Continual Learning in Task-Oriented Dialogue Systems

Andrea Madotto[1*], **Zhaojiang Lin**[1*], **Zhenpeng Zhou**[2], **Seungwhan Moon**[2]
Paul Crook[2], **Bing Liu**[2], **Zhou Yu**[3], **Eunjoon Cho**[2], **Pascale Fung**[1], **Zhiguang Wang**[2]
[1]The Hong Kong University of Science and Technology
[2]Facebook, [3]Columbia University
amadotto@connect.ust.hk, zgwang@fb.coms

Abstract

Continual learning in task-oriented dialogue systems allows the system to add new domains and functionalities over time after deployment, without incurring the high cost of retraining the whole system each time. In this paper, we propose a first-ever continual learning benchmark for task-oriented dialogue systems with 37 domains to be learned continuously in both modularized and end-to-end learning settings. In addition, we implement and compare multiple existing continual learning baselines, and we propose a simple yet effective architectural method based on residual adapters. We also suggest that the upper bound performance of continual learning should be equivalent to multitask learning when data from all domain is available at once. Our experiments demonstrate that the proposed architectural method and a simple replay-based strategy perform better, by a large margin, compared to other continuous learning techniques, and only slightly worse than the multitask learning upper bound while being 20X faster in learning new domains. We also report several trade-offs in terms of parameter usage, memory size and training time, which are important in the design of a task-oriented dialogue system. The proposed benchmark is released to promote more research in this direction[1].

1 Introduction

Task-oriented dialogue systems (ToDs) are the core technology of the current state-of-the-art smart assistants (e.g. Alexa, Siri, Portal, etc.). These systems are either modularized as a pipeline of multiple components, namely, natural language understanding (NLU), dialogue state tracking (DST), dialogue policy (DP) and natural language generation (NLG), or end-to-end, where a single model implicitly learns how to issue APIs and system responses.

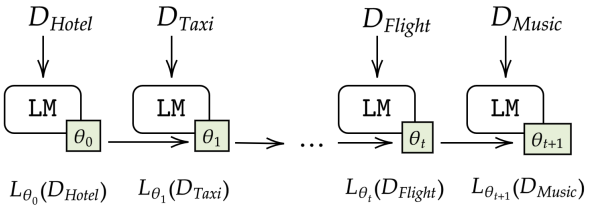

Figure 1: In continual learning, the model is trained one dataset at the time. In this instance, the model is first trained on data from the hotel domain D_{Hotel} then on the Taxi domain D_{Taxi} and so on. The parameters of the model are updated sequentially based on the loss function L.

These systems are continuously updated with new features based on the user's needs, e.g., adding new slots and intents, or even completely new domains. However, existing dialogue models are trained with the assumption of having a fixed dataset at the beginning of the training, and they are not designed to add new domains and functionalities through time without incurring the high cost of retraining the whole system. The ability to acquire new knowledge continuously, a.k.a. continual learning (CL) (Thrun and Pratt, 2012), represents a common challenge to many production and on-device dialogue systems where there is a continual growth of 1st and 3rd-party-developer domains that are added after deployment. Therefore, it is crucial to design dialogue systems with CL ability. Figure 1 shows an high-level intuition of CL in ToDs.

In the CL setting the main challenge is *catastrophic forgetting* (McCloskey and Cohen, 1989). This phenomena happens because there is a distributional shift between the tasks in the curriculum which leads to catastrophic forgetting of the previously acquired knowledge. To overcome this challenge three kinds of methods are usually deployed: loss *regularization*, for avoiding interference with the previously learned tasks, *rehearsal*, which use episodic memory to recall previously learned tasks, and *architectural*, which add task-specific parame-

[*] Work done during internship at Facebook
[1] https://github.com/andreamad8/ToDCL

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 7452–7467
November 7–11, 2021.

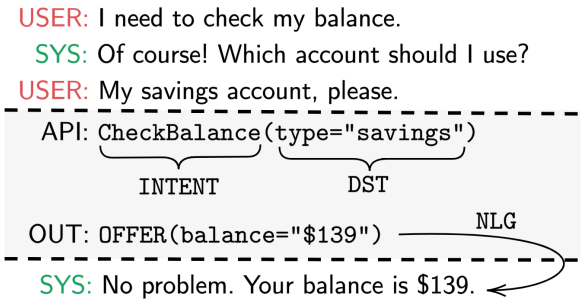

Figure 2: Example of input-out pairs, for the four settings, INTENT, DST, NLG and end-to-end (E2E).

ters for each learned task. However, architectural methods are usually not considered as a baseline, especially in sequence-to-sequence (Seq2Seq) generation tasks (Sun et al., 2019), because they usually *require an additional step* during testing to select which parameter to use for the given task.

To the best our knowledge, continual learning in task-oriented dialogue systems (Lee, 2017) is mostly unexplored or has been studied only in limited settings (e.g., NLG (Mi et al., 2020)) using only a few tasks learned continuously. Given the importance of the task in the dialogue setting, we believe that a more comprehensive investigation is required, especially by comparing multiple settings and baselines. Therefore, in this paper, we make the following contributions:

1. We propose a benchmark for continual learning in ToDs, with 37 tasks to be learned continuously on four settings.
2. We propose a simple yet effective architectural CL method based on residual adapters (Houlsby et al., 2019) that can continuously learn tasks without the need of a task classifier at testing time.
3. We analyse the trade-off between number-of-parameters, episodic memory sizes, and training time of the three main categories of CL methods (regularization, rehearsal, architectural).

2 Background

2.1 Task-Oriented Dialogue Modelling

In this paper, we model task-oriented dialogue systems as a seq2seq generation task (Lei et al., 2018; Lin et al., 2020b; Byrne et al., 2020; Lin et al., 2021) that generates both API-calls and system responses. As shown in Figure 2, the model takes as input a dialogue history, which is the concatenation of user intents and current dialogue states, and then uses its API-call returns, which can be empty or system speech-acts, to generate its system response. This modelling choice is guided by the existing annotated dialogue datasets, which provide the intent and the dialogue state of the user at every turn, and the speech-act of the system; and it allows us to define four distinct settings for studying CL: intent recognition (INTENT), dialogue state tracking (DST), natural language generation (NLG) and end-to-end (E2E). In the coming paragraphs, we formally describe the four settings as different input-out pairs for a seq2seq model.

Data-Formatting Let us define the dialogue history H as a single sequence of tokens from the concatenation of the alternating utterances from the user and the system turns respectively. Without loss of generality, we assume that H has all the dialogue history without the last system utterance, denoted as S. To distinguish between speakers, we add two special tokens at the beginning of every utterance: `USER:` for the user utterance and `SYSTEM:` for the system utterance. Then, we define an API-call, denoted by S_{API}, as the concatenation of the API-name, i.e., the user-intent, and its arguments, i.e., slot-value pairs from the DST. The following syntax is used:

$$S_{API} = \underbrace{\mathbf{I}}_{\text{Intent}} \underbrace{(s_1 = v_1, \ldots, s_k = v_p)}_{\text{Slot-value pairs}} \quad (1)$$

where $\mathbf{I}$ is an intent or the API-name, s_i the slot-name and v_i one of the possible values for the slot s_i. The return of the API-call is either an empty string, thus the model uses the dialogue history to generate a response, or a speech-act, denoted as S_{OUT}, in the same format as the API-call in Equation 1. Similar to the dialogue history, we define two special tokens `API:` and `OUT:` for triggering the model to generate the API-call and for distinguishing the return of the API from the dialogue history respectively. Based on this pre-processing, we define the four settings used in this paper.

Without loss of generality, we define the three modularized settings by their input-out pairs:

$$\begin{array}{cc} H \rightarrow \mathbf{I} & \text{(INTENT)} \\ H \rightarrow \mathbf{I}(s_1 = v_1, \ldots, s_k = v_p) & \text{(DST)} \\ \underbrace{\mathbf{I}(s_1 = v_1, \ldots, s_k = v_p)}_{S_{OUT}} \rightarrow S & \text{(NLG)} \end{array}$$

whereas for the end-to-end (E2E) setting we define the pairs as:

$$H \rightarrow \underbrace{\mathbf{I}(s_1 = v_1, \ldots, s_k = v_p)}_{S_{API}}$$
$$H + \underbrace{\mathbf{I}(s_1 = v_1, \ldots, s_k = v_p)}_{S_{OUT}} \rightarrow S$$

Often, S_{OUT} is empty and thus the model maps the dialogue history to the response ($H \rightarrow S$). An example of input-out pairs is shown in Figure 2.

Finally, we define a dialogue dataset as $D_K = \{(X_i, Y_i)\}_i^n$, where (X_i, Y_i) is a general input-out pair from one of the four settings in consideration, and K is the dialogue domain under consideration (e.g., hotel).

Model In this paper, we employ decoder-only language models (e.g., GPT-2), which are often used in the current state-of-the-art task-oriented dialogue models as in Peng et al. (2020) and Hosseini-Asl et al. (2020). Then, given the concatenation of the input $X = \{x_0, \ldots, x_n\}$ and output $Y = \{x_{n+1}, \ldots, x_m\}$ sequences, we compute the conditional language model distribution using the chain rule of probability as

$$p_\theta(Y|X) = \prod_{i=0}^{n+m} p_\theta(x_i|x_0, \cdots, x_{i-1}), \quad (2)$$

where θ are the model's parameters. The parameters are trained to minimize the negative log-likelihood over a dataset D of input-out pairs, which in our case is the data of the four settings. Formally, we define the loss $\mathcal{L}_\theta$ as:

$$\mathcal{L}_\theta(D) = -\sum_{j}^{|D|} \sum_{i=0}^{n+m} \log p_\theta(x_i^{(j)}|x_0^{(j)}, \cdots, x_{i-1}^{(j)}), \quad (3)$$

where $n + m$ is a maximum sequence length in D. At inference time, given the input sequence X, the model parameterized by θ autoregressively generates the output sequence Y.

2.2 Continual learning

The goal of continual learning is to learn a set of tasks sequentially without catastrophically forgetting the previously learned tasks. In task-oriented dialogue systems, we cast CL as learning a sequence of domains sequentially, (as opposed to multitask learning where all domains are assumed to be present and learned together). Let us define a curriculum of T domains as an ordered set $\mathcal{D} = \{D_1, \cdots, D_T\}$, where D_K is a dataset under the domain K. In addition, we denote the models' parameters after learning the task K by θ_K.

Following the recently defined taxonomy for CL (Wortsman et al., 2020), we study the settings in which the task-id is provided during training, but not during testing [2], meaning that, during training the model is aware of which domain it is currently learning, but during testing, the model is evaluated **without** specifying the dialogue domain. This assumption makes our CL setting more challenging but more realistic, since during inference times users do not explicitly specify in which domain they want to operate. In this paper, we consider three continual learning approaches:

- *Regularization* methods add a regularization term to the current learned θ_t to avoid interfering with the previously learned θ_{t-1}. Formally, the loss at task t is:

$$L_{\theta_t}(D_t) = L_{\theta_t}(D_t) + \lambda\Omega(\theta_t - \theta^*_{t-1})^2, \quad (4)$$

  where θ^*_{t-1} are copies of the previously learned parameters frozen at this stage. In our experiments, we consider two kind of Ω: the identity function (**L2**) and the Fisher information matrix (Kirkpatrick et al., 2017) (**EWC**).
- *Rehearsal* methods use an episodic memory $\mathcal{M}$ to store examples from the previously learned domains, and re-use them while learning new tasks. The most straightforward method is to add the content of the memory $\mathcal{M}$ to the current task data D_t. Following our notation, the model is optimized using $L_{\theta_t}(D_t + \mathcal{M})$, and we refer to this method as **REPLAY**. Another rehearsal method is to constrain the gradients updates so that the loss of the samples in memory never increases. More formally,

$$L_{\theta_t}(D_t) \text{ s.t. } L_{\theta_t}(\mathcal{M}) \leq L_{\theta_{t-1}}(\mathcal{M}). \quad (5)$$

  Of this kind, the method Gradient Episodic Memory (GEM) (Lopez-Paz and Ranzato, 2017) computes the gradient constraint via a quadratic programming solver that scales with the number of parameters of the model. After our first investigation, we discover that it is impractical for large-language models

[2] GNs: Task Given during train, Not inference; shared labels.

to use GEM, since they have millions of parameters and the constraints are computed for each batch. To cope with this computational complexity, Chaudhry et al. (2018) proposed **A-GEM**, which efficiently computes the gradient constraints while being effective in CL tasks. Finally, a rehearsal method specific to language tasks is **LAMOL** (Sun et al., 2019), which instead of storing samples in $\mathcal{M}$, trains a model that simultaneously learns to solve tasks and generate training samples.

- *Architectural* methods add task-specific parameters to an existing base model for each task. Of this kind, multiple models have been proposed, such as Progressive Net (Rusu et al., 2016), Dynamically Expandable Networks (DEN) (Yoon et al., 2017) and Learn-to-Grow (Li et al., 2019b). On the other hand, there are fixed-capacity methods, that do not add specific parameters, but learn parameter masks (Fernando et al., 2017), usually binary (Mallya et al., 2018), to select subnetworks that are task-specific. To the best of our knowledge, these models have been tested mostly on computer vision tasks, and they can not easily handle our CL setting (i.e., no task-id during testing).

3 AdapterCL

Motivated by the lack of architectural baselines for CL in Seq2Seq modelling, we propose a novel architectural method called AdapterCL. Our proposed method parameterizes each task using residual adapters (Houlsby et al., 2019; Lin et al., 2020a) and uses an entropy-based classifier to select which adapter to use at testing time. This method is designed for large pre-trained language models, e.g., GPT-2, since only the task-specific parameters are trained, while the original weights are left frozen.

Residual adapters are trainable parameters added on top of each transformer layer, which steer the output distribution of a pre-trained model without modifying its original weights. An adapter block consists of layer normalization (Ba et al., 2016), followed by two linear layers (Hinton and Zemel, 1994) with a residual connection. Given the hidden representation at layer l, denoted as $H \in \mathbb{R}^{p \times d}$, of a transformer (Vaswani et al., 2017), where d is the hidden size and p is the sequence length, the residual adapter computes

$$\text{Adapter}_{\mu_i^l}(H) = \text{ReLU}(\text{LN}(x)W_l^E)W_l^D + H, \tag{6}$$

where W_l^E and W_l^D are trainable parameters of dimensions $d \times b$ and $b \times d$ respectively, and $\texttt{LN}(\cdot)$ denotes the layer normalization. The bottleneck dimension b is a tunable hyper-parameter that allows to adjust the capacity of the adapter according to the complexity of the target task. We define the set of $\mu_i = \{W_0^E, W_0^D, \cdots, W_L^E, W_L^D\}$ as the set of parameters for the adapter$_i$ for a model with L layers.

To continuously learn new tasks, we first spawn a new adapter, parameterized by μ, and then we train its parameters as in Equation 3. For instance, given the dataset D_t and the model with its corresponding adapter μ_t, the loss is defined as:

$$\mathcal{L}_{\mu_t}(D_t) = -\sum_{j}^{|D_t|}\sum_{i=0}^{n+m} \log p_{\mu_t}(x_i^{(j)}|x_0^{(j)}, \cdots x_{i-1}^{(j)}). \tag{7}$$

Importantly, the loss is optimized over μ_t to guarantee that each task is independently learned. An high-level representation of AdapterCL is shown in Figure 3.

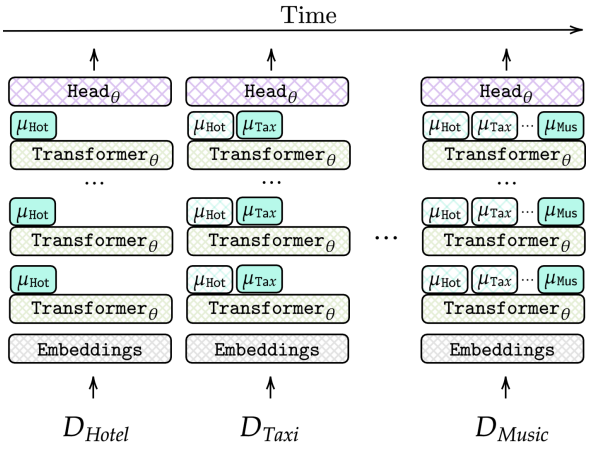

Figure 3: High-level representation of the AdapterCL.

Perplexity-Based Classifier In our CL setting the task-id is provided during training and thus each μ_t is optimized over D_t. During testing, however, the task-id is not provided and thus the model has to predict which adapter to use for accomplishing the task. This step is not required in regularization and rehearsal approaches since a single set of parameters is optimised during training.

Inspired by Wortsman et al. (2020), we propose to utilize the perplexity of each adapter over the

input X as a measure of uncertainty. Thus, by selecting the adapter with the lowest perplexity, we select the most confident model to generate the output sequence. The perplexity of an input sequence $X = x_0, \cdots, x_n$ is defined as

$$\text{PPL}_\theta(X) = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{p_\theta\left(x_i \mid x_0, \cdots, x_{i-1}\right)}} \quad (8)$$

Therefore, given the set of adapters parameterized by $\mu_0, \ldots, \mu_N$, each of which is trained respectively with $D_0, \ldots, D_N$, and an input sample X, we compute:

$$\alpha_t = \text{PPL}_{\mu_t}(X)\ \forall t \in 1, \cdots, N, \quad (9)$$

where each α_t represents the confidence of the adapter t for the input X. The task-id t is thus selected as

$$t^* = \text{argmin}\, \alpha_0, \cdots, \alpha_N \quad (10)$$

The perplexity-based selector requires a linear number of forwards with respect to the number of adapters (Equation 9), but it has the advantage of not requiring a further classifier, which itself would suffer from catastrophic forgetting. In Section 5.1, we analyze the time required for the adapter selection.

4 Experimental Settings

In this section we describe 1) the datasets used for creating the learning curriculum, 2) the evaluation metric used to evaluate the different settings, and 3) the experimental setups.

4.1 Datasets

To the best of our knowledge, there is no benchmark for CL in dialogue systems with a high number of tasks to be learned sequentially and with multiple training settings. The closest to ours is the work of Mi et al. (2020), which continuously learns five domains in the NLG setting. In general, NLP benchmarks for continual learning use no more than 10 tasks or domains (Sun et al., 2019; d'Autume et al., 2019). Consequently, in this paper, we propose a CL benchmark by jointly pre-processing four task-oriented datasets: Task-Master 2019 (TM19) (Byrne et al., 2019), Task-Master 2020 (TM20) (Byrne et al., 2019), Schema Guided Dialogue (SGD) (Rastogi et al., 2019) and MultiWoZ (Budzianowski et al., 2018). This results in a curriculum of 37 domains to be learned continuously under four settings: INTENT classification, DST, NLG, and finally end2end (E2E). This is possible because the four datasets provide the speech act annotation for both the user and the system turns, and the dialogue state as well. To avoid any domain overlapping during learning, we select only dialogues with a single domain at a time for 1) having a controlled setting to better studying the continual learning problem in ToDs and 2) having a long and diverse curriculum (37 domains, which is x8 larger than any existing previous benchmark) instead of fewer domains but mixed. Given the difficulty of the problem, we also believe this is a starting point for stimulating new research in CL for ToDs.

Finally, the datasets are pre-processed as in Section 2.1 to form the four settings, and the main statistics are shown in Table 5 in the appendix. In Appendix Table 4, we report the number of samples by each domain and setting, where we notice the that the domains are hugely imbalanced, with sample sizes ranging from a few hundred samples (e.g., SGD-travel) to 15K (e.g., TM19-flight). Importantly, since not all the datasets provide a delexicalized version of the responses, we decide to keep all the datasets in their plain text form.

4.2 Evaluation Metrics

Automatic evaluations for E2E task-oriented dialogue systems are challenging, especially for the response generation task. To overcome this issue, in this paper we use well-defined metrics based on the three modularized settings. In all of the three sub-tasks, we define the relevant metrics as:

- *INTENT* recognition is evaluated using the *accuracy* between the generated intents and the gold labels.
- *DST* is evaluated with the Joint Goal Accuracy (JGA) (Wu et al., 2019) over the gold dialogue states.
- *NLG* is evaluated using both the BLEU score (Papineni et al., 2002) and the slot error rate (EER) (Wen et al., 2015) which is computed as the ratio between the total number of slots and the values not appearing in the response. In datasets such as SGD, the slot has binary values, e.g., yes or no, and thus we exclude these from the count, as in Kale and Rastogi (2020).

Independently of these metrics, we also compute CL-specific metrics such as the average metrics

				INTENT	DST	NLG	
Method	**+Param.**	**Mem.**	**Hours↓**	*Accuracy↑*	*JGA↑*	*EER↓*	*BLEU↑*
VANILLA	-	$\emptyset$	0.21 ± 0.02	4.1 ± 1.4	4.91 ± 4.5	48.7 ± 3.9	6.38 ± 0.6
L2	$\|\theta\|$	$\emptyset$	0.56 ± 0.06	3.8 ± 1.4	3.81 ± 3.4	55.7 ± 7.1	5.4 ± 0.9
EWC	$2\|\theta\|$	$\emptyset$	0.91 ± 0.10	3.9 ± 1.3	5.22 ± 4.5	58.2 ± 3.7	5.06 ± 0.5
AGEM	-	$t\|\mathcal{M}\|$	0.38 ± 0.04	34.0 ± 6.4	6.37 ± 4.0	62.1 ± 6.9	4.54 ± 0.6
LAMOL	-	$\emptyset$	2.32 ± 1.24	7.5 ± 6.4	4.55 ± 3.5	66.1 ± 6.9	3.0 ± 0.9
REPLAY	-	$t\|\mathcal{M}\|$	0.62 ± 0.23	81.1 ± 1.4	30.33 ± 1.2	$\mathbf{17.8} \pm 0.9$	$\mathbf{17.4} \pm 0.7$
ADAPT	$t\|\mu\|$	$\emptyset$	$\mathbf{0.20} \pm 0.02$	$\mathbf{90.5} \pm 0.6$	$\mathbf{35.1} \pm 0.5$	31.78 ± 1.3	16.76 ± 0.4
MULTI	-	-	4.14 ± 2.23	95.5 ± 0.1	48.9 ± 0.2	12.56 ± 0.2	23.61 ± 0.1

Table 1: E2E results in terms of Intent Accuracy, Joint Goal Accuracy (JGA), Slot Error Rate (EER) and BLUE. +Param. shows the additional number of parameters per task (θ base model and μ task-specific parameters), and Mem. the episodic memory size (denoted as $|\mathcal{M}|$) needed per task, and Hours is the average hours per epoch on a single NVIDIA 2080Ti required for training a new domain (Figure 6 for more details).

through time (Avg. Metric), as in Lopez-Paz and Ranzato (2017). We consider access to the test set for each of the T tasks, and after the model finishes learning the task t_i, we evaluate its test performance on all tasks in the curriculum. To elaborate, we construct the matrix $R \in \mathbb{R}^{T \times T}$, where $R_{i,j}$ is the test metric (e.g., BLEU, JGA) of the model on task t_j after observing the last sample from task t_i. Then we define the average accuracy as

$$\text{Avg.Metric} = \frac{1}{T} \sum_{i=1}^{T} R_{T,i} \qquad (11)$$

The Avg. Metric score is useful for understanding the learning dynamics through time of different baselines. Further metrics such as Backward-Transfer and Forward-Transfer (Lopez-Paz and Ranzato, 2017) are available to distinguish baselines with similar Avg. Metric scores, but in this paper we limit our evaluation to this metric, since there is a large gap among the baselines. Finally, to evaluate the adapter selection, we use the accuracy over the gold task-id.

4.3 Baselines and Settings

The main goal of this paper is to compare the performance of different CL approaches and to understand the trade-offs between them. Therefore, following the definition provided in Section 2.2, we compare 1) EWC and L2, 2) A-GEM, LAMOL, and REPLAY, and 3) AdapterCL. Additionally, we provide baselines trained on each task continuously, namely, VANILLA, without any regularization or memory, and a multitask baseline (MULTI), which is trained on all the data in the curriculum at the same time. In L2, EWC, and A-GEM we tune

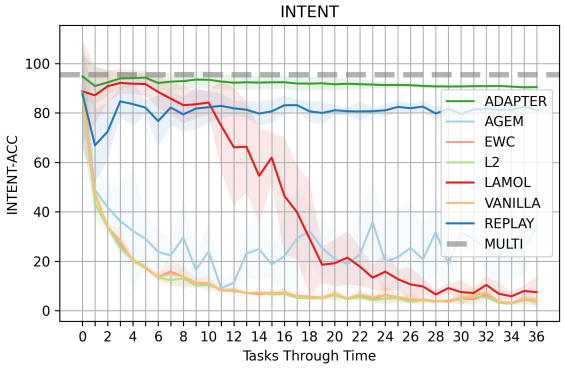

Figure 4: Avg. Metric for the Intent Accuracy in the E2E setting.

different λ in the range 0.0001 to 100, and in *rehearsal*-based methods, such as REPLAY and GEM, we keep 50 samples per task, for a total of 1,850 sample in $\mathcal{M}$ at the end of the curriculum. This is particularly important since if we store in memory all the samples of the seen tasks, the model would incur a high training cost. Arguably, this could be an option if the per-task sample size is small, but this is not always possible, e.g, large language models (Brown et al., 2020). Therefore, the assumption of minimizing the number of samples in memory is valid and widely used in the CL literature (Mi et al., 2020). Finally, for the AdapterCL, we tune the bottleneck size b between 10, 50, 100, and 200. Interested readers can refer to the Appendix for further details of the selected hyper-parameters. In continual learning the model is not able decide the order of tasks. Therefore, we create five learning curricula by randomly permuting the 37 tasks.

5 Results & Analysis

The main results in the E2E setting are summarized in Table 1, while the results for the modularized settings are in Table 2 in the Appendix. Due to space constraints in these tables, we report the Avg. Metric at the end of the curriculum, which is equivalent to the average test set performance in all the tasks, and the resources used by each model. The results on the full curriculum of 37 tasks are reported in Figure 4 and 5.

Main Results From the tables, we can observe that 1) both regularization-based methods (L2/EWC) and some rehearsal-based methods (AGEM/LAMOL) cannot continually learn tasks without incurring in catastrophic forgetting, 2) REPLAY and AdapterCL perform comparably well on the Intent and DST tasks, 3) REPLAY works the best on the NLG task, showing that transferring knowledge between tasks is needed, and 4) no CL methods can reach the performance of the multi-task baseline, especially on the DST task. In addition, the adapter selection accuracy based on Equation 10 is 95.44±0.2% in E22, 98.03±0.1% in Intent Recognition, 98.19±0.1% in DST, and 93.98±0.1% in the NLG.

Although these numbers are meaningful, they do not describe the entire learning history of the curriculum. To better understand these dynamics, we plot the Avg. Metric in Equation 11 after each task is learned ($t = T$ in the equation). Figure 4, 5 shows the plot for two of the considered metrics and all the baselines. From this figure we can better understand how REPLAY and AdapterCL outperform the other baselines and, interestingly, that LAMOL performs as well as REPLAY on the first 12 tasks. This is because LAMOL learns to generate training samples instead of using an explicit memory, and thus the generation becomes harder when more and more task are shown. This result further strengthens our motivation to have a benchmark with a long curriculum. In the Appendix, Figure 13 and 14 show the remaining two metrics for the E2E setting and Figure 16, 17, 18, and 19 show the same plots for the individual module training.

5.1 Training Time Analysis

From Figure 6 we plot the training time (Hours × Epochs) required to add a new domain to an existing model. A clear trend is shown where rehearsal based methods (REPLAY, LAMOL) requires a linearly increasing amount of time to add new domains, while in AdapterCL and VANILLA the time remains constant across the curriculum. This is even more evident when the entire training-set of all the previous tasks is used for training (REPLAY-ALL), which lead to an expensive re-training process to add new domains. The average time across domain for all the baseline is shown in Table 1. AdapterCL requires also an additional cost in selecting which parameters to use during testing. By using a single NVIDIA 2080ti, the average time to select the adapter is $0.069 \pm 0,003$ seconds, which is as expensive as decoding 4 tokens.

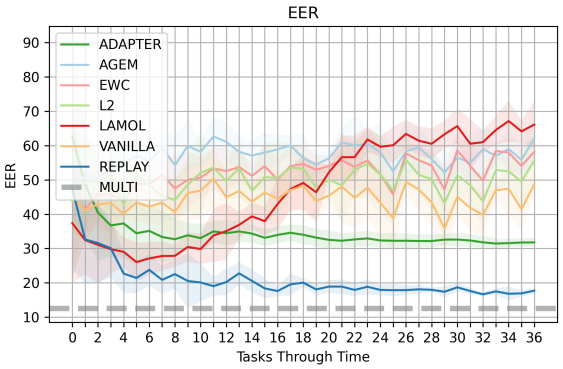

Figure 5: Avg. Metric for the EER in the E2E setting.

5.2 No Free Lunch

Finally, based on the results shown in Table 1, and especially based on the resources used by each method, we conclude that there is a no free lunch in terms of resources needed to avoid the catastrophic forgetting problem. To elaborate, in both REPLAY and AdapterCL, the resources used grow linearly with the number of tasks; i.e., in REPLAY the number of samples stored in the episodic memory grows linearly (50 times the number of tasks), and in AdapterCL the number of parameters grows linearly (number of adapter parameters times the number of tasks). Figure 11 in the Appendix describes the high-level intuition behind this concept by plotting the number of tasks and parameters and the episodic memory sizes needed.

5.3 Analysis: Episodic Memory Size

In this section, we analyze the effect of increasing the episodic memory size for the REPLAY method. Trivially, by including all the training samples in the memory, the model, at the last task, converges to the multitask baseline. Then, the question of how many samples to keep per task to avoid catastrophic forgetting is important. In light of this, Figure 7

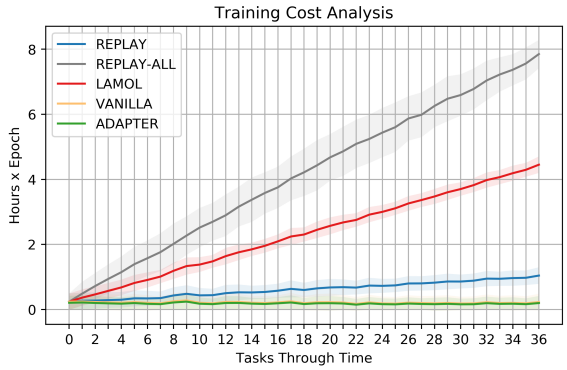

Figure 6: Training time for adding a new domain for different CL techniques.

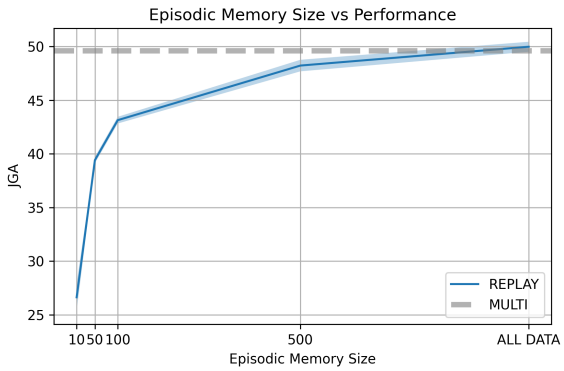

Figure 7: Ablation study on the size of the episodic-memory vs JGA.

shows the performance of the model at different episodic memory sizes on the DST task. Here, we observe that by storing only a few samples per task (10-50) the model still greatly suffers from catastrophic forgetting, where with around 500 samples, which is equivalent to a total of 18,500 samples in our setting, the performance is closer to that of the multitask baseline (i.e., a possible upper bound). Similar observations are shown for the other two tasks in Figure 8, 9, and 10 in the Appendix.

6 Related Work

Continual learning methods are usually developed and benchmarked on computer visions tasks. Interested readers may refer to Mundt et al. (2020); Parisi et al. (2019); De Lange et al. (2019) for an overview of the existing approaches, and to Section 2.2 for more details on the three main CL approaches studied in this paper. Continual learning has also been studied in the Long Life Learning (LLL) scenario, where a learner continuously accumulates knowledge and makes use of it in the future (Chen and Liu, 2018; Liu and Mei, 2020). In this paper, we study the setting in which a series of tasks is learned continuously.

CL in NLP has been explored for both classification (d'Autume et al., 2019; Sprechmann et al., 2018; Wang et al., 2020) and generation (Sun et al., 2019; Hu et al., 2020) tasks. For instance, Sun et al. (2019); Chuang et al. (2020) proposed LAMOL, which we use as our baseline, and studied its effectiveness on a subset of DecaNLP (McCann et al., 2018). On the other hand, the work of d'Autume et al. (2019); Sprechmann et al. (2018) is not suitable for interactive systems as dialogue systems, since their methods require local adaptation (i.e., a fine-tuning step) during inference. Finally, continual learning has been used for sentence encoding (Liu et al., 2019), composition language learning (Li et al., 2019c) and relation learning (Han et al., 2020). However, these methods are specific to particular applications not generalizable to ToDs.

CL in Dialogue Systems The very early work on CL for Task-Oriented dialogue is from Lee (2017), who used EWC to avoid catastrophic forgetting on three domains learned sequentially. Continual learning has also been studied in the NLG setting, where a single model was trained to learn one domain at the time in MWoZ (Mi et al., 2020). The authors used episodic memory to replay the example in combination with EWC. In this paper, we compare similar baselines but on a larger benchmark that also includes MWoZ and the NLG setting. For the DST setting, CL was studied by (Wu et al., 2019) using MWoZ, where several baselines such as L2, EWC and GEM were compared. Differently, Li et al. (2019a) leveraged CL for evaluating the quality of chat-bot models, and He et al. (2019) studied the catastrophic forgetting problem in chit-chat systems. Finally, Shuster et al. (2020) showed that by training models on humans-machine conversations in an open-domain fantasy world game (Fan et al., 2020) the models progressively improved.

7 Conclusion

In this paper, we proposed a benchmark for continual learning in task-oriented dialogue systems, with 37 tasks to be learned sequentially on four settings: intent recognition, dialogue state tracking, natural language generation, and end-to-end. Then, we implemented three continual learning methodologies, namely regularization, rehearsal and architectural.

For the latter, we proposed a simple yet effective method based on residual adapters and a perplexity-based classifier to select which adapter to use at inference time. Finally, we analyzed the trade-off between the performance, the number of parameters, training time and the episodic memory sizes of the evaluated baselines.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Ultes Stefan, Ramadan Osman, 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 (EMNLP)*.

Bill Byrne, Karthik Krishnamoorthi, Saravanan Ganesh, and Mihir Sanjay Kale. 2020. Tickettalk: Toward human-level performance with end-to-end, transaction-based dialog systems. *arXiv preprint arXiv:2012.12458*.

Bill Byrne, Karthik Krishnamoorthi, Chinnadhurai Sankar, Arvind Neelakantan, Daniel Duckworth, Semih Yavuz, Ben Goodrich, Amit Dubey, Kyu-Young Kim, and Andy Cedilnik. 2019. Taskmaster-1:toward a realistic and diverse dialog dataset. In *2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing*, Hong Kong.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2018. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*.

Zhiyuan Chen and Bing Liu. 2018. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207.

Yung-Sung Chuang, Shang-Yu Su, and Yun-Nung Chen. 2020. Lifelong language knowledge distillation. *arXiv preprint arXiv:2010.02123*.

Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic memory in lifelong language learning. *arXiv preprint arXiv:1906.01076*.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2019. A continual learning survey: Defying forgetting in classification tasks. *arXiv preprint arXiv:1909.08383*.

Angela Fan, Jack Urbanek, Pratik Ringshia, Emily Dinan, Emma Qian, Siddharth Karamcheti, Shrimai Prabhumoye, Douwe Kiela, Tim Rocktäschel, Arthur Szlam, et al. 2020. Generating interactive worlds with text. In *AAAI*, pages 1693–1700.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. 2017. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*.

Xu Han, Yi Dai, Tianyu Gao, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2020. Continual relation learning via episodic memory activation and reconsolidation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6429–6440.

Tianxing He, Jun Liu, Kyunghyun Cho, Myle Ott, Bing Liu, James Glass, and Fuchun Peng. 2019. Mix-review: Alleviate forgetting in the pretrain-finetune framework for neural language generation models. *arXiv preprint arXiv:1910.07117*.

Geoffrey E Hinton and Richard S Zemel. 1994. Autoencoders, minimum description length and helmholtz free energy. In *Advances in neural information processing systems*, pages 3–10.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *arXiv preprint arXiv:2005.00796*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. *arXiv preprint arXiv:1902.00751*.

Hexiang Hu, Ozan Sener, Fei Sha, and Vladlen Koltun. 2020. Drinking from a firehose: Continual learning with web-scale natural language. *arXiv preprint arXiv:2007.09335*.

Mihir Kale and Abhinav Rastogi. 2020. Few-shot natural language generation by rewriting templates. *arXiv preprint arXiv:2004.15006*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Sungjin Lee. 2017. Toward continual learning for conversational agents. *arXiv preprint arXiv:1712.09943*.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1437–1447.

Lu Li, Zhongheng He, Xiangyang Zhou, and Dianhai Yu. 2019a. How to evaluate the next system: Automatic dialogue evaluation from the perspective of continual learning. *arXiv preprint arXiv:1912.04664*.

Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. 2019b. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. *arXiv preprint arXiv:1904.00310*.

Yuanpeng Li, Liang Zhao, Kenneth Church, and Mohamed Elhoseiny. 2019c. Compositional language continual learning. In *International Conference on Learning Representations*.

Zhaojiang Lin, Andrea Madotto, and Pascale Fung. 2020a. Exploring versatile generative language model via parameter-efficient transfer learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 441–459.

Zhaojiang Lin, Andrea Madotto, Genta Indra Winata, and Pascale Fung. 2020b. Mintl: Minimalist transfer learning for task-oriented dialogue systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3391–3405.

Zhaojiang Lin, Andrea Madotto, Genta Indra Winata, Peng Xu, Feijun Jiang, Yuxiang Hu, Chen Shi, and Pascale Fung. 2021. Bitod: A bilingual multi-domain dataset for task-oriented dialogue modeling. *arXiv preprint arXiv:2106.02787*.

Bing Liu and Chuhe Mei. 2020. Lifelong knowledge learning in rule-based dialogue systems. *arXiv preprint arXiv:2011.09811*.

Tianlin Liu, Lyle Ungar, and João Sedoc. 2019. Continual learning for sentence representations using conceptors. *arXiv preprint arXiv:1904.09187*.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *Advances in neural information processing systems*, pages 6467–6476.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. 2018. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Fei Mi, Liangwei Chen, Mengjie Zhao, Minlie Huang, and Boi Faltings. 2020. Continual learning for natural language generation in task-oriented dialog systems. *arXiv preprint arXiv:2010.00910*.

Martin Mundt, Yong Won Hong, Iuliia Pliushch, and Visvanathan Ramesh. 2020. A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning. *arXiv preprint arXiv:2009.01797*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

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

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2020. Soloist: Few-shot task-oriented dialog with a single pre-trained auto-regressive model. *arXiv preprint arXiv:2005.05298*.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2019. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *arXiv preprint arXiv:1909.05855*.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Kurt Shuster, Jack Urbanek, Emily Dinan, Arthur Szlam, and Jason Weston. 2020. Deploying lifelong open-domain dialogue learning. *arXiv preprint arXiv:2008.08076*.

Pablo Sprechmann, Siddhant M Jayakumar, Jack W Rae, Alexander Pritzel, Adria Puigdomenech Badia, Benigno Uria, Oriol Vinyals, Demis Hassabis, Razvan Pascanu, and Charles Blundell. 2018. Memory-based parameter adaptation. *arXiv preprint arXiv:1802.10542*.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2019. Lamol: Language modeling for lifelong language learning. In *International Conference on Learning Representations*.

Sebastian Thrun and Lorien Pratt. 2012. *Learning to learn*. Springer Science & Business Media.

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*, pages 5998–6008.

Zirui Wang, Sanket Vaibhav Mehta, Barnabás Póczos, and Jaime Carbonell. 2020. Efficient meta lifelong-learning with limited memory. *arXiv preprint arXiv:2010.02500*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. 2020. Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33.

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. *arXiv preprint arXiv:1905.08743*.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. 2017. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*.

8 Appendix

Hyper-Parameters All the experiments uses GPT-2 small (117M parameters). In all the setting and baselines, we run a small grid-search over several hyper-parameters. In VANILLA and MULTI, we used a learning rate of 0.001 with a warm up schedule and 10 epochs with an early stopping over the validation set. In L2, EWC we tune different λ in the range 0.0001 to 100, resulting in $\lambda = 0.001$. In A-GEM, we tune λ also in the range 0.0001 to 100, resulting in $\lambda = 1$. In REPLAY, we use same setting as VANILLA and MULTI, and we tune different Episodic Memory $\mathcal{M}$ in the range 1, 50, 100, 500 and all data sample per task. We select 50 sample per task for a good balance between memory and performance, and we ablate over the memory size. In AdapterCL, we tune the bottleneck size b between 10, 50, 100, and 200, and we select 50 for the modularized settings (INTENT, DST, NLG) and 100 for the E2E. For an adapter with a 50 bottleneck size we add 2.5% additional parameters per task while with 100 bottleneck size we add 5% additional parameters.

Link for downloading the data The dataset are available online at:

- MWOZ: https://github.com/budzianowski/multiwoz
- SGD: https://github.com/google-research-datasets/dstc8-schema-guided-dialogue
- Task Master: https://github.com/google-research-datasets/Taskmaster

Computing Infrastructure All of the experiments have been run on a single Nvidia-V100 32gb.

			INTENT	DST	NLG	
Method	**+Parm.**	**Mem.**	*Accuracy* ↑	*JGA* ↑	*EER* ↓	*BLEU* ↑
VANILLA	-	$\emptyset$	3.27 ± 0.3	5.34 ± 4.4	14.81 ± 7.7	11.06 ± 2.9
L2	$\|\theta\|$	$\emptyset$	3.52 ± 0.7	4.95 ± 4.4	12.93 ± 5.5	11.99 ± 1.4
EWC	$2\|\theta\|$	$\emptyset$	3.21 ± 0.3	5.36 ± 4.3	14.2 ± 6.2	11.19 ± 2.4
AGEM	-	$t\|M\|$	9.74 ± 2.6	5.17 ± 4.0	34.2 ± 8.6	5.51 ± 2.1
LAMOL	-	$\emptyset$	3.73 ± 1.0	4.03 ± 3.9	29.61 ± 3.1	5.42 ± 1.7
REPLAY	-	$t\|M\|$	76.45 ± 1.5	**39.42** ± 0.2	**4.95** ± 1.5	**21.72** ± 0.3
ADAPT	$t\|\mu\|$	$\emptyset$	**85.05** ± 0.6	37.9 ± 0.6	14.36 ± 0.7	21.48 ± 0.2
MULTI	-	-	87.50 ± 0.2	50.04 ± 0.1	2.84 ± 0.2	26.15 ± 0.2

Table 2: Modularized Results.

	INTENT	DST	NLG	
$\|\mathcal{M}\|$	*Accuracy* ↑	*JGA*↑	*EER*↓	*BLEU* ↑
10	57.286±3.80	26.63±1.26	8.44±0.97	18.86±0.68
50	76.446±1.55	39.41±0.28	6.63±0.53	21.11±0.41
100	81.496±0.86	43.13±0.31	5.75±0.19	21.71±0.25
500	85.91±0.55	48.22±0.53	4.96±0.10	22.86±0.25
ALL	87.784±0.16	49.97±0.46	4.36±0.24	23.85±0.12
MULTI	87.5±0.1	50.04 ± 0.6	3.42±0.1	26.15±0.1

Table 3: Ablation study over episodic memory size $|\mathcal{M}|$. In the table $|\mathcal{M}|$ represents the number of samples per task kept in memory.

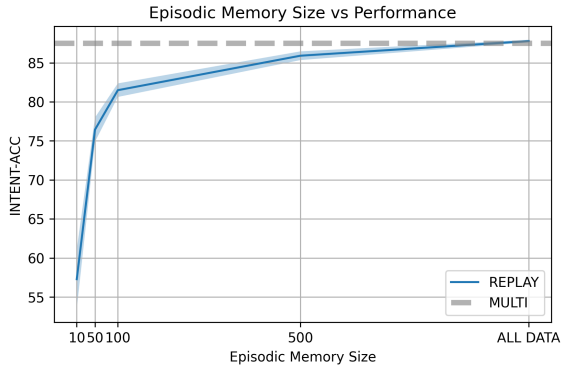

Figure 8: Ablation study on the size of the episodic-memory vs the intent accuracy.

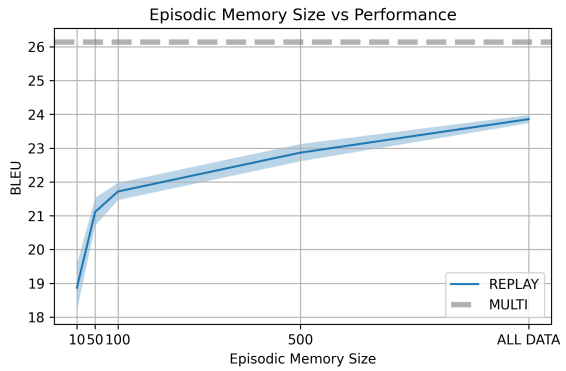

Figure 9: Ablation study on the size of the episodic-memory vs BLEU.

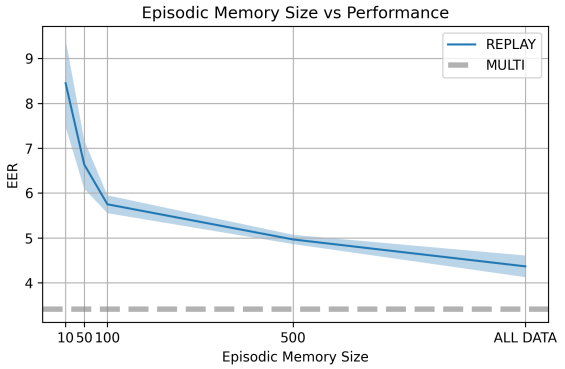

Figure 10: Ablation study on the size of the episodic-memory vs EER.

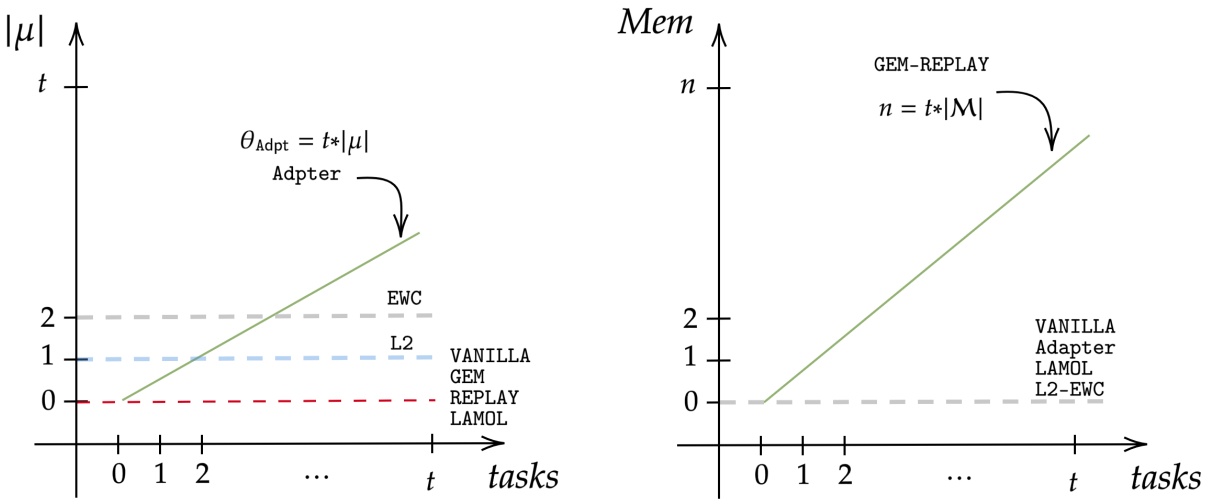

Figure 11: "No free lunch" in CL. Plot of the trade-off between number of parameters added per task and the size of the episodic memory $\mathcal{M}$.

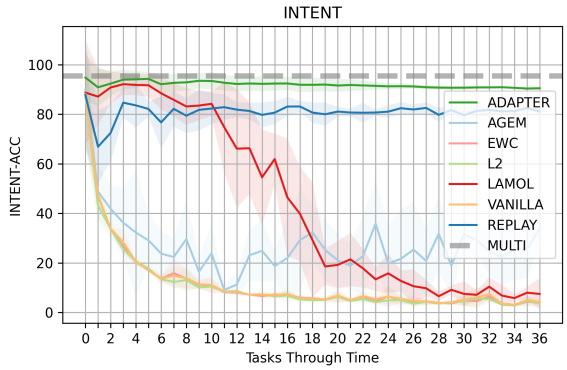

Figure 12: Avg. Metric for the Intent Accuracy in the E2E setting.

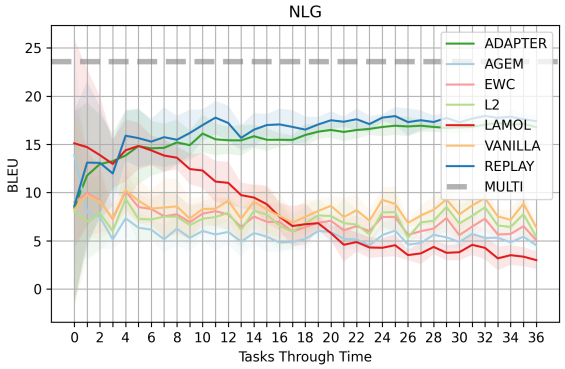

Figure 14: Avg. Metric for the response generation, BLUE in the E2E setting.

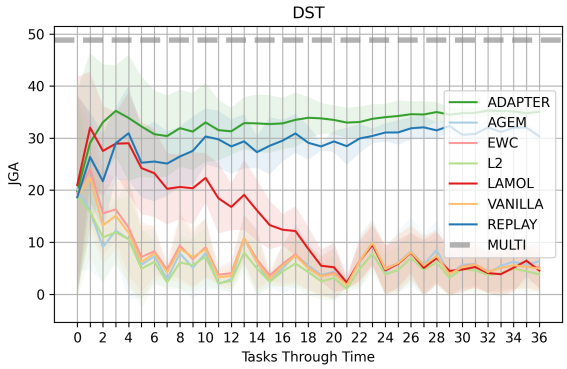

Figure 13: Avg. Metric for the JGA in the E2E setting.

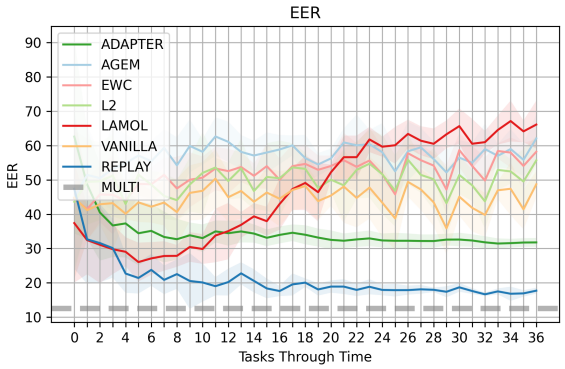

Figure 15: Avg. Metric for the response generation, EER in the E2E setting.

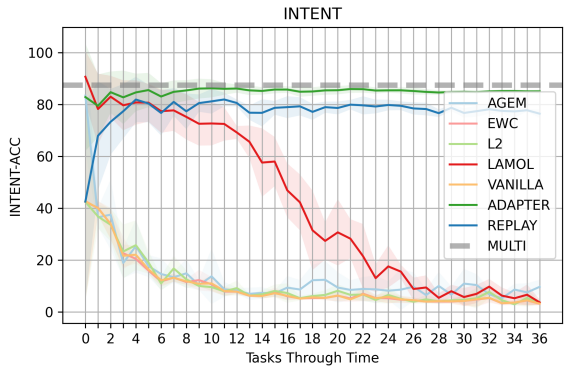

Figure 16: Avg. Metric for the Intent Accuracy in the modularized setting (INTENT).

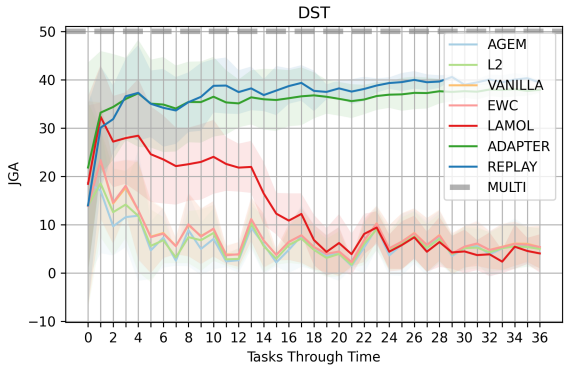

Figure 17: Avg. Metric for the JGA in the modularized setting (DST).

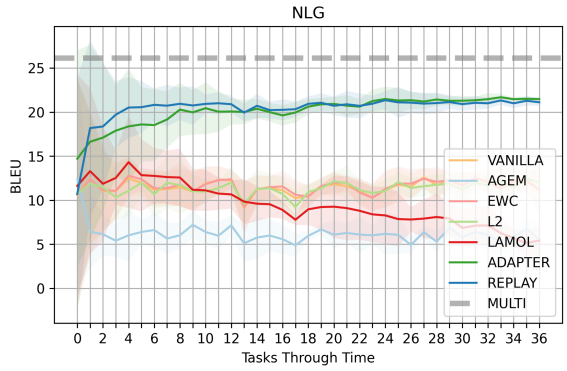

Figure 18: Avg. Metric for the BLUE in the modularized setting (NLG).

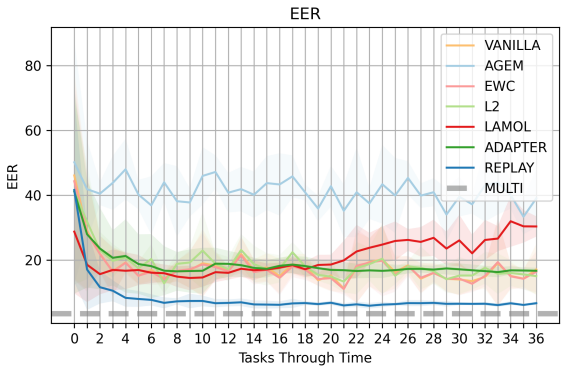

Figure 19: Avg. Metric for the EER in the modularized setting (NLG).

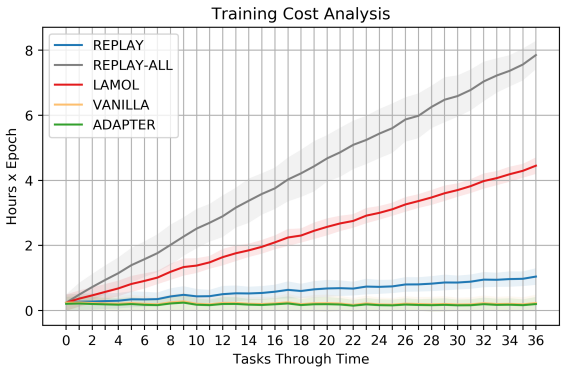

Figure 20: Training time analysis for adding new domains including all the baselines.

Domains	DST-INTENT			NLG			End-to-End		
	Train	*Dev*	*Test*	*Train*	*Dev*	*Test*	*Train*	*Dev*	*Test*
TM19 movie	4733	584	500	3010	366	341	12766	1632	1481
TM19 auto	3897	448	522	2128	223	283	10918	1248	1443
TM19 restaurant	4434	568	561	2582	330	333	12862	1669	1630
TM19 pizza	2883	381	359	1326	171	171	8720	1145	1083
TM19 uber	4378	535	525	2418	290	278	11331	1362	1361
TM19 coffee	2591	302	335	1381	151	184	7429	894	936
TM20 flight	15868	1974	1940	10148	1272	1245	36778	4579	4569
TM20 food-ordering	3404	411	431	2394	277	287	7838	941	986
TM20 hotel	15029	1908	1960	6590	842	869	35022	4400	4532
TM20 music	5917	764	769	4196	537	523	13723	1773	1787
TM20 restaurant	13738	1761	1691	8356	1063	994	34560	4398	4297
TM20 sport	13072	1668	1654	12044	1553	1542	29391	3765	3723
TM20 movie	13221	1703	1567	9406	1203	1093	32423	4158	3881
MWOZ taxi	1239	234	194	402	71	56	2478	468	388
MWOZ train	1452	158	160	563	63	59	2905	316	320
MWOZ restaurant	5227	243	281	3333	141	177	10461	486	563
MWOZ hotel	2798	289	385	1924	194	258	5602	579	771
MWOZ attraction	484	43	42	295	27	26	975	86	85
sgd restaurants	2686	278	616	1720	166	386	5756	606	1354
sgd media	1411	230	458	988	167	324	3114	502	1005
sgd events	4881	598	989	3241	389	590	10555	1317	2197
sgd music	1892	275	556	1506	224	464	4040	597	1215
sgd movies	1665	181	52	996	114	44	3760	420	126
sgd flights	4766	1041	1756	2571	627	982	10429	2244	3833
sgd ridesharing	652	85	187	377	48	107	1448	188	418
sgd rentalcars	1510	250	469	865	153	280	3277	538	1009
sgd buses	1862	331	653	1102	218	412	4050	709	1393
sgd hotels	3237	394	948	1997	243	597	6983	858	2053
sgd services	3328	360	926	2225	230	611	7262	803	2016
sgd homes	2098	170	533	1312	96	338	4519	394	1158
sgd banks	1188	139	293	723	84	181	2599	319	667
sgd calendar	592	115	236	397	65	133	1313	246	501
sgd alarm	212	34	91	221	30	74	580	82	198
sgd weather	196	32	80	123	23	59	433	70	169
sgd travel	186	23	48	121	14	30	420	53	106
sgd payment	227	21	51	143	14	32	497	44	113
sgd trains	300	73	128	149	43	66	668	158	274
Total	147254	18604	22946	93273	11722	14429	347885	44047	53641

Table 4: All data samples used in the experiments.

Name	Train	Valid	Test	Dom.	Intent	Turns
TM19	4,403	551	553	6	112	19.97
TM20	13,839	1,731	1,734	7	128	16.92
MWoZ	7,906	1,000	1,000	5	15	13.93
SGD	5,278	761	1,531	19	43	14.71
Total	31,426	4,043	4,818	37	280	16.23

Table 5: Main dataset statistics.