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.

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., 2020; 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). Moreover, a model should only have minimal increase in model capacity and access to previous tasks. Mitigating catastrophic forgetting is often prioritised. However, all of these points are critical (Hadsell et al., 2020).

Despite progress in continual learning (Lange et al., 2019; Parisi et al., 2019; Biesialska et al., 2020; Khetarpal et al., 2020; Hadsell et al., 2020), the task-oriented dialogue systems research community has barely touched the topic (Lee, 2017; Wu et al., 2019; Mi et al., 2020; Madotto et al., 2021; Geng et al., 2021). Only one of the prior works considers action prediction (Lee, 2017) and – to the best of our knowledge – none address 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 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

2. 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

---

1We will release the code and training logs upon publication of this work.
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 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 predictions when the domain changes (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 dialogue 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 continual RL, a task $z$ constitutes a stationary Markov decision process (MDP) $M^{(z)} = \langle S^{(z)}, A^{(z)}, p^{(z)}, r^{(z)}, γ^{(z)} \rangle$ (Khetarpal et al., 2020; Mendez et al., 2020). In the context of dialogue a task usually refers to a domain as in Madotto et al. (2021). A state-of-the-art method for continual RL that uses a replay memory is CLEAR (Rolnick et al., 2018). CLEAR deals with the tug-of-war dynamics in which the current task and previous tasks pull from different sides through the following loss functions. For policy $\pi$ with parameters $\theta$, CLEAR uses the loss

$$L_{\text{policy}}(\theta) = \alpha_1 \cdot L_{\text{vtrace-policy}}(\theta) + \alpha_2 \cdot L_{\text{policy-cloning}}(\theta) + \alpha_3 \cdot L_{\text{entropy}}(\theta),$$

where $L_{\text{vtrace-policy}}(\theta)$ is the policy loss described in Espeholt et al. (2018), using a mixture of on-policy experience and off-policy experience. $L_{\text{policy-cloning}}$ uses only off-policy experience and aims at preventing forgetting through minimising the Kullback-Leibler (KL) divergence between current and behaviour policy. Finally, $L_{\text{entropy}}$ tries to prevent premature convergence by increasing the entropy of the policy.

The critic, parameterised by $\psi$, is defined in a similar fashion, using the critic loss defined in Espeholt et al. (2018) and an additional mean-squared error loss on the off-policy experience:

$$L_{\text{value}}(\psi) = \beta_1 \cdot L_{\text{vtrace-critic}}(\psi) + \beta_2 \cdot L_{\text{value-cloning}}(\psi).$$

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 Niekerv, 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. We propose an architecture that uses the transformer encoder with information embeddings (Section 4.1) and the transformer decoder that leverages the domain gate (Section 4.2 and 4.3), which we call dynamic dialogue policy transformer (DDPT). It is clear that the capacities of our model greatly depend on the choice of the state and action representation as these highly affect forward and backward transfer as well as the plasticity of the model.

### 4.1 State Representation

Recall from Section 3.2 that the policy 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 policy has access to an external dictionary providing a natural language description \(\text{descr}_f\) of each of these, e.g. “area of the hotel” or “number of hotel database results”. See Appendix A.4 for the 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

\[
e_{\text{info}_f} = \text{Lin} \left( \text{LM} \left( \text{descr}_f \right), \text{Lin}(v_f) \right) \in \mathbb{R}^h
\]

where \(\text{LM}\) denotes applying a language model such as RoBERTa (Liu et al., 2019) and averaging of the token embeddings, and \(\text{Lin}\) denotes a linear layer. \(e_{\text{info}_f}\) represents information in a high-dimensional vector space. The list of information embeddings are the input to a transformer encoder (Vaswani et al., 2017). The attention mechanism allows us to decide for every information embedding \(e_{\text{info}_f}\) on which other embeddings \(e_{\text{info}_g}\) 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
Figure 2: 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. In addition, the previous output is fed into the decoder. The probability distribution is calculated through a scalar product between the decoder output and embeddings obtained with RoBERTa. In case of domain prediction, we propose a domain gate that decides whether to choose a domain that the user currently talks about. The action decoding terminates when the model predicts the stop domain.

effective state space remains small, making it possible to handle a very large number of domains. Our proposed state encoder is depicted in Figure 1(c).

In this way, the state representation meets the 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 $D$, intent set $I$ and slot set $S$ as follows. The domain set $D$ consists of all domains the model has seen so far plus an additional stop domain. The intent set $I$ and slot set $S$ 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 $b^d \forall d \in D$, $b^i \forall i \in I$, and $b^s \forall s \in S$.

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 $(domain_m, intent_m, slot_m)^n_{m=0}$. 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 $b_{t-1} + l_t$, where $b_{t-1}$ is the embedding of the previous prediction and $l_t$ is a token embedding for token domain, intent or slot that indicates what needs to be predicted in turn $t$. $b_{-1}$ 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 $o_t$ and the different domain embeddings $b^d$ and apply the softmax function to obtain a probability distribution $\text{softmax} [o_t \odot b^d, d \in D]$ 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 of our model.

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. The action selection process described can be easily generalised to actions that only have a description and are associated to a domain. 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 dia-
We follow the setup recently proposed by Pow-

ers et al. (2021), which assumes that our $N$ tasks
$z_1, ..., z_N$ 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 $x_1, ..., x_{N:M}$. 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 perfor-

mance 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 train-
ing 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

$$F_{i,j} = \frac{1}{s} \sum_{k=0}^{j-1} \max_{s} \{m_{i,k} - m_{i,j}\}. \quad (1)$$

This term 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 $F_{i,j}$
is positive, the agent has become worse at past task
$z_i$ after training on task $x_j$, indicating forgetting.

When $F_{i,j}$ is negative, the agent has become better
at task $z_i$, indicating backward transfer. Note that
because we cycle through the tasks multiple times,
tasks $x_j$ and $x_i$ can refer to the same task $z$ (for
instance if $l = j + N$). Similar to Powers et al.
(2021), we average the metrics $F_{i,j}$ and $F_{j,i}$ if $j$
and $l$ refer to the same task $z$. We define $Z_i$ as
the average over the $F_{i,j}$.

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

$$Z_{i,j} = \frac{1}{s} \sum_{s} m_{i,j}. \quad (2)$$

We do not subtract initial performance as in Pow-

er et al. (2021) as we are interested in the absolute
performance that tells us how well we do on future
tasks $z_i$ after training on a task $z_j$. We define $Z_i$
as the average over the $Z_{i,j}$.

Forward transfer and forgetting allow us to un-
derstand relationships between domains and which
domains benefit from or interfere with each other.

5.2 Baselines

We implemented two baselines in order to compare
against our proposed DDPT architecture.

5.2.1 Baseline State Representations

We will abbreviate the two 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 param-
eters need to be initialised. The state encoding
can be seen in Figure 1(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 embed-
dings for domains, intents, slots and values that can
capture semantic meaning and allow cross-domain
transfer. Using trainable embeddings, one repre-
sentation 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.3. This baseline serves as a representative of methods where feature representations remain fixed.

### 5.2.2 Baseline Action Prediction

Unlike DDPT, which uses a transformer for action prediction, both baselines use an RNN model for action prediction. 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 $(domain_m, intent_m, slot_m)$ as $r_{io} = -\rho \cdot n$, where $\rho \in \mathbb{N}$ defines the degree of the penalty. 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 3. The plots show the performance averaged over domains. The plots show the performance averaged over domains. The vertical line at 8000 dialogues indicates the start of cycle 2. The shaded area represents standard deviation.

![Average return plots](image)

**Figure 3:** 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 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.

- **(a) easy-to-hard order**
- **(b) hard-to-easy order**
- **(c) mixed domain order**
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 still 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.6.

### 6.2 Forward Transfer and Forgetting

We calculated forward and forgetting metrics as explained in Section 5.1. In Tables 1 and 2 we show success rates instead of average return because success is easier to interpret than return. We can see for every model and every domain \(i\) the continual learning metrics \(Z_i\) and \(F_i\), respectively. Recall that \(Z_i\) encodes how much zero-shot transfer we obtain on average on task \(i\) by training on previous tasks and \(F_i\) encodes how much is forgotten on average on task \(i\) when training on later tasks.

Firstly, Table 1 reveals that DDPT has by far the largest zero-shot performance in every domain order (last row). 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, therefore enabling the model to draw connections between learned tasks and new ones. As the choice of domain is left to the domain gate, the policy does not have to explore domains first. The last column showing the average over models and orders for every domain reveals that by far the smallest transfer is obtained for the train domain. In contrast, large zero-shot transfer is achieved for the restaurant and taxi domains. If we focus on hard-to-easy, where we first train on the hotel domain and then on the restaurant domain, we observe large forward transfer to restaurant. This confirms the intuition that the similar domains hotel and restaurant should require similar behaviour.

Secondly, Table 2 shows that DDPT hardly suffers from forgetting compared to the baselines (last row). We suspect that the frozen language model embeddings are more robust to domain changes, whereas the trainable embeddings are pulled towards the recent domain, leading to more forgetting of old behaviour. We can observe that for all models forgetting is highest for the starting domain (except for Bin and Sem in mixed order). This is reasonable: training becomes more robust the more tasks we observe, as the replay buffer leads to a kind of multi-task learning. On average, attention suffers the most from forgetting.

### 7 Conclusion

In this work we focused on continual RL for the dialogue policy. We provided an algorithm, baseline models and evaluation metrics to enable continual RL experiments. 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 allows integration of new information seamlessly as long as it is descriptive, and obtains significant zero-shot performance on unseen domains while being robust to forgetting. We hope that this work accelerates research in continual RL for the dialogue policy. In the future we would like to expand task definitions to multiple domains in a single dialogue and evaluate algorithms that aim for fast adaptation with our DDPT model.
References

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

Pawel 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.

Iñigo Casanueva, Pawel 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.

Arslan Chaudhry, Puneet Kumar Dokania, Thalayasingam 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 Cosgum. 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. 2020. 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 Timme Tuytelaars. 2019. Continual learning: A comparative study on how to defy forgetting in classification tasks. CoRR, abs/1909.08383.

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 Geishauer, 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, Seunghwan 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.

Jorge A. Mendez, Boyu Wang, and Eric Eaton. 2020. Lifelong policy gradient learning of factored policies for faster training without forgetting. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, NeurIPS 2020, December 6-12, 2020, virtual.

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.

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.

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*.

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, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz 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, Pawel 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.
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 $\tilde{\rho}$ 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. We set $\alpha_1 = \beta_1 = 1.0$. The weights $\alpha_2$ and $\beta_2$ are set to 0.1. The weight $\alpha_3$ for the entropy loss is set to 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.2 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 there is no entity found with the current constraints, the policy is not allowed to inform the user to say that there are no entities available.
- If the database query tells us that entities for a domain are available, the policy is not allowed to inform about information about entities.
- The Booking domain is only usable for hotel and restaurant.

A.3 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 \(\{d, i_k, s_k\}\) is first embedded as \(s_{u-act,d} = \frac{1}{n} \sum_{k=0}^{n}[v_d, v_{i_k}, v_{s_k}]\), where \(v_d, v_{i_k}\) and \(v_{s_k}\) are trainable embeddings for domain \(d\), intents \(i_k\) and slots \(s_k\) and afterwards fed through a residual block, leading to \(s_{u-act,d} = s_{u-act,d} + \text{ReLU}(W_v s_{u-act,d} + b_v)\).

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 \(s_u = \frac{1}{|D|} \sum_{d \in D} s_{u-act,d}\).

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 1(b). We abbreviate this baselines as \(Sem\) as it uses semantic features.

### A.4 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.

- 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.
Figure 4: 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.

Figure 5: Success rate for each individual domain, where algorithms are trained in the order easy-to-hard.

tracks are capable of providing the information about active domains and often themselves utilise concept descriptions. Another possible limitation is the computing time, which we hope to improve in future. Concerning risk, one possible issue, intrinsic to all deep learning approaches, is the lack of interpretability. We hope that we can leverage the attention mechanism of the transformer to obtain a better understanding of the choices our model takes.
Figure 6: Average return for each individual domain, where algorithms are trained in the order easy-to-hard.

Figure 7: Success rate for each individual domain, where algorithms are trained in the order hard-to-easy.
Figure 8: Average return for each individual domain, where algorithms are trained in the order hard-to-easy.

Figure 9: Success rate for each individual domain, where algorithms are trained in the order mixed.
Figure 10: Average return for each individual domain, where algorithms are trained in the order mixed.