbf17ea20-Abstract.html.

Geishauser, C., S. Hu, H. Lin, N. Lubis, M. Heck, S. Feng, C. van Niekerk, and M. Gašić (2021). "What does the User Want? Information Gain for Hierarchical Dialogue Policy Optimisation". In: *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 969–976. DOI: 10.1109/ASRU51503.2021.9687856.

Geishauser, Christian, Carel van Niekerk, Hsien-chin Lin, Nurul Lubis, Michael Heck, Shutong Feng, and Milica Gašić (Oct. 2022). "Dynamic Dialogue Policy for Continual Reinforcement Learning". In: *Proceedings of the 29th International Conference on Computational*

Linguistics. Gyeongju, Republic of Korea: International Committee on Computational Linguistics, pp. 266–284. URL: https://aclanthology.org/2022.coling-1.21.

Geishauser, Christian, Carel van Niekerk, Nurul Lubis, Hsien-chin Lin, Michael Heck, Shutong Feng, Benjamin Ruppik, Renato Vukovic, and Milica Gašić (2024). "Learning With an Open Horizon in Ever-Changing Dialogue Circumstances". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 32, pp. 2352–2366. DOI: 10.1109/TASLP.2024.3385289.

Geng, Binzong, Fajie Yuan, Qiancheng Xu, Ying Shen, Ruifeng Xu, and Min Yang (Aug. 2021). "Continual Learning for Task-oriented Dialogue System with Iterative Network Pruning, Expanding and Masking". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Online: Association for Computational Linguistics, pp. 517–523. DOI: 10.18653/v1/2021.acl-short.66. URL: https://aclanthology.org/2021.acl-short.66.

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, pp. 249–256. URL: https://proceedings.mlr.press/v9/glorot10a.html.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Grossberg, Stephen (1982). "How Does a Brain Build a Cognitive Code?" In: *Studies of Mind and Brain: Neural Principles of Learning, Perception, Development, Cognition, and Motor Control*. Dordrecht: Springer Netherlands, pp. 1–52. ISBN: 978-94-009-7758-7. DOI: 10.1007/978-94-009-7758-7_1. URL: https://doi.org/10.1007/978-94-009-7758-7_1.

Grossberg, Stephen (2013). "Adaptive Resonance Theory: How a Brain Learns to Consciously Attend, Learn, and Recognize a Changing World". In: *Neural Netw.* 37, 1–47. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2012.09.017. URL: https://doi.org/10.1016/j.neunet.2012.09.017.

Gur, Izzeddin, Dilek Hakkani-Tür, Gökhan Tür, and Pararth Shah (2018). "User Modeling for Task Oriented Dialogues". In: *2018 IEEE Spoken Language Technology Workshop, SLT 2018, Athens, Greece, December 18-21, 2018*. IEEE, pp. 900–906. DOI: 10.1109/SLT.2018.8639652. URL: https://doi.org/10.1109/SLT.2018.8639652.

Hadsell, Raia, Dushyant Rao, Andrei Rusu, and Razvan Pascanu (Dec. 2020). "Embracing Change: Continual Learning in Deep Neural Networks". In: *Trends in Cognitive Sciences* 24, pp. 1028–1040. DOI: 10.1016/j.tics.2020.09.004.

Harney, Hanns L. (2003). "Bayes' Theorem". In: *Bayesian Inference: Parameter Estimation and Decisions*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 8–18. ISBN: 978-3-662-06006-3. DOI: 10.1007/978-3-662-06006-3_2. URL: https://doi.org/10.1007/978-3-662-06006-3_2.

Hasselt, Hado (2010). "Double Q-learning". In: *Advances in Neural Information Processing Systems*. Ed. by J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta. Vol. 23. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2010/file/091d584fced301b442654dd8c23b3fc9-Paper.pdf.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015). "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*. ICCV '15. USA: IEEE

Computer Society, 1026–1034. ISBN: 9781467383912. DOI: 10.1109/ICCV.2015.123. URL: https://doi.org/10.1109/ICCV.2015.123.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

Heck, Michael, Nurul Lubis, Benjamin Ruppik, Renato Vukovic, Shutong Feng, Christian Geishauser, Hsien-chin Lin, Carel van Niekerk, and Milica Gasic (July 2023). "ChatGPT for Zero-shot Dialogue State Tracking: A Solution or an Opportunity?" In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Toronto, Canada: Association for Computational Linguistics, pp. 936–950. DOI: 10.18653/v1/2023.acl-short.81. URL: https://aclanthology.org/2023.acl-short.81.

Heck, Michael, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gasic (July 2020). "TripPy: A Triple Copy Strategy for Value Independent Neural Dialog State Tracking". In: *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 1st virtual meeting: Association for Computational Linguistics, pp. 35–44. URL: https://aclanthology.org/2020.sigdial-1.4.

Hessel, Matteo et al. (2018). "Rainbow: Combining Improvements in Deep Reinforcement Learning." In: *AAAI*. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, pp. 3215–3222. URL: http://dblp.uni-trier.de/db/conf/aaai/aaai2018.html#HesselMHSODHPAS18.

Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean (2015). *Distilling the Knowledge in a Neural Network*. arXiv: 1503.02531 [stat.ML].

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5, pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208.

Hou, Zhengxu, Bang Liu, Ruihui Zhao, Zijing Ou, Yafei Liu, Xi Chen, and Yefeng Zheng (June 2021). "Imperfect also Deserves Reward: Multi-Level and Sequential Reward Modeling for Better Dialog Management". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, pp. 2993–3001. DOI: 10.18653/v1/2021.naacl-main.238. URL: https://aclanthology.org/2021.naacl-main.238.

Huang, Xinting, Jianzhong Qi, Yu Sun, and Rui Zhang (July 2020). "Semi-Supervised Dialogue Policy Learning via Stochastic Reward Estimation". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 660–670. DOI: 10.18653/v1/2020.acl-main.62. URL: https://aclanthology.org/2020.acl-main.62.

Hudeček, Vojtěch and Ondrej Dusek (Sept. 2023). "Are Large Language Models All You Need for Task-Oriented Dialogue?" In: *Proceedings of the 24th Meeting of the Special Interest Group on Discourse and Dialogue*. Prague, Czechia: Association for Computational Linguistics, pp. 216–228. URL: https://aclanthology.org/2023.sigdial-1.21.

Imani, Ehsan, Eric Graves, and Martha White (2018). "An Off-Policy Policy Gradient Theorem Using Emphatic Weightings". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Montréal, Canada: Curran Associates Inc., 96–106.

Isele, David and Akansel Cosgun (2018). "Selective Experience Replay for Lifelong Learning". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI'18/IAAI'18/EAAI'18. New Orleans, Louisiana, USA: AAAI Press. ISBN: 978-1-57735-800-8.

Jang, Joel, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, Stanley Jungkyu Choi, and Minjoon Seo (2022). "Towards Continual Knowledge Learning of Language Models". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net. URL: https://openreview.net/forum?id=vfsRB5MImo9.

Javed, Khurram and Martha White (2019). "Meta-Learning Representations for Continual Learning". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/f4dd765c12f2ef67f98f3558c282a9cd-Paper.pdf.

Jhunjhunwala, Megha, Caleb Bryant, and Pararth Shah (July 2020). "Multi-Action Dialog Policy Learning with Interactive Human Teaching". In: *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 1st virtual meeting: Association for Computational Linguistics, pp. 290–296. URL: https://aclanthology.org/2020.sigdial-1.36.

Kaelbling, Leslie Pack, Michael L. Littman, and Anthony R. Cassandra (1998). "Planning and acting in partially observable stochastic domains". In: *Artificial Intelligence* 101.1, pp. 99–134. ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(98)00023-X. URL: https://www.sciencedirect.com/science/article/pii/S000437029800023X.

Kaplanis, Christos, Murray Shanahan, and Claudia Clopath (2019). "Policy Consolidation for Continual Reinforcement Learning". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 3242–3251. URL: http://proceedings.mlr.press/v97/kaplanis19a.html.

Khetarpal, Khimya, Matthew Riemer, Irina Rish, and Doina Precup (Dec. 2022). "Towards Continual Reinforcement Learning: A Review and Perspectives". In: *Journal of Artificial Intelligence Research* 75, pp. 1401–1476. DOI: 10.1613/jair.1.13673.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1412.6980.

Kirkpatrick, James et al. (2017). "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the National Academy of Sciences* 114.13, pp. 3521–3526. DOI: 10.1073/pnas.1611835114. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1611835114. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1611835114.

Kwan, Wai-Chung, Huimin Wang, Hongru Wang, Zezhong Wang, Xian Wu, Yefeng Zheng, and Kam-Fai Wong (2023). *JoTR: A Joint Transformer and Reinforcement Learning Framework for Dialog Policy Learning*. arXiv: 2309.00230 [cs.CL].

Ladosz, Pawel, Lilian Weng, Minwoo Kim, and Hyondong Oh (2022). "Exploration in Deep Reinforcement Learning: A Survey". In: *Inf. Fusion* 85.C, 1–22. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2022.03.003. URL: https://doi.org/10.1016/j.inffus.2022.03.003.

Lee, Hwaran, Jinsik Lee, and Tae-Yoon Kim (July 2019). "SUMBT: Slot-Utterance Matching for Universal and Scalable Belief Tracking". In: *Proceedings of the 57th Annual Meeting of* *the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 5478–5483. DOI: 10.18653/v1/P19-1546. URL: https://aclanthology.org/P19-1546.

Lee, Sungjin (2017). "Toward Continual Learning for Conversational Agents". In: *CoRR* abs/1712.09943. arXiv: 1712.09943. URL: http://arxiv.org/abs/1712.09943.

Lengyel, Máté and Peter Dayan (2007). "Hippocampal Contributions to Control: The Third Way". In: *Advances in Neural Information Processing Systems*. Ed. by J. Platt, D. Koller, Y. Singer, and S. Roweis. Vol. 20. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2007/file/1f4477bad7af3616c1f933a02bfabe4e-Paper.pdf.

Levin, Esther and Roberto Pieraccini (1997). "A Stochastic Model of Computer-Human Interaction for Learning Dialogue Strategies". In: *EUROSPEECH 97*, pp. 1883–1886.

Li, Siheng, Cheng Yang, Yichun Yin, Xinyu Zhu, Zesen Cheng, Lifeng Shang, Xin Jiang, Qun Liu, and Yujiu Yang (July 2023). "AutoConv: Automatically Generating Information-seeking Conversations with Large Language Models". In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Toronto, Canada: Association for Computational Linguistics, pp. 1751–1762. DOI: 10.18653/v1/2023.acl-short.149. URL: https://aclanthology.org/2023.acl-short.149.

Li, Zhizhong and Derek Hoiem (2018). "Learning without Forgetting". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 40.12, 2935–2947. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2017.2773081. URL: https://doi.org/10.1109/TPAMI.2017.2773081.

Li, Ziming, Sungjin Lee, Baolin Peng, Jinchao Li, Julia Kiseleva, Maarten de Rijke, Shahin Shayandeh, and Jianfeng Gao (Nov. 2020). "Guided Dialogue Policy Learning without Adversarial Learning in the Loop". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, pp. 2308–2317. DOI: 10.18653/v1/2020.findings-emnlp.209. URL: https://aclanthology.org/2020.findings-emnlp.209.

Lin, Hsien-chin, Nurul Lubis, Songbo Hu, Carel van Niekerk, Christian Geishauser, Michael Heck, Shutong Feng, and Milica Gašić (July 2021). "Domain-independent User Simulation with Transformers for Task-oriented Dialogue Systems". In: *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Singapore and Online: Association for Computational Linguistics, pp. 445–456. URL: https://aclanthology.org/2021.sigdial-1.47.

Lin, Long-Ji (1992). "Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching". In: *Mach. Learn.* 8.3–4, 293–321. ISSN: 0885-6125. DOI: 10.1007/BF00992699. URL: https://doi.org/10.1007/BF00992699.

Liotet, Pierre, Francesco Vidaich, Alberto Maria Metelli, and Marcello Restelli (2022). "Lifelong Hyper-Policy Optimization with Multiple Importance Sampling Regularization". In: *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*. AAAI Press, pp. 7525–7533. URL: https://ojs.aaai.org/index.php/AAAI/article/view/20717.

Lipton, Zachary C., Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng (2018). "BBQ-Networks: Efficient Exploration in Deep Reinforcement Learning for Task-Oriented Dialogue Systems". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New*

Orleans, Louisiana, USA, February 2-7, 2018. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, pp. 5237–5244. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16189.

Liu, Qingbin, Pengfei Cao, Cao Liu, Jiansong Chen, Xunliang Cai, Fan Yang, Shizhu He, Kang Liu, and Jun Zhao (Nov. 2021). "Domain-Lifelong Learning for Dialogue State Tracking via Knowledge Preservation Networks". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 2301–2311. DOI: 10.18653/v1/2021.emnlp-main.176. URL: https://aclanthology.org/2021.emnlp-main.176.

Lomonaco, Vincenzo et al. (2021). "Avalanche: an End-to-End Library for Continual Learning". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 3595–3605. DOI: 10.1109/CVPRW53098.2021.00399.

Lopez-Paz, David and Marc' Aurelio Ranzato (2017). "Gradient Episodic Memory for Continual Learning". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/f87522788a2be2d171666752f97ddebb-Paper.pdf.

Lu, Liang, Xingxing Zhang, Kyunghyun Cho, and Steve Renals (2015). "A study of the recurrent neural network encoder-decoder for large vocabulary speech recognition". English (US). In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* 2015-January. 16th Annual Conference of the International Speech Communication Association, INTERSPEECH 2015 ; Conference date: 06-09-2015 Through 10-09-2015, pp. 3249–3253. ISSN: 2308-457X.

Luong, Thang, Hieu Pham, and Christopher D. Manning (Sept. 2015). "Effective Approaches to Attention-based Neural Machine Translation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1412–1421. DOI: 10.18653/v1/D15-1166. URL: https://aclanthology.org/D15-1166.

Madotto, Andrea et al. (Nov. 2021). "Continual Learning in Task-Oriented Dialogue Systems". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 7452–7467. URL: https://aclanthology.org/2021.emnlp-main.590.

Madureira, Brielen and David Schlangen (2020). "An Overview of Natural Language State Representation for Reinforcement Learning". In: *ArXiv* abs/2007.09774.

Marsden, Jerrold E. (2012). *Vector calculus*. Ed. by Anthony Tromba. Array. New York: W.H. Freeman, p. 545.

Mazumder, Sahisnu and Bing Liu (2022). *Lifelong and Continual Learning Dialogue Systems*. arXiv: 2211.06553 [cs.CL].

McCloskey, Michael and Neal J. Cohen (1989). "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem". In: ed. by Gordon H. Bower. Vol. 24. Psychology of Learning and Motivation. Academic Press, pp. 109–165. DOI: https://doi.org/10.1016/S0079-7421(08)60536-8. URL: https://www.sciencedirect.com/science/article/pii/S0079742108605368.

Mi, Fei, Liangwei Chen, Mengjie Zhao, Minlie Huang, and Boi Faltings (Nov. 2020). "Continual Learning for Natural Language Generation in Task-oriented Dialog Systems". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, pp. 3461–3474. DOI: 10.18653/v1/2020.findings-emnlp.310. URL: https://aclanthology.org/2020.findings-emnlp.310.

Mnih, Volodymyr et al. (Feb. 2015). "Human-level control through deep reinforcement learning". In: *Nature* 518.7540, pp. 529–533. ISSN: 00280836. URL: `http://dx.doi.org/10.1038/nature14236`.

Mrkšić, Nikola, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young (July 2017). "Neural Belief Tracker: Data-Driven Dialogue State Tracking". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 1777–1788. DOI: `10.18653/v1/P17-1163`. URL: `https://aclanthology.org/P17-1163`.

Namazifar, Mahdi, Alexandros Papangelis, Gokhan Tur, and Dilek Hakkani-Tür (2021). "Language model is all you need: Natural language understanding as question answering". In: *ICASSP 2021*. URL: `https://www.amazon.science/publications/language-model-is-all-you-need-natural-language-understanding-as-question-answering`.

Ng, Andrew Y., Daishi Harada, and Stuart J. Russell (1999). "Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping". In: *Proceedings of the Sixteenth International Conference on Machine Learning*. ICML '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 278–287. ISBN: 1558606122.

Normandin, Fabrice et al. (2021). "Sequoia: A Software Framework to Unify Continual Learning Research". In: *CoRR* abs/2108.01005. arXiv: `2108.01005`. URL: `https://arxiv.org/abs/2108.01005`.

Oord, Aäron van den, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu (2016). "WaveNet: A Generative Model for Raw Audio". In: *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*. ISCA, p. 125. URL: `http://www.isca-speech.org/archive/SSW\_2016/abstracts/ssw9\_DS-4\_van\_den\_Oord.html`.

Ouyang, Long et al. (2022). *Training language models to follow instructions with human feedback*. arXiv: `2203.02155 [cs.CL]`.

Padakandla, Sindhu, Prabuchandran K. J., and Shalabh Bhatnagar (2019). "Reinforcement Learning in Non-Stationary Environments". In: *CoRR* abs/1905.03970. arXiv: `1905.03970`. URL: `http://arxiv.org/abs/1905.03970`.

Parisi, German, Ronald Kemker, Jose Part, Christopher Kanan, and Stefan Wermter (Feb. 2019). "Continual Lifelong Learning with Neural Networks: A Review". In: *Neural Networks* 113, pp. 54–71. DOI: `10.1016/j.neunet.2019.01.012`.

Park, Jeiyoon, Chanhee Lee, Kuekyeng Kim, and Heuiseok Lim (2020). "Variational Reward Estimator Bottleneck: Learning Robust Reward Estimator for Multi-Domain Task-Oriented Dialog". In: *CoRR* abs/2006.00417. arXiv: `2006.00417`. URL: `https://arxiv.org/abs/2006.00417`.

Peng, Baolin, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao (Nov. 2020). "Few-shot Natural Language Generation for Task-Oriented Dialog". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, pp. 172–182. DOI: `10.18653/v1/2020.findings-emnlp.17`. URL: `https://aclanthology.org/2020.findings-emnlp.17`.

Platanios, Emmanouil Antonios, Abulhair Saparov, and Tom M. Mitchell (2020). "Jelly Bean World: A Testbed for Never-Ending Learning". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. URL: `https://openreview.net/forum?id=Byx\_YAVYPH`.

Power, Jonathan and Bradley Schlaggar (Nov. 2016). "Neural plasticity across the lifespan". In: *Wiley Interdisciplinary Reviews: Developmental Biology* 6. DOI: `10.1002/wdev.216`.

Powers, Sam, Eliot Xing, Eric Kolve, Roozbeh Mottaghi, and Abhinav Gupta (2022). “CORA: Benchmarks, Baselines, and Metrics as a Platform for Continual Reinforcement Learning Agents”. In: *Proceedings of The 1st Conference on Lifelong Learning Agents*. Ed. by Sarath Chandar, Razvan Pascanu, and Doina Precup. Vol. 199. Proceedings of Machine Learning Research. PMLR, pp. 705–743. URL: https://proceedings.mlr.press/v199/powers22b.html.

Pritzel, Alexander, Benigno Uria, Sriram Srinivasan, Adrià Puigdomènech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell (2017). “Neural Episodic Control”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 2827–2836. URL: https://proceedings.mlr.press/v70/pritzel17a.html.

Puterman, Martin L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. 1st. USA: John Wiley & Sons, Inc. ISBN: 0471619779.

Qian, Ning (1999). “On the momentum term in gradient descent learning algorithms”. In: *Neural Networks* 12.1, pp. 145–151. ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(98)00116-6. URL: https://www.sciencedirect.com/science/article/pii/S0893608098001166.

Ramadan, Osman, Paweł Budzianowski, and Milica Gašić (July 2018). “Large-Scale Multi-Domain Belief Tracking with Knowledge Sharing”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 432–437. DOI: 10.18653/v1/P18-2069. URL: https://aclanthology.org/P18-2069.

Rastogi, Abhinav, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan (2020). “Towards Scalable Multi-Domain Conversational Agents: The Schema-Guided Dialogue Dataset”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.05, pp. 8689–8696. DOI: 10.1609/aaai.v34i05.6394. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6394.

Riemer, Matthew, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro (2019). “Learning to Learn without Forgetting By Maximizing Transfer and Minimizing Interference”. In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=B1gTShAct7.

Rolnick, David, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne (2019). “Experience Replay for Continual Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Paper.pdf.

Rosenbaum, Andy, Saleh Soltan, Wael Hamza, Yannick Versley, and Markus Boese (Oct. 2022). “LINGUIST: Language Model Instruction Tuning to Generate Annotated Utterances for Intent Classification and Slot Tagging”. In: *Proceedings of the 29th International Conference on Computational Linguistics*. Gyeongju, Republic of Korea: International Committee on Computational Linguistics, pp. 218–241. URL: https://aclanthology.org/2022.coling-1.18.

Rosenthal, Jeffrey S. (2006). *A first look at rigorous probability theory*. Second. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, pp. xvi+219. ISBN: 978-981-270-371-2; 981-270-371-3.

Ruder, Sebastian (2016). “An overview of gradient descent optimization algorithms”. In: *CoRR* abs/1609.04747. arXiv: 1609.04747. URL: http://arxiv.org/abs/1609.04747.

Rusu, Andrei A., Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell (2016). "Progressive Neural Networks". In: *CoRR* abs/1606.04671. arXiv: `1606.04671`. URL: `http://arxiv.org/abs/1606.04671`.

Schatzmann, Jost, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young (Apr. 2007). "Agenda-Based User Simulation for Bootstrapping a POMDP Dialogue System". In: *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*. Rochester, New York: Association for Computational Linguistics, pp. 149–152. URL: `https://aclanthology.org/N07-2038`.

Schaul, Tom et al. (2018). "The Barbados 2018 List of Open Issues in Continual Learning". In: *CoRR* abs/1811.07004. arXiv: `1811.07004`. URL: `http://arxiv.org/abs/1811.07004`.

Schulman, John, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel (2016). "High-Dimensional Continuous Control Using Generalized Advantage Estimation". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: `http://arxiv.org/abs/1506.02438`.

Schuster, M. and K.K. Paliwal (1997). "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681. DOI: `10.1109/78.650093`.

Schwarz, Jonathan, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell (2018). "Progress & Compress: A scalable framework for continual learning". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 4535–4544. URL: `http://proceedings.mlr.press/v80/schwarz18a.html`.

Shin, Hanul, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim (2017). "Continual Learning with Deep Generative Replay". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2994–3003. ISBN: 9781510860964.

Shu, Lei, Hu Xu, Bing Liu, and Piero Molino (Nov. 2019). "Modeling Multi-Action Policy for Task-Oriented Dialogues". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 1304–1310. DOI: `10.18653/v1/D19-1130`. URL: `https://aclanthology.org/D19-1130`.

Silver, David, Satinder Singh, Doina Precup, and Richard S. Sutton (2021). "Reward is enough". In: *Artificial Intelligence* 299, p. 103535. ISSN: 0004-3702. DOI: `https://doi.org/10.1016/j.artint.2021.103535`. URL: `https://www.sciencedirect.com/science/article/pii/S0004370221000862`.

Sinitsin, Anton, Vsevolod Plokhotnyuk, Dmitry Pyrkin, Sergei Popov, and Artem Babenko (2020). "Editable Neural Networks". In: *International Conference on Learning Representations*. URL: `https://openreview.net/forum?id=HJedXaEtvS`.

Su, Pei-Hao, Milica Gašić, Nikola Mrkšić, Lina Maria Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve J. Young (2016). "Continuously Learning Neural Dialogue Management". In: *CoRR* abs/1606.02689. arXiv: `1606.02689`. URL: `http://arxiv.org/abs/1606.02689`.

Su, Pei-Hao, David Vandyke, Milica Gašić, Nikola Mrkšić, Tsung-Hsien Wen, and Steve Young (Sept. 2015). "Reward Shaping with Recurrent Neural Networks for Speeding up On-Line Policy Learning in Spoken Dialogue Systems". In: *Proceedings of the 16th Annual*

Meeting of the Special Interest Group on Discourse and Dialogue. Prague, Czech Republic: Association for Computational Linguistics, pp. 417–421. DOI: 10.18653/v1/W15-4655. URL: https://aclanthology.org/W15-4655.

Sutton, Richard S. and Andrew G. Barto (2018). *Reinforcement learning: an introduction*. Second. The MIT Press. URL: http://incompleteideas.net/book/the-book-2nd.html.

Sutton, Richard S, David McAllester, Satinder Singh, and Yishay Mansour (1999). "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *Advances in Neural Information Processing Systems*. Ed. by S. Solla, T. Leen, and K. Müller. Vol. 12. MIT Press.

Takanobu, Ryuichi, Hanlin Zhu, and Minlie Huang (Nov. 2019). "Guided Dialog Policy Learning: Reward Estimation for Multi-Domain Task-Oriented Dialog". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 100–110. DOI: 10.18653/v1/D19-1010. URL: https://www.aclweb.org/anthology/D19-1010.

Taylor, Matthew E. and Peter Stone (2009). "Transfer Learning for Reinforcement Learning Domains: A Survey". In: *J. Mach. Learn. Res.* 10, 1633–1685. ISSN: 1532-4435.

Teh, Yee Whye, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu (2017). "Distral: Robust Multitask Reinforcement Learning". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 4499–4509. ISBN: 9781510860964.

Touvron, Hugo et al. (2023). *Llama 2: Open Foundation and Fine-Tuned Chat Models*. arXiv: 2307.09288 [cs.CL].

Traoré, René, Hugo Caselles-Dupré, Timothée Lesort, Te Sun, Guanghang Cai, Natalia Díaz Rodríguez, and David Filliat (2019). "DisCoRL: Continual Reinforcement Learning via Policy Distillation". In: *CoRR* abs/1907.05855. arXiv: 1907.05855. URL: http://arxiv.org/abs/1907.05855.

Tseng, Bo-Hsiang, Yinpei Dai, Florian Kreyssig, and Bill Byrne (Aug. 2021). "Transferable Dialogue Systems and User Simulators". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, pp. 152–166. DOI: 10.18653/v1/2021.acl-long.13. URL: https://aclanthology.org/2021.acl-long.13.

Tur, Gokhan, Li Deng, Dilek Hakkani-Tür, and Xiaodong He (2012). "Towards deeper understanding: Deep convex networks for semantic utterance classification". In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5045–5048. DOI: 10.1109/ICASSP.2012.6289054.

van Niekerk, Carel, Michael Heck, Christian Geishauser, Hsien-chin Lin, Nurul Lubis, Marco Moresi, and Milica Gasic (Nov. 2020). "Knowing What You Know: Calibrating Dialogue Belief State Distributions via Ensembles". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, pp. 3096–3102. DOI: 10.18653/v1/2020.findings-emnlp.277. URL: https://aclanthology.org/2020.findings-emnlp.277.

van Niekerk, Carel, Andrey Malinin, Christian Geishauser, Michael Heck, Hsien-chin Lin, Nurul Lubis, Shutong Feng, and Milica Gasic (Nov. 2021). "Uncertainty Measures in Neural Belief Tracking and the Effects on Dialogue Policy Performance". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta

Cana, Dominican Republic: Association for Computational Linguistics, pp. 7901–7914. DOI: 10.18653/v1/2021.emnlp-main.623. URL: https://aclanthology.org/2021.emnlp-main.623.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Ven, Michiel van der and Andreas S. Tolias (2018). “Generative replay with feedback connections as a general strategy for continual learning”. In: *CoRR* abs/1809.10635. arXiv: 1809.10635. URL: http://arxiv.org/abs/1809.10635.

Vlasov, Vladimir, Johannes E. M. Mosig, and Alan Nichol (2019). “Dialogue Transformers”. In: *CoRR* abs/1910.00486. arXiv: 1910.00486. URL: http://arxiv.org/abs/1910.00486.

Wang, Zhuoran, Tsung-Hsien Wen, Pei-Hao Su, and Yannis Stylianou (Sept. 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*. Prague, Czech Republic: Association for Computational Linguistics, pp. 412–416. DOI: 10.18653/v1/W15-4654. URL: https://www.aclweb.org/anthology/W15-4654.

Wang, Ziyu, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas (2016). “Dueling Network Architectures for Deep Reinforcement Learning”. In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, pp. 1995–2003. URL: https://proceedings.mlr.press/v48/wangf16.html.

Watkins, Christopher J. C. H. and Peter Dayan (May 1992). “Q-learning”. In: *Machine Learning* 8.3, pp. 279–292. ISSN: 1573-0565. DOI: 10.1007/BF00992698. URL: https://doi.org/10.1007/BF00992698.

Weisz, Gellért, Paweł Budzianowski, Pei-Hao Su, and Milica Gašić (2018). “Sample Efficient Deep Reinforcement Learning for Dialogue Systems With Large Action Spaces”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.11, pp. 2083–2097. DOI: 10.1109/TASLP.2018.2851664.

Wen, Tsung-Hsien, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young (Sept. 2015). “Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1711–1721. DOI: 10.18653/v1/D15-1199. URL: https://aclanthology.org/D15-1199.

Wesselmann, Paula, Yen-Chen Wu, and Milica Gašić (2019). “Curiosity-driven Reinforcement Learning for Dialogue Management”. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7210–7214. DOI: 10.1109/ICASSP.2019.8683033.

Williams, Jason D. and Steve Young (Apr. 2007). “Partially Observable Markov Decision Processes for Spoken Dialog Systems”. In: *Comput. Speech Lang.* 21.2, 393–422. ISSN: 0885-2308. DOI: 10.1016/j.csl.2006.06.008. URL: https://doi.org/10.1016/j.csl.2006.06.008.

Wołczyk, Maciej, Michał Zając, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś (2021). “Continual World: A Robotic Benchmark For Continual Reinforcement Learning”. In:

Advances in Neural Information Processing Systems. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., pp. 28496–28510. URL: https://proceedings.neurips.cc/paper/2021/file/ef8446f35513a8d6aa2308357a268a7e-Paper.pdf.

Wortsman, Mitchell, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi (2020). "Supermasks in Superposition". In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS'20. Vancouver, BC, Canada: Curran Associates Inc. ISBN: 9781713829546.

Wu, Chien-Sheng, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung (July 2019). "Transferable Multi-Domain State Generator for Task-Oriented Dialogue Systems". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 808–819. DOI: 10.18653/v1/P19-1078. URL: https://aclanthology.org/P19-1078.

Xie, Annie, James Harrison, and Chelsea Finn (2021). "Deep Reinforcement Learning amidst Continual Structured Non-Stationarity". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 11393–11403. URL: https://proceedings.mlr.press/v139/xie21c.html.

Xu, Ju and Zhanxing Zhu (2018). "Reinforced Continual Learning". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Montréal, Canada: Curran Associates Inc., 907–916.

Xu, Yumo, Chenguang Zhu, Baolin Peng, and Michael Zeng (2020). "Meta Dialogue Policy Learning". In: *CoRR* abs/2006.02588. arXiv: 2006.02588. URL: https://arxiv.org/abs/2006.02588.

Xu, Zhongwen, Hado P van Hasselt, and David Silver (2018). "Meta-Gradient Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/2715518c875999308842e3455eda2fe3-Paper.pdf.

Yoon, Jaehong, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang (2018). "Lifelong Learning with Dynamically Expandable Networks". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=Sk7KsfW0-.

Young, Steve, Jost Schatzmann, Karl Weilhammer, and Hui Ye (2007). "The Hidden Information State Approach to Dialog Management". In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*. Vol. 4, pp. IV–149–IV–152. DOI: 10.1109/ICASSP.2007.367185.

Zenke, Friedemann, Ben Poole, and Surya Ganguli (2017). "Continual Learning through Synaptic Intelligence". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 3987–3995.

Zhang, Yichi, Zhijian Ou, and Zhou Yu (2020). "Task-Oriented Dialog Systems That Consider Multiple Appropriate Responses under the Same Context". In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, pp. 9604–9611. URL: https://aaai.org/ojs/index.php/AAAI/article/view/6507.

Zheng, Zeyu, Junhyuk Oh, Matteo Hessel, Zhongwen Xu, Manuel Kroiss, Hado Van Hasselt, David Silver, and Satinder Singh (2020). "What Can Learned Intrinsic Rewards Capture?" In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 11436–11446. URL: https://proceedings.mlr.press/v119/zheng20b.html.

Zheng, Zeyu, Junhyuk Oh, and Satinder Singh (2018). "On Learning Intrinsic Rewards for Policy Gradient Methods". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/51de85ddd068f0bc787691d356176df9-Paper.pdf.

Zhu, Qi, Bing Li, Fei Mi, Xiaoyan Zhu, and Minlie Huang (May 2022a). "Continual Prompt Tuning for Dialog State Tracking". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, pp. 1124–1137. DOI: 10.18653/v1/2022.acl-long.80. URL: https://aclanthology.org/2022.acl-long.80.

Zhu, Qi et al. (July 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*. Online: Association for Computational Linguistics, pp. 142–149. DOI: 10.18653/v1/2020.acl-demos.19. URL: https://aclanthology.org/2020.acl-demos.19.

Zhu, Qi et al. (2022b). *ConvLab-3: A Flexible Dialogue System Toolkit Based on a Unified Data Format*. DOI: 10.48550/ARXIV.2211.17148. URL: https://arxiv.org/abs/2211.17148.

Zhu, Zhuangdi, Kaixiang Lin, Bo Dai, and Jiayu Zhou (2022c). "Self-Adaptive Imitation Learning: Learning Tasks with Delayed Rewards from Sub-optimal Demonstrations". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.8, pp. 9269–9277. DOI: 10.1609/aaai.v36i8.20914. URL: https://ojs.aaai.org/index.php/AAAI/article/view/20914.

Ziegler, Daniel M., Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul F. Christiano, and Geoffrey Irving (2019). "Fine-Tuning Language Models from Human Preferences". In: *CoRR* abs/1909.08593. arXiv: 1909.08593. URL: http://arxiv.org/abs/1909.08593.