Lifelong Sequence Generation with Dynamic Module Expansion and Adaptation

Anonymous ACL submission

Abstract

Lifelong sequence generation (LSG), a problem in continual learning, aims to continually train a model on a sequence of generation tasks to learn constantly emerging new generation patterns while avoiding the forgetting of previous knowledge. Existing LSG methods mainly focus on maintaining old knowledge while paying little attention to knowledge transfer across tasks. In contrast, humans can better learn new tasks by leveraging previously acquired knowledge from similar tasks. Inspired by the learning paradigm of humans, we propose Dynamic Module Expansion and Adaptation (DMEA), which enables the model to dynamically determine the architecture for acquiring new knowledge based on task correlation and select the most similar previous tasks to facilitate adaptation to new tasks. In addition, as the learning process can easily be biased towards the current task which might cause more severe forgetting of previously learned knowledge, we propose dynamic gradient scaling to balance the learning of the current task and replayed tasks. With extensive experiments, we demonstrate that DMEA can consistently outperform existing methods in different LSG settings.

1 Introduction

With the recent advancements in pre-trained language models (LMs), current sequence generation methods have achieved impressive performance on a variety of generation tasks (Radford et al., 2019; Raffel et al., 2020). Typically, these models are trained on a fixed corpus, assuming the underlying data distribution to be static (Ham et al., 2020; El-Kassas et al., 2021). However, real cognitive tasks are generally more complex involving changing contexts and dynamic environments. The ever-changing data distribution causes the models to face challenges in acquiring new knowledge, while retaining the prior knowledge. Speaking about what is next for NLP, Kathleen McKeown in a recent interview said: "Most models are static. But the world changes every minute, every second. Dealing with a dynamic world is a new area that's up and coming." (Source)

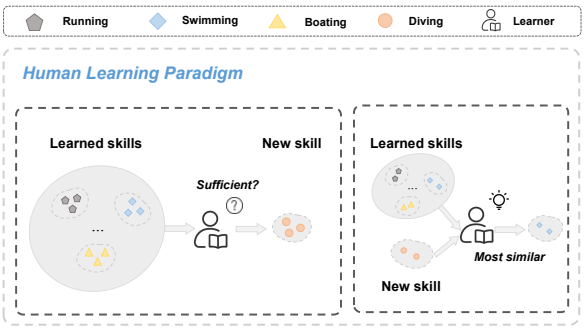

Figure 1: Illustration of human learning. Given three learned skills, *i.e.,* swimming, running and boating, humans can determine that these skills are *not* sufficient for diving. And after realizing that swimming is the most similar learned skill, they only need to learn the new aspect, *i.e.,* how to safely jump off the diving platform to master the new diving skill.

A potential solution is to formalize sequence generation as lifelong sequence generation or LSG (Sun et al., 2020), where the model is expected to learn sequentially from a stream of generation tasks with potentially different data distributions. In such cases of distribution shift, the model might forget previously acquired knowledge upon learning new tasks, a phenomenon known as *catastrophic forgetting* (McCloskey and Cohen, 1989). Previous LSG methods (Mi et al., 2020; Sun et al., 2020; Madotto et al., 2021) mainly explore different ways to alleviate forgetting. Recently, Zhang et al. (2022) propose Adaptive Compositional Modules (ACM) which dynamically adds modules for new tasks depending on whether there are reusable previous modules, achieving SOTA performance on LSG.

Despite its effectiveness, ACM has several key limitations. First, it pays little attention to knowledge transfer across tasks which is as important for continual learning as preventing forgetting (Ke

et al., 2020). In fact, a hallmark of human intelligence is that humans can better learn new tasks by leveraging previously acquired knowledge from similar tasks (Lake et al., 2017). They can not only determine whether previously acquired skills are sufficient to solve a new task, but also exploit the most similar learned skills to facilitate the learning of the task; see Fig. 1 for an illustration. Second, ACM does not consider the correlation between learned tasks and the new task when adding modules, which might hinder finding the optimal architecture. Finally, the learning process in ACM can be biased towards the new task as the gradient norm of the new task on reused modules is typically much larger than that of replayed tasks, which may affect previously acquired knowledge.

Inspired by the learning paradigm of humans and to address the above limitations of ACM, in this work we propose Dynamic Module[1] Expansion and Adaptation (DMEA). We divide the learning process of a new task into three stages: expansion, selection and adaptation. In the expansion stage, DMEA determines whether to reuse modules of previous tasks or insert new modules for learning novel knowledge. Inspired by Zhang et al. (2022), it utilizes differentiable architecture search (Liu et al., 2019) to enable the model to dynamically determine the architecture for solving the new task. To leverage the correlation between tasks, the learnable coefficients in architecture search are initialized based on the cosine similarity of word frequency distributions between learned tasks and the new task. In the selection stage, DMEA selects the top-K most similar previous tasks through input subspace (Lin et al., 2022b). Finally, in the adaptation stage, it utilizes the selected similar tasks to facilitate adaptation to the new task. The output of selected similar tasks is fused with that of the new task using learnable coefficients in every transformer layer to enable forward knowledge transfer. This is indeed an instance of mixture-of-experts (Masoudnia and Ebrahimpour, 2014).

In addition, when the model learns a new task, DMEA also incorporates pseudo-sample replay (Sun et al., 2020) to further mitigate catastrophic forgetting. To address the "bias to the new task" in the gradient update, we introduce dynamic gradient scaling to balance the learning of the new task and replayed tasks. To verify the effectiveness of DMEA, we conduct extensive experiments on various generation tasks in different LSG settings. The empirical results show that DMEA can consistently outperform previous state-of-the-art baselines.

In summary, our main contributions are:

- To the best of our knowledge, we are the first to explore solving LSG from the perspective of human learning. We propose DMEA, a novel method based on dynamic module expansion and adaptation, to alleviate catastrophic forgetting and facilitate knowledge transfer in LSG.
- With extensive experiments and analysis, we demonstrate the effectiveness of our method compared to existing ones in different LSG settings. Our code base is available at `<redacted>`.

2 Related Work

Lifelong Learning (LL) aims to continually learn knowledge from a sequence of tasks with different distributions. The goal is twofold: alleviate *catastrophic forgetting* (McCloskey and Cohen, 1989) of learned tasks, and facilitate knowledge transfer (Lopez-Paz and Ranzato, 2017) across tasks.

Catastrophic forgetting typically means that the model forgets previously acquired knowledge after learning new tasks. Prior LL methods mainly focus on mitigating this problem and can be divided into three categories. First, *regularization-based* methods constrain the update of parameters that are important to learned tasks to retain previous knowledge (Kirkpatrick et al., 2017; Li and Hoiem, 2017; Zenke et al., 2017; Ritter et al., 2018). Second, *architecture-based* methods dynamically adjust the model architecture to acquire new information while preventing the forgetting of previously learned tasks (Rusu et al., 2016; Chen et al., 2016; Fernando et al., 2017; Madotto et al., 2021; Zhang et al., 2022). Finally, *memory-based* methods keep a number of key samples from previous tasks in memory to alleviate forgetting (Rebuffi et al., 2017; Shin et al., 2017; Chaudhry et al., 2019; Qin and Joty, 2022). The memory data can be either real examples (Han et al., 2020) or generated by language models (Sun et al., 2020).

More recently, researchers have considered exploring knowledge transfer in LL, *i.e.,* learning on a task can benefit from learning on another task by transferring related knowledge. This includes CTR (Ke et al., 2021) and CUBER (Lin et al., 2022a). Despite their effectiveness, these methods mainly focus on classification tasks, while generation tasks

[1]Following Zhang et al. (2022), we use an Adapter (Houlsby et al., 2019) as the insertable module.

typically have more complex label space. Note that this line of research is different from transfer learning (Ruder et al., 2019), which mainly focuses on exploring better ways to reuse learned knowledge which is usually static, *e.g.,* a frozen language model. In contrast, the acquired knowledge is continually accumulated in lifelong learning.

Lifelong Sequence Generation (LSG) enables the model to learn sequentially from a stream of generation tasks. Sun et al. (2020) propose LAMOL which formalizes different types of tasks as question answering and utilizes pseudo-sample replay to alleviate forgetting. Chuang et al. (2020) further improve LAMOL by knowledge distillation (Hinton et al., 2015). AdapterCL (Madotto et al., 2021) inserts task-specific modules into every transformer layer to learn new tasks while keeping the pre-trained LM and previous modules frozen. On the basis of AdapterCL, Zhang et al. (2022) introduce ACM which dynamically adds modules for learning new tasks depending on whether there are reusable previously inserted modules. Though ACM can enable knowledge transfer to some extent via module sharing, there is no explicit mechanism to encourage knowledge transfer across tasks, a common phenomenon of human learning.

Summary. Existing work in LSG mainly focuses on mitigating the catastrophic forgetting of previously learned knowledge while paying little attention to knowledge transfer across tasks. In contrast to these lines of work, we aim to explicitly encourage forward knowledge transfer in LSG inspired by the way humans learn (Lake et al., 2017).

3 Problem Formulation

LSG involves learning from a stream of sequence generation tasks $\mathbb{T} = (\mathcal{T}^1, ..., \mathcal{T}^n)$, where every task $\mathcal{T}^i$ has its own training set D^i_{train}, validation set D^i_{valid}, and test set D^i_{test}. Every dataset D contains a set of examples $\{(X_j, Y_j)\}_{j=1}^{|D|}$, where X_j and Y_j denote the input and output texts, respectively. At time step k, the model is trained on the training set D^k_{train} of task $\mathcal{T}^k$ and has no access to real samples of previously learned tasks.

After the training on D^k_{train}, the model is expected to perform well on all the tasks learned so far, *i.e.,* $\mathcal{T}^1, ..., \mathcal{T}^k$, and will be evaluated on the test set D^i_{test} of each task $\mathcal{T}^i (1 \leq i \leq k)$ with corresponding evaluation metrics separately. Therefore, to achieve the goal of LSG, the model is required to alleviate the forgetting of acquired knowledge and better learn new patterns through possible forward knowledge transfer.

3.1 Data Format

Given an input-output text pair (X, Y) for a task, the model learns to decode the output text Y after reading the input X. Following Zhang et al. (2022), a natural language question Q describing the purpose of each task (task instruction) is inserted after the input to form a triple (X, Q, Y); see Appendix A.1 for an example. To learn a new task, the model is optimized to decode Y given X and Q. Denoting the concatenation of X, Q and Y as A, the autoregressive training objective is:

$$\mathcal{L}_{\text{task}} = -\sum_{j=m+1}^{n} \log p_\theta(A_j | A_{<j}) \qquad (1)$$

where n is the total number of tokens in A and $(A_1, ..., A_m)$ is the concatenation of X and Q, and θ denotes the model parameters.

4 Methodology

Inspired by how humans learn a new task (Fig. 1), DMEA divides the learning process into three stages. The *expansion* stage (§4.1) first determines the model architecture dynamically. The *selection* stage (§4.2) then selects the top-K most similar previous tasks which are utilized in the final *adaptation* stage (§4.3) to facilitate adaptation to the new task. We also employ pseudo-sample replay along with a dynamic gradient scaling method to balance the learning of the new and replayed tasks.

4.1 Expansion Stage

Humans are able to determine whether previously acquired skills are sufficient to solve a new task. Our method DMEA aims to mimic this learning process in the expansion stage. It can dynamically decide whether to reuse modules of previous tasks or insert a new module in every transformer layer to learn novel knowledge. Inspired by Zhang et al. (2022), we utilize differentiable architecture search (Liu et al., 2019) to achieve this goal.

Specifically, assuming that there are k modules (*i.e.,* Adapter (Houlsby et al., 2019)) $\{m^l_1, ..., m^l_k\}$ in layer l of the transformer model before learning a new task $\mathcal{T}^j$, we temporarily insert a new module m^l_{k+1} into this layer at the beginning of the expansion stage. For each forward pass, after calculating the output h^l_t of every module m^l_t in the layer separately, we fuse all outputs $\{h^l_1, ..., h^l_{k+1}\}$ through

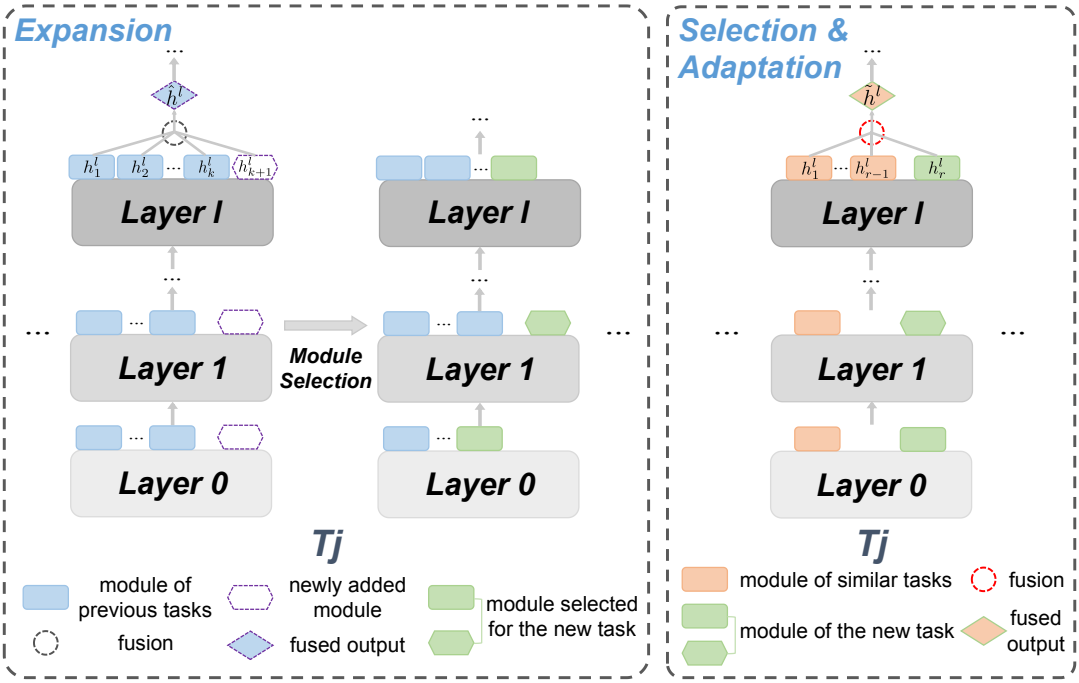

Figure 2: Overview of the proposed DMEA method. In the **expansion** stage (left), after inserting a new module (adapter) into each layer, DMEA dynamically determines the architecture by differentiable architecture search. The learnable coefficients are initialized based on the cosine similarity between word frequency distributions of learned tasks and the new task to leverage the correlation between tasks. After training, the module with the *largest* coefficient in every layer is selected for the new task. Newly added modules that are not selected will be discarded. In the **selection** stage (right), DMEA selects the top-K most similar previous tasks through input subspace to facilitate adaptation to a new task. Finally, during **adaptation**, the output of the selected similar tasks is fused with that of the new task in every layer to enable forward knowledge transfer. Note that only newly added modules and modules selected for the new task are learnable modules in the expansion and adaptation stages, respectively. In addition, DMEA introduces dynamic gradient scaling to balance the learning of the new task and replayed tasks.

learnable coefficients $\{\lambda_1^l, ..., \lambda_{k+1}^l\}$ as follows.

$$\hat{h}^l = \sum_{t=1}^{k+1} \frac{e^{\lambda_t^l}}{\sum_{s=1}^{k+1} e^{\lambda_s^l}} h_t^l \quad (2)$$

The weighted average $\hat{h}^l$ is then passed to the next part of the model for learning. After training the model on D^j_{train} for several epochs using $\mathcal{L}_{\text{train}}$ (defined in §4.3), we select the module with the **largest** coefficient in every layer for the new task $\mathcal{T}^j$.

Different from Zhang et al. (2022) which initialize $\{\lambda_1^l, ..., \lambda_{k+1}^l\}$ with predefined hyperparameters, we propose to dynamically initialize learnable coefficients based on the correlation between the learned tasks $\mathcal{T}^1, ..., \mathcal{T}^{j-1}$ and new task $\mathcal{T}^j$. Denoting the word frequency distribution of $\mathcal{T}^i$ as f^i and all previous tasks sharing the module m_t^l as $\mathcal{Z}_t^l$, the learnable coefficient λ_t^l is initialized as:

$$\lambda_t^l = \begin{cases} \max\limits_{\mathcal{T}^i \in \mathcal{Z}_t^l} \cos(f^i, f^{k+1}), & 1 \le t \le k \\ \min\limits_{1 \le i \le k} \lambda_i^l, & t = k+1 \end{cases} \quad (3)$$

where cos is the cosine similarity function and f^i is calculated based on the training set D^i_{train}. In this way, a previous module shared by tasks with higher word frequency distribution similarity to the new task has a larger initial coefficient, increasing the tendency to reuse it. In addition, the coefficient λ_{k+1}^l of the newly added module m_{k+1}^l is initialized to the minimum value of the initial coefficients $\{\lambda_1^l, ..., \lambda_k^l\}$ of previously added modules $\{m_1^l, ..., m_k^l\}$ to encourage module reuse.

The selected module in layer l can be either from previous modules $\{m_1^l, ..., m_k^l\}$ or the newly added one m_{k+1}^l and will be tuned in the adaptation stage to accommodate new knowledge. We then discard newly added modules that are not selected. Note that only newly added modules and coefficients are learnable in the expansion stage; the pre-trained LM and previous modules are kept frozen.

4.2 Selection Stage

As humans, we can better acquire new knowledge by recognizing and utilizing knowledge from previously learned tasks that are similar (Lake et al.,

2017). Based on the observation that the correlation between input subspaces (the norm of projected subspace onto the other subspace) for two tasks can serve as a similarity measure between the tasks (Lin et al., 2022b), we select the top-K most similar previous tasks by input subspace to facilitate adaptation to the new task $\mathcal{T}^j$. The model architecture induced from the expansion stage is used for selection and adaptation.

Similar to Saha et al. (2021), we adopt Singular Value Decomposition (SVD) to obtain the input subspace of each task. After training the model on D^j_{train} for several epochs in the expansion stage, we randomly select n samples $\{X_1, ..., X_n\}$ from D^j_{train} and obtain their representations $\{\boldsymbol{X}_1, ..., \boldsymbol{X}_n\} \in \mathbb{R}^m$ by forward-propagating them through the network before selecting modules for $\mathcal{T}^j$. Following Radford et al. (2018), we use the final-layer representation of the last non-padding token in the input as the representation of the sample.

After obtaining the representation matrix $\boldsymbol{R}^j = [\boldsymbol{X}_1, ..., \boldsymbol{X}_n] \in \mathbb{R}^{m \times n}$ for task $\mathcal{T}^j$, we apply SVD to $\boldsymbol{R}^j$, *i.e.,* $\boldsymbol{R}^j = \boldsymbol{U}^j \boldsymbol{\Sigma}^j (\boldsymbol{V}^j)'$, where $\boldsymbol{U}^j = [\boldsymbol{u}^j_1, ..., \boldsymbol{u}^j_m] \in \mathbb{R}^{m \times m}$ is composed of left-singular vectors $\boldsymbol{u}^j_i$, $\boldsymbol{\Sigma}^j \in \mathbb{R}^{m \times n}$ is a rectangular diagonal matrix with singular values on the diagonal, and $\boldsymbol{V}^j = [\boldsymbol{v}^j_1, ..., \boldsymbol{v}^j_n] \in \mathbb{R}^{n \times n}$ is composed of right-singular vectors $\boldsymbol{v}^j_i$. To obtain the input subspace $\boldsymbol{S}^j$ of $\mathcal{T}^j$, we select the first k left-singular vectors in $\boldsymbol{U}^j$ to form the bases $\boldsymbol{B}^j = [\boldsymbol{u}^j_1, ..., \boldsymbol{u}^j_k]$ for $\boldsymbol{S}^j$, where k is determined by the requirement: $||\boldsymbol{R}^j_k||^2_F \geq \epsilon^j ||\boldsymbol{R}^j||^2_F$ with $\boldsymbol{R}^j_k$ being the k-rank approximation of $\boldsymbol{R}^j$, F being the Frobenius norm, and ϵ^j being a predefined threshold.

Inspired by Lin et al. (2022b), for a new task $\mathcal{T}^j$, the norm of its subspace projection onto the subspace of a previously learned task $\mathcal{T}^i$ could characterize the similarity $\boldsymbol{Q}_{j,i}$ between these two tasks. More formally,

$$\boldsymbol{Q}_{j,i} = \frac{||\text{Proj}_{\boldsymbol{S}^i}(\boldsymbol{S}^j)||_2}{||\boldsymbol{B}^j||_2} \tag{4}$$

where $\text{Proj}_{\boldsymbol{S}^i}(\boldsymbol{S}^j) = \boldsymbol{B}^j \boldsymbol{B}^i (\boldsymbol{B}^i)'$ denotes the subspace projection. After getting the similarity scores $\boldsymbol{Q}_{j,i}, 1 \leq i < j$ of all previous tasks, we pick K tasks $\mathbb{T}^{\text{sim}} = (\mathcal{T}^1, ..., \mathcal{T}^K)$ with the top-K highest scores to facilitate adaptation to the new task $\mathcal{T}^j$.

4.3 Adaptation Stage

For adaptation to $\mathcal{T}^j$, assume that $\mathbb{T}^{\text{all}} = (\mathcal{T}^1, ..., \mathcal{T}^K, \mathcal{T}^j)$ contains a total of r modules $\{m^l_1, ..., m^l_r\}$ in layer l. During the training on D^j_{train} using $\mathcal{L}_{\text{train}}$ (see Eq. (7)), for each sample in D^j_{train}, we fuse the output h^l_s of each module $m^l_s \in \{m^l_1, ..., m^l_r\}$ by learnable coefficients $\{\alpha^l_1, ..., \alpha^l_r\}$ to enable *forward knowledge transfer*:

$$\tilde{h}^l = \sum_{s=1}^{r} \frac{e^{\alpha^l_s}}{\sum_{u=1}^{r} e^{\alpha^l_u}} h^l_s \tag{5}$$

The learnable coefficients $\{\alpha^l_1, ..., \alpha^l_r\}$ are equally initialized to 1.0. Similar to the expansion stage, the fused output $\tilde{h}^l$ is passed to the next part of the model for learning. After training, the learnable coefficients will be saved for inference. Note that we only tune modules selected in the expansion stage (can be modules of previous tasks or newly added modules) and learnable coefficients while keeping the pre-trained language model and other modules frozen.

As there is no saved real sample of previously learned tasks when the model adapts to a new task, we also incorporate pseudo-sample replay (Sun et al., 2020) to alleviate the forgetting of acquired knowledge. We achieve this by simultaneously training the model as a task solver ($\mathcal{L}_{\text{task}}$ in §3.1) and as a data generator. When training as a data generator, the model learns to generate the triple (X, Q, Y) given a task-specific generation token G as input. Then before learning a new task, the model can generate pseudo samples of previous tasks, which are combined with new data for training to mitigate forgetting. Denoting the concatenation of G, X, Q and Y as A', the data generation loss is expressed as:

$$\mathcal{L}_{\text{data}} = -\sum_{i=2}^{m} \log p_\theta(A'_i | A'_{<i}) \tag{6}$$

where m is the total number of tokens in A'. The overall loss that DMEA optimizes for adapting to a new task is:

$$\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{task}} + \mu \mathcal{L}_{\text{data}} \tag{7}$$

where μ is the weight of data generation loss.

After the expansion stage, if the new task reuses some modules of previously learned tasks, the model will generate some pseudo samples of these tasks and train the model using $\mathcal{L}_{\text{train}}$ on the combination of new data and pseudo data. As the model has not seen new data before, the gradient norm of the new task on reused modules is much larger

Methods		Finetune	EWC	LAMOL	Adapter +LAMOL	AdapterCL	ACM	DMEA
Tune Whole Model?		✓	✓	✓	✗	✗	✗	✗
Pseudo-sample Replay?		✗	✗	✓	✓	✗	✓	✓
Similar Tasks	# 1	43.0	56.7	65.6	64.4	63.3	64.7	**65.8**
	# 2	51.6	60.9	65.1	64.7	63.3	64.9	**65.5**
	# 3	45.4	57.6	**65.7**	64.2	63.3	64.7	**65.6**
	# 4	31.5	46.6	65.6	65.0	63.3	65.3	**66.1**
Average		42.9	55.5	65.5	64.6	63.3	64.9	**65.8**
Random Tasks	# 1	33.7	37.7	55.6	53.1	56.1	56.8	**57.6**
	# 2	33.0	38.2	62.4	61.7	64.0	64.9	**65.6**
	# 3	25.8	43.4	54.7	53.9	55.5	56.3	**57.4**
	# 4	34.1	48.8	64.9	62.4	65.6	66.2	**67.2**
Average		31.7	42.0	59.4	57.8	60.3	61.0	**62.0**

Table 1: The average performance score for each task sequence after learning all tasks. We run experiments twice with different random seeds for every sequence. **Bold** indicates the best score. In each scenario, DMEA is significantly better than ACM with p-value < 0.05 (paired t-test). Note that while LAMOL is not directly comparable to other adapter-based methods as its learnable parameters are orders of magnitude larger, DMEA still outperforms LAMOL in most cases.

than that of replayed tasks. The learning process can easily be biased towards the new task which may affect previously acquired knowledge.

Therefore, to balance the learning of the new task and replayed tasks, we introduce *dynamic gradient scaling*. Specifically, assuming that the new task $\mathcal{T}^j$ reuses s modules $\{m_1, ..., m_s\}$ of a previous task $\mathcal{T}^i$ in all layers, we randomly select q examples from D^j_{train} and pseudo samples of $\mathcal{T}^i$ separately and forwards them through the model to obtain the gradient of $\mathcal{T}^j$ and $\mathcal{T}^i$ using $\mathcal{L}_{\text{train}}$ with regard to reused modules $\{m_1, ..., m_s\}$, denoted as g^j and g^i, respectively. The dynamic scale factor η^i_t is then calculated as:

$$\eta^i_t = (\frac{||g^j||_2}{||g^i||_2} - 1)e^{-t} + 1 \quad (8)$$

where t is the number of completed training epochs. After dynamic gradient scaling, the total loss for jointly learning $\mathcal{T}^j$ and $\mathcal{T}^i$ is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}^j_{\text{train}} + \eta^i_t \mathcal{L}^i_{\text{train}} \quad (9)$$

Note that in the early stage of training, the value of t is small. η_t is greater than 1 to balance the gradient of the new task $\mathcal{T}^j$ and the replayed task $\mathcal{T}^i$. When the model has seen enough new data in the late stage of training (no need to balance), η_t is approximately equal to 1 as the value of t is large.

5 Experimental Setup

In this section, we first describe investigated tasks and then introduce methods compared in our work.

5.1 Tasks

Four representative sequence generation tasks are investigated in our work: natural language generation, summarization, task-oriented dialogue and SQL query generation. Following Zhang et al. (2022), we consider two different scenarios: (*i*) LSG on *similar* tasks where the model learns a sequence of tasks of the same type but different domains, and (*ii*) LSG on *random* tasks where the model learns knowledge from different types of tasks. For LSG on similar tasks, we use five different domains from two natural language generation datasets (RNNLG (Wen et al., 2015) and E2ENLG (Novikova et al., 2017)) to form the task sequences. We further incorporate summarization (CNNDM (See et al., 2017)), task-oriented dialogue (MultiWOZ (Budzianowski et al., 2018)) and SQL query generation (WikiSQL (Zhong et al., 2017)) to form the task sequences for LSG on random tasks. As the task order might influence the performance, we randomly select four different orders for each scenario[2] (Appendix A.2). For each order, we report the average of all learned tasks' performance scores following Zhang et al. (2022), which could reflect whether the model can alleviate catastrophic forgetting while better acquiring new knowledge through positive forward knowledge transfer; see Appendix A.3 for details of task-specific evaluation metrics.

5.2 Methods Compared

Following Zhang et al. (2022), we use GPT-2 (Radford et al., 2019) as the backbone model and

[2]Zhang et al. (2022) sample data from the original set for data balance. To ensure a fair comparison among all methods, we resample new data for experiments.

Adapter (Houlsby et al., 2019) as the insertable module, and compare with the following methods:

- **Finetune** tunes the whole GPT-2 model only on the training data of the new task during the LSG process.
- **EWC** (Kirkpatrick et al., 2017) constrains the update of parameters that are important to previously learned tasks to alleviate forgetting.
- **LAMOL** (Sun et al., 2020) tunes the whole GPT-2 model with pseudo-sample replay.
- **Adapter+LAMOL** only inserts adapter modules for the first task and tunes these modules with pseudo-sample replay while keeping the backbone model frozen.
- **AdapterCL** (Madotto et al., 2021) inserts task-specific adapter modules for every new task while keeping the backbone model and previous modules frozen.
- **ACM** (Zhang et al., 2022) dynamically adds adapter modules for new tasks depending on whether there are reusable previous modules to improve the performance and parameter efficiency of AdapterCL. It is the state-of-the-art on LSG.

6 Results and Analysis

6.1 Main Results

Table 1 shows the average performance score for each task sequence after learning all tasks (see Appendix A.5 for the performance of each task). From the results, we can see that DMEA outperforms previous baselines in all LSG settings, which demonstrates the superiority of our method. Note that while the learnable parameters of LAMOL are orders of magnitude larger, DMEA still achieves better performance than LAMOL in 7 out of 8 runs, showing its effectiveness in LSG.

Simply fine-tuning the model with new samples leads to poor performance due to catastrophic forgetting. Although EWC adopts Fisher information matrix to alleviate forgetting, its performance is still much worse than other memory-based baselines, indicating the importance of pseudo-sample replay. When learning from a sequence of similar tasks, Adapter+LAMOL performs better than AdapterCL as AdapterCL applies parameter isolation to different tasks which might prevent positive knowledge transfer across tasks. However, this is not the case when learning from random tasks: AdapterCL achieves much better results than Adapter+LAMOL as AdapterCL can avoid catastrophic forgetting by assigning different learnable parameters to each task. The performance of ACM is superior to Adapter+LAMOL and AdapterCL in both scenarios, showing the effectiveness of its adaptive compositional architecture. However, ACM has no explicit mechanism to encourage forward knowledge transfer in LSG, which is actually the human learning paradigm. Our proposed DMEA consistently outperforms ACM by dynamically leveraging previously acquired knowledge to facilitate adaptation to new tasks.

Method	Similar Tasks	Random Tasks
DMEA	**65.8**	**57.4**
w.o. transfer	64.9	56.6
w.o. scaling	65.5	56.9
w.o. initialization	65.4	57.1
w.o. replay	48.3	52.2

Table 2: The average performance score for different ablations: (i) without forward knowledge transfer, (ii) without dynamic gradient scaling, (iii) without dynamic initialization, and (iv) without pseudo-sample replay. All components improve the performance of our method.

6.2 Ablation Study

We conduct several ablations to analyze the contribution of different components of DMEA. In particular, we investigate four variants of DMEA (*a*) without selecting similar previous tasks for forward knowledge transfer (*w.o.* transfer), (*b*) removing dynamic gradient scaling (*w.o.* scaling), (*c*) without dynamically initializing learnable coefficients (*w.o.* initialization), and (*d*) removing pseudo-sample replay (*w.o.* replay). For each scenario, *i.e.,* similar tasks or random tasks, we randomly pick one sequence for experiments. Table 2 reports the average performance score after learning all tasks for different ablations.

From the results, we can observe that all components contribute to the average performance. Removing forward knowledge transfer leads to a significant performance drop in both scenarios, indicating that selecting top-K most similar previous tasks can indeed discover and transfer useful learned knowledge to facilitate adaptation to the new task. The adoption of dynamic gradient scaling yields a moderate performance boost as it can balance the learning of the new task and replayed tasks to mitigate catastrophic forgetting. Dynamic initialization of learnable coefficients also facilitates performance improvement, demonstrating the

Time Step		AdapterCL	ACM	DMEA
Similar	2	55.8(+0.0)	56.0(+0.1)	**56.3**(+**0.3**)
	3	58.6(+0.0)	59.1(+0.4)	**59.5**(+**0.6**)
	4	61.2(+0.0)	62.5(+0.6)	**63.2**(+**0.9**)
	5	63.3(+0.0)	64.7(+0.3)	**65.8**(+**1.0**)
Random	2	55.4(+0.0)	56.3(+0.9)	**57.1**(+**2.1**)
	3	58.4(+0.0)	58.9(+0.3)	**59.7**(+**1.3**)
	4	64.0(+0.0)	64.3(+0.7)	**65.4**(+**1.4**)
	5	64.0(+0.0)	64.9(+0.6)	**65.6**(+**1.1**)

Table 3: The average performance score and forward knowledge transfer (FKT) of different methods at every time step. FKT is reported in parentheses.

effectiveness of leveraging the similarity of word frequency distributions between tasks. Without pseudo-sample replay, the performance score drops more than 5% due to the severe forgetting of previously acquired knowledge.

6.3 Further Analysis

Quantify Forward Knowledge Transfer. Following Ke et al. (2020), we define metrics quantifying forward knowledge transfer (FKT) at every time step t as:

$$\text{FWT} = \frac{1}{t-1}\sum_{i=2}^{t} R_{i,i} - \bar{d}_i. \qquad (10)$$

where $R_{i,j}$ is the performance score on $\mathcal{T}^j$ after learning $\mathcal{T}^i$ and $\bar{d}_i$ refers to the performance of training $\mathcal{T}^i$ individually, which is actually the result of AdapterCL. For each scenario, we randomly select one sequence for analysis and report the average performance score along with FKT at each step in Table 3. From the results, we can see that DMEA consistently outperforms ACM in terms of the average performance score and FKT at all steps, demonstrating that DMEA can better facilitate positive knowledge transfer.

Input Subspace vs. Other Similarity Metrics. The ablation (*w.o.* transfer) in §6.2 demonstrates the importance of selecting similar learned tasks. To further investigate whether different similarity metrics influence the performance of DMEA, we conduct controlled experiments with two new metrics: (*a*) cosine similarity of word frequency distributions between different tasks (*frequency*), and (*b*) cosine similarity of the representations of selected samples from different tasks[3] (*representation*). For each scenario, we use the same sequence as §6.2. From the results in Table 4, we can observe that selecting similar previous tasks by input subspace consistently outperforms using other similarity metrics, demonstrating its superiority.

[3]For a pair of tasks, we compute the cosine similarity for every representation pair and use the average as the similarity.

Metrics	Similar Tasks	Random Tasks
Input Subspace	**65.8**	**57.4**
Frequency	65.3	57.0
Representation	65.2	57.0
w.o. transfer	64.9	56.6

Table 4: The average performance score using different similarity metrics.

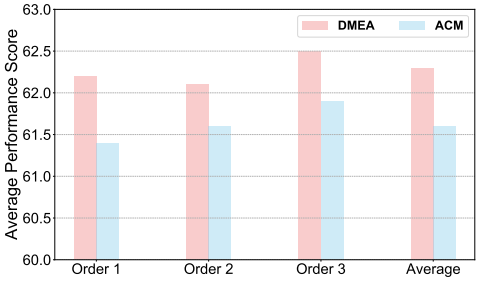

Figure 3: The average performance score for every order after learning all 8 tasks of the longer sequence.

Longer Sequence. As mentioned in §5.1, we mainly conduct experiments on sequences consisting of 5 tasks following Zhang et al. (2022). To verify whether DMEA can still outperform the baselines when learning from a larger number of tasks, we further combine all tasks investigated in this work to form a longer sequence of 8 tasks. We evaluate ACM and DMEA on this longer sequence with 3 different orders and report the average performance score for each order after learning all tasks in Fig. 3. We can see that DMEA is still superior to ACM when learning from longer sequences.

In addition, we report results on other types of tasks and evaluate the quality of pseudo data in Appendix A.6 and A.7, respectively.

7 Conclusion

In this work, we have introduced DMEA for lifelong sequence generation (LSG). DMEA uses task correlations to dynamically determine the suitable architecture required to acquire novel knowledge of a new task and select the most similar previous tasks through input subspace to facilitate knowledge transfer. It uses pseudo-sample replay along with dynamic gradient scaling to balance the learning of the new task and replayed tasks to further alleviate forgetting. With extensive experiments and analysis we have shown that DMEA consistently outperforms previous methods in different LSG settings. In the future, we would like to investigate ways to improve the quality of pseudo data and explore more metrics for task similarity.

Limitations

Although effective, DMEA has couple of limitations:

- DMEA mainly focuses on the setting where every task has plenty of training samples. In contrast, humans can easily learn to perform new tasks with only few data, which is a hallmark of human intelligence. We leave how to explore lifelong sequence generation in few-shot settings as future work.
- DMEA does not consider machine translation, a sequence generation task that might involve vocabulary changes. One potential solution is to use multilingual pre-trained language models.

References

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

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2019. Efficient lifelong learning with A-GEM. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Tianqi Chen, Ian J. Goodfellow, and Jonathon Shlens. 2016. Net2net: Accelerating learning via knowledge transfer. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Yung-Sung Chuang, Shang-Yu Su, and Yun-Nung Chen. 2020. Lifelong language knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2914–2924, Online. Association for Computational Linguistics.

Wafaa S El-Kassas, Cherif R Salama, Ahmed A Rafea, and Hoda K Mohamed. 2021. Automatic text summarization: A comprehensive survey. *Expert Systems with Applications*, 165:113679.

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*.

Donghoon Ham, Jeong-Gwan Lee, Youngsoo Jang, and Kee-Eung Kim. 2020. End-to-end neural pipeline for goal-oriented dialogue systems using GPT-2. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 583–592, Online. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 643–653, Lisbon, Portugal. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).

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. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Zixuan Ke, Bing Liu, and Xingchang Huang. 2020. Continual learning of a mixed sequence of similar and dissimilar tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 18493–18504. Curran Associates, Inc.

Zixuan Ke, Bing Liu, Nianzu Ma, Hu Xu, and Lei Shu. 2021. Achieving forgetting prevention and knowledge transfer in continual learning. *Advances in Neural Information Processing Systems*, 34:22443–22456.

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.

Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. 2017. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40:e253.

Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.

Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. 2022a. Beyond not-forgetting: Continual learning with backward knowledge transfer. In *Advances in Neural Information Processing Systems*.

Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. 2022b. TRGP: Trust region gradient projection for continual learning. In *International Conference on Learning Representations*.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30.

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

Saeed Masoudnia and Reza Ebrahimpour. 2014. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2):275–293.

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. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3461–3474, Online. Association for Computational Linguistics.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.

Chengwei Qin and Shafiq Joty. 2022. Continual few-shot relation learning via embedding space regularization and data augmentation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2776–2789, Dublin, Ireland. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. 2017. icarl: Incremental classifier and representation learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5533–5542. IEEE Computer Society.

Hippolyt Ritter, Aleksandar Botev, and David Barber. 2018. Online structured laplace approximations for overcoming catastrophic forgetting. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3742–3752.

Sebastian Ruder, Matthew E. Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18, Minneapolis, Minnesota. Association for Computational Linguistics.

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*.

Gobinda Saha, Isha Garg, and Kaushik Roy. 2021. Gradient projection memory for continual learning. In *International Conference on Learning Representations*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2990–2999.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2020. Lamol: Language modeling for lifelong language learning. In *International Conference on Learning Representations*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995. PMLR.

Yanzhe Zhang, Xuezhi Wang, and Diyi Yang. 2022. Continual sequence generation with adaptive compositional modules. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3653–3667, Dublin, Ireland. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

A Appendix

A.1 Task Instruction Example

Following Zhang et al. (2022), we insert a natural language question describing the purpose of every task (task instruction) after the input of each sample. Fig. 4 shows an example of the task instruction for E2ENLG (Novikova et al., 2017).

name[The Vaults], eatType[pub], priceRange[moderate], customer rating[1 out of 5], near[Caf Adriatic] what is the natural language form? A moderately priced pub, named The Vaults, is located near Caf Adriatic. It has a customer rating of 1 out of 5.

Figure 4: An example of the task instruction for E2ENLG. We color the task instruction in blue.

Order		Task Sequence
Similar Tasks	# 1	e2e → res → hotel → tv → laptop
	# 2	e2e → tv → res → laptop → hotel
	# 3	res → hotel → e2e → laptop → tv
	# 4	laptop → hotel → res → tv → e2e
Random Tasks	# 1	mwoz → cnn → e2e → res → hotel
	# 2	e2e → sql → hotel → mwoz → res
	# 3	cnn → hotel → sql → e2e → mwoz
	# 4	e2e → mwoz → laptop → sql → tv

Table 5: Different task orders for each scenario. 'e2e' stands for E2ENLG. 'res', 'hotel', 'laptop' and 'tv' are four domains in RNNLG (restaurant, hotel, laptop and television). 'sql', 'cnn' and 'mwoz' respectively stand for 'WikiSQL' (SQL query generation), 'CNNDM' (summarization) and 'MultiWOZ' (task-oriented dialogue).

A.2 Task Orders

We present different task orders for two LSG scenarios in Table 5.

A.3 Task-specific Evaluation Metrics

We report details of task-specific evaluation metrics in Table 6.

A.4 Implementation Details

All methods are implemented with PyTorch/Transformers library (Wolf et al., 2020). We adopt AdapterHub (Pfeiffer et al., 2020) to implement adapter modules. For hyperparameters, we mainly follow the settings in (Zhang et al., 2022) to have a fair comparison. In the expansion stage, we train the model for 6 epochs before selecting modules. In the adaptation stage, we set the number (n) of samples selected to obtain the input subspace as 100. The threshold ϵ is set as 0.95 for selecting left-singular vectors. We adopt 1 for the number of similar tasks K. For dynamic gradient scaling, we set 100 for the number (q) of examples selected to calculate the gradient.

A.5 Performance of Each Task

Table 7 shows the performance of each task for every task sequence after learning all tasks.

Dataset	Metric
RNNLG E2ENLG CNNDM	A-RG
WikiSQL	lfEM
MultiWOZ	dsEM

Table 6: Details of task-specific evaluation metrics. 'A-RG', 'lfEM' and 'dsEM' respectively stand for 'average of ROUGE-1, ROUGE-2 and ROUGE-L scores', 'exact match of logical forms' and 'exact match of dialogue state'.

High-quality Data

name[The Cricketers], eatType[restaurant], food[French], priceRange[moderate], near[Rainbow Vegetarian Cafe] what is the natural language form? The Cricketers is a French restaurant next to the Rainbow Vegetarian Cafe with moderate prices and a French taste.

Noisy Data

the table has columns week number, date, opponent, result, record and key words max, min, count, sum, avg, =, >, <, op, select, where, and, col, table, caption - - who the opponent was on the weekend where the record was 0 – 0? what is the translation from english to sql? select opponent from table where record = 0

Figure 5: Some examples of generated pseudo data. We color the task instruction in blue and output text in gray. Missing/wrong information is colored in red.

A.6 Other Types of Tasks

To explore whether the performance gain of DMEA is consistent on other types of tasks, we further include three new tasks: sentiment analysis (SST (Socher et al., 2013)), semantic role labeling (SRL (He et al., 2015)) and question answering (SQuAD (Rajpurkar et al., 2016)). We randomly select two tasks from the original task set three times and combine them with new tasks to form three task sequences. From the results shown in Table 8, we can observe that DMEA performs better than ACM on all sequences, showing its robustness to task types.

A.7 Quality of Pseudo Data

The ablation study in §6.2 demonstrates the importance of pseudo-sample replay. We further show several pseudo samples generated by DMEA in Fig. 5. We can see that DMEA can indeed generate high-quality pseudo samples to mitigate the forgetting of previously learned knowledge. However, the generated pseudo data could also be noisy as shown at the bottom of the figure, which might hinder further performance improvement.

Similar #1	e2e	res	hotel	tv	laptop	**Avg**
ACM	49.6	65.5	65.8	71.6	71.0	64.7
DMEA	49.2	67.1	68.1	72.5	72.0	65.8
Similar #2	e2e	tv	res	laptop	hotel	**Avg**
ACM	48.7	74.0	64.4	72.6	65.0	64.9
DMEA	47.8	74.7	64.9	74.0	65.9	65.5
Similar #3	res	hotel	e2e	laptop	tv	**Avg**
ACM	65.8	67.6	48.6	72.0	69.6	64.7
DMEA	66.9	66.6	49.5	73.9	71.1	65.6
Similar #4	laptop	hotel	res	tv	e2e	**Avg**
ACM	72.8	66.6	66.9	72.4	47.5	65.3
DMEA	74.5	67.6	67.3	72.9	48.0	66.1
Random #1	mwoz	cnn	e2e	res	hotel	**Avg**
ACM	81.7	26.1	47.6	64.6	64.2	56.8
DMEA	81.7	26.6	48.2	65.8	65.5	57.6
Random #2	e2e	sql	hotel	mwoz	res	**Avg**
ACM	48.4	62.7	64.6	84.8	64.0	64.9
DMEA	48.7	64.9	64.9	84.9	64.6	65.6
Random #3	cnn	hotel	sql	e2e	mwoz	**Avg**
ACM	26.3	63.2	62.1	47.8	82.4	56.3
DMEA	26.6	65.0	63.3	48.2	83.8	57.4
Random #4	e2e	mwoz	laptop	sql	tv	**Avg**
ACM	48.6	80.5	70.3	63.5	68.2	66.2
DMEA	48.9	82.6	71.4	64.1	68.9	67.2

Table 7: The performance of each task for every sequence after learning all tasks.

Method	Sequence			Average
	(i)	(ii)	(iii)	
ACM	68.4	63.6	71.8	67.9
DMEA	**69.5**	**64.4**	**73.0**	**69.0**

Table 8: The average performance score for every sequence after learning all new types of tasks.

A.8 Hyperparameter Search

We select the number of training epochs before modules selection from $\{6, 9, 12\}$, the number (n) of samples picked to obtain the input subspace from $\{50, 100, 200, 500\}$ and the threshold ϵ for selecting left-singular vectors from $\{0.90, 0.95, 0.99\}$. The number of similar previous tasks K is selected from $\{1, 2, 3\}$. The number (q) of examples for calculating the gradient in dynamic gradient scaling is selected from $\{20, 50, 100, 200\}$.