Domain Incremental Lifelong Learning in an Open World

Yi Dai[1*†], **Hao Lang**[2†], **Yinhe Zheng**[2‡], **Bowen Yu**[2], **Fei Huang**[2], **Yongbin Li**[2‡]
[1] Department of Computer Science and Technology, Tsinghua University [2] Alibaba Group
{hao.lang, yubowen.ybw, f.huang, shuide.lyb}@alibaba-inc.com,
dai-y21@mails.tsinghua.edu.cn, zhengyinhe1@163.com

Abstract

Lifelong learning (LL) is an important ability for NLP models to learn new tasks continuously. Architecture-based approaches are reported to be effective implementations for LL models. However, it is non-trivial to extend previous approaches to domain incremental LL scenarios since they either require access to task identities in the testing phase or cannot handle samples from unseen tasks. In this paper, we propose **Diana**: a dynamic architecture-based lifelong learning model that tries to learn a sequence of tasks with a prompt-enhanced language model. Four types of hierarchically organized prompts are used in Diana to capture knowledge from different granularities. Specifically, we dedicate task-level prompts to capture task-specific knowledge to retain high LL performances and maintain instance-level prompts to learn knowledge shared across input samples to improve the model's generalization performance. Moreover, we dedicate separate prompts to explicitly model unseen tasks and introduce a set of prompt key vectors to facilitate knowledge sharing between tasks. Extensive experiments demonstrate that Diana outperforms state-of-the-art LL models, especially in handling unseen tasks. We release the code and data at https://github.com/AlibabaResearch/DAMO-ConvAI/tree/main/diana.

1 Introduction

An essential ability of humans is to learn new tasks continuously in their lifetime since our surrounding world is ever involving (Thrun and Mitchell, 1995). Humans need to learn inputs from unseen new tasks everyday. However, neural network based NLP models tend to rapidly lose previously acquired knowledge when trained on new tasks. This phenomenon is referred to as catastrophic forgetting (French, 1999), and it's important to equip NLP models with the lifelong learning (LL) ability to alleviate this issue in advanced AI applications.

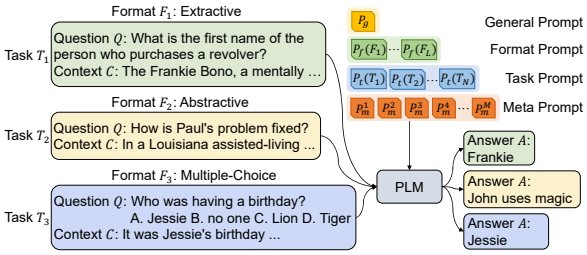

Figure 1: An overview of Diana. A pre-trained language model is used to learn tasks in different formats with hierarchically organized prompts.

An effective method to build LL models is the architecture-based approach (Chen et al., 2016; Rusu et al., 2016; Fernando et al., 2017; Wiwatcharakoses and Berrar, 2020), in which task-specific components are used to isolate knowledge for each separate task (Mancini et al., 2018). Recently, to leverage the power of pre-trained language model (PLM), some architecture-based LL models convert NLP tasks into a unified language modeling (LM) format (Sanh et al., 2021; Xie et al., 2022) and learn these tasks using a PLM. Separate prompts (Qin and Joty, 2022) or adapters (Madotto et al., 2021b) are allocated for different tasks to avoid the catastrophic forgetting issue.

However, despite the reported effectiveness, most above models are designed for the task incremental learning scenario, in which we assume task IDs for testing samples are available (Wang et al., 2022a,b). This setting limits the application of LL models because practical applications usually follow a more general domain incremental learning scenario (van de Ven et al., 2022), i.e., we cannot access the task IDs of most input samples.

There are generally two approaches to building LL models for domain incremental learning. One is to predict the task ID of each testing sample (Worts-

[*] Work done while the author was interning at Alibaba.
[†] Equal contribution.
[‡] Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2023, pages 5844–5865
July 9-14, 2023

man et al., 2020), and activate specified components based on the prediction (Figure 2a). This scheme achieves high LL performances if the predicted ID is correct (Madotto et al., 2021a). However, these models cannot handle samples from unseen tasks since there are no components designated for these samples and thus no task IDs to be predicted. This hinders the application of LL models because we often encounter samples from unseen tasks in practical situations (Dietterich, 2017).

Another approach to building domain incremental LL models is to organize model components at the instance-level, i.e., a pool of fine-grained components are dynamically combined in the forward pass for each input instance (Figure 2b). This approach avoids the trouble of explicitly determining task IDs. However, it usually yields low LL performance because there are no dedicated components for each task to capture task-specific knowledge (Wang et al., 2022a).

In this study, we combine the advantages of the above two approaches and propose **Diana**: a dynamic architecture-based lifelong learning model. We convert different NLP tasks into a unified LM format and propose to learn these tasks using a prompt-enhanced PLM (Figure 1). Specifically, Diana maintains four types of prompts to capture task knowledge from different granularities: 1. A *general prompt* P_g is used for all tasks; 2. The *format prompts* P_f are shared between tasks in a similar format; 3. A *task prompt* P_t is assigned for each incoming task; 4. A pool of *meta prompts* P_m are dynamically combined for each input instance. These four types of prompts present a hierarchical structure with a decreasing knowledge granularity, i.e., P_g captures global knowledge between all tasks, while P_m captures local knowledge that is shared between instances.

Diana can better generalize to unseen tasks while achieving high LL performances since its components are organized at both task and instance level. Moreover, we also maintain key vectors for P_t and P_m to better share task knowledge, and allocate separate task prompts to explicitly model samples for unseen tasks. Extensive experiments on benchmark NLP tasks indicate that Diana outperforms state-of-the-art (SOTA) baselines, especially in handling unseen tasks. Our main contributions are:

1. We propose Diana: a novel architecture-based domain incremental LL model that uses hierarchically organized prompts to capture knowledge in different granularities.

2. We are the first to consider unseen tasks in the testing phase of LL models. Specific prompts are designated in Diana to handle unseen tasks, and prompt keys are built to facilitate sharing of task knowledge.

3. Extensive experiments show that Diana outperformed SOTA baselines.

2 Related Work

Lifelong Learning aims at incrementally acquiring new knowledge without catastrophically forgetting previously learned ones. Generally, three categories of LL methods are proposed: **1.** Rehearsal-based methods (Rebuffi et al., 2017; Shin et al., 2017; Sun et al., 2019a; Chaudhry et al., 2019a; Buzzega et al., 2020) preserve past knowledge by replaying data from learned tasks; **2.** Regularization-based methods (Kirkpatrick et al., 2017; Zenke et al., 2017; Li and Hoiem, 2017; Ritter et al., 2018; Farajtabar et al., 2020) consolidate model parameters that are important to previous tasks by introducing additional regularization terms; **3.** Architecture-based methods (Chen et al., 2016; Rusu et al., 2016; Fernando et al., 2017; Maltoni and Lomonaco, 2019) add task-specific parameters to an existing base model for each task to prevent forgetting.

Experiment settings of LL methods can be generally classified into three scenarios based on whether the task ID is provided for testing samples and whether it must be inferred (van de Ven and Tolias, 2019), i.e., task-incremental learning (Mallya and Lazebnik, 2018; Ebrahimi et al., 2020), domain-incremental learning (Pu et al., 2021; Gao et al., 2022), and class-incremental learning (Zhang et al., 2020). In this work, we focus on the domain-incremental learning setting, where task ID is not provided for each testing sample. One line of methods in this category attempt to detect the task ID for each input sample (Madotto et al., 2021a). However, these methods fail to generalize to unseen tasks (Wang et al., 2022a). Another line of methods try to build a dynamic architecture for each input sample, for example, maintaining a pool of prompts that can be dynamically combined (Wang et al., 2022b). However, these methods yield sub-optimal performance since no task-specific parameters are used. Our model Diana is the first attempt to take advantage of the two aforementioned types of methods.

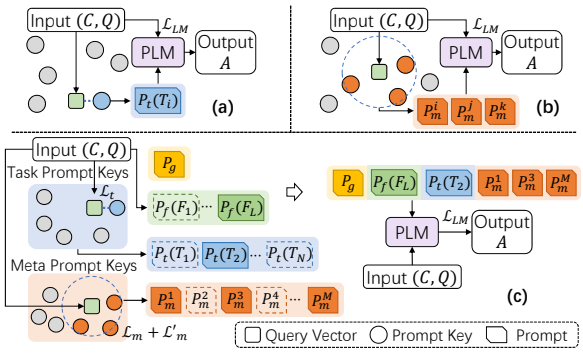

Figure 2: Different prompt organization schemes. (a) Each task is assigned a separate prompt and the closest prompt to the query vector is activated. (b) A pool of prompts are maintained and the top-M' closest prompts to the query vector are activated and combined. (c) Four kinds of prompts are hierarchically organized and combined based on the task format and distances between the query vector and prompt keys.

Pre-trained LM is becoming the de facto standard component for NLP models. To encourage knowledge sharing, existing approaches attempt to cast all NLP tasks into a unified text-to-text format (McCann et al., 2019) and learn these tasks by finetuning a PLM. A similar work compared to ours is ProQA (Zhong et al., 2022a), in which different QA tasks are unified and a set of structured prompts are used. However, ProQA only considers two QA tasks and is limited to the task incremental learning scenario, while our model is designed to tackle more general NLP tasks in a more general domain incremental learning scenario.

3 Method

3.1 Task Formulation

In this study, we aim to sequentially learn N tasks $T_1, \cdots, T_N$ that are presented in L different formats $F_1, \cdots, F_L$, $(L \leq N)$. Each task T_i is presented in a specific format F_j (such as "Classification" or "Summarization"), and each training sample of T_i is a tuple of a context C, a question Q, and an answer A: (C, Q, A). Note that the format of each task can be easily inferred from the context-question pair (C, Q). Our model g_θ is built to predict A based on C and Q. We also consider a more challenging open domain lifelong learning setting, i.e., the model needs to predict answers for unseen tasks. Therefore, we collect another N' unseen tasks $T_{N+1}, \cdots, T_{N+N'}$ that are only used for testing. We assume that all task identities of inputs are not available in the testing phase.

3.2 Framework of Hierarchical Prompts

We follow previous approaches to serialize the context C, question Q, and answer A into text sequences (Khashabi et al., 2020; Zhong et al., 2022a) and use a prompt-enhanced encoder-decoder model g_θ to learn each task T_i in Diana. We use soft prompts (Liu et al., 2021; Lester et al., 2021; Vu et al., 2022) in our study, i.e., each prompt is a sequence of trainable embeddings that are randomly initialized and learned in the training process. For each training sample (C, Q, A) from task T_i, we first construct a prompt $P(C, Q)$ based on (C, Q). Then the encoder takes in the concatenation of $P(C, Q)$, C, and Q and the decoder predicts A, i.e., $A = g_\theta([P(C, Q); C; Q])$, in which "$[;]$" denotes the sequence concatenation operation.

Four types of prompts are contained in $P(C, Q)$, i.e., $P(C, Q) = [P_g; P_f(F_j); P_t(T_i); P_m(C, Q)]$ (Figure 2c). Specifically, P_g is a general prompt, $P_f(F_j)$ is a format prompt (where F_j is the format of task T_i), $P_t(T_i)$ is a task prompt and $P_m(C, Q)$ is a combined meta prompt. These four types of prompts are organized hierarchically so that they are shared by samples in different granularities:

1. General Prompt P_g is shared for all training tasks so that it encodes global task knowledge.

2. Format Prompt $P_f(F_j)$ is shared between tasks in the same format F_j so that it captures format-related knowledge, i.e., knowledge that is shared between tasks in the format F_j.

3. Task Prompt $P_t(T_i)$ is specifically allocated for the task T_i and it is only shared for samples from T_i. We use $P_t(T_i)$ to learn task-specific knowledge. Moreover, to explicitly model samples from unseen tasks, we enlarge the set of task prompts with L extra prompts $\hat{P}_t(F_1), \cdots, \hat{P}_t(F_L)$, in which each prompt $\hat{P}_t(F_j)$ models the unseen task for a particular format F_j.

4. Meta Prompt $P_m(C, Q)$ is a dynamic combination of various instance-level prompts. Specifically, we maintain M instance-level meta prompts $\{P_m^i\}_{i=1}^M$ and dynamically combine these prompts based on the (C, Q) to obtain $P_m(C, Q)$. $P_m(C, Q)$ captures the knowledge shared between similar training instances.

We expect these four types of prompts can capture knowledge from different granularities since they are shared in different scopes. Moreover, to facilitate knowledge sharing, we allocate a key vector $\boldsymbol{k}_t(T_i)$ and $\boldsymbol{k}_m^j$ to each task prompt $P_t(T_i)$ and meta prompt P_m^j, respectively, and build a fixed text en-

coder h to map a context-question pair (C, Q) to a query vector $\boldsymbol{q} = h(C, Q)$. A two-stage learning process is introduced in Diana to learn these keys and $P(C, Q)$. Specifically, the first stage focuses on learning a representation space for prompt keys so that we can determine proper prompts to construct $P(C, Q)$. The second stage optimizes the constructed prompt $P(C, Q)$ and the backbone language model. These two stages are detailed in the following sections.

3.3 Key Vector Space Learning

We first optimize key vectors assigned to each task prompt and meta prompt to construct the prompt $P(C, Q)$ for each input (C, Q). Note that these key vectors are only used to determine the task prompt and meta prompt in $P(C, Q)$ because the general prompt P_g is shared by all tasks in Diana, and the format prompt $P_f(F_j)$ can be determined based on the format of C and Q directly.

Task Prompt Keys help to determine the task prompt in $P(C, Q)$. Specifically, for a given input (C, Q), we first calculate its query vector $\boldsymbol{q}$ and then determine the most similar task prompt key $\boldsymbol{k}_t(T_i)$ to $\boldsymbol{q}$. The task prompt $P_t(T_i)$ associated with $\boldsymbol{k}_t(T_i)$ is used to construct $P(C, Q)$.

Ideally, the key vector $\boldsymbol{k}_t(T_i)$ for a task prompt $P_t(T_i)$ should be located near samples from task T_i and distant to samples from other tasks T_j $(j \neq i)$. Therefore, when learning each task T_i, we maintain a small memory buffer $\mathcal{M}$ for samples from previously learned tasks T_j, $(j < i)$, and design the following exponential angular triplet loss (Ye et al., 2021) to enforce the above property:

$$\begin{aligned}\mathcal{L}_t =& \exp(||h(C, Q), \boldsymbol{k}_t(T_i)||+ \\ & \max(1 - ||h(C_n, Q_n), \boldsymbol{k}_t(T_i)||, 0)),\end{aligned} \tag{1}$$

in which the operator $||\cdot, \cdot||$ determines the distance between two input vectors (here we use cosine distance), (C_n, Q_n) is a negative sample extracted from the memory buffer $\mathcal{M}$:

$$(C_n, Q_n) = \underset{(C', Q') \in \mathcal{M}}{\operatorname{argmin}} ||h(C', Q'), \boldsymbol{k}_t(T_i)||. \tag{2}$$

Meta Prompt Keys help to combine these instance-level meta prompts $\{P_m^i\}_{i=1}^M$ to produce $P_m(C, Q)$. Specifically, for each input (C, Q), we select M' meta prompt keys that are closest to its query vector $\boldsymbol{q} = h(C, Q)$. Then $P_m(C, Q)$ is obtained by concatenating these M' meta prompts. Intuitively, the knowledge associated with (C, Q, A) is distributed in these M' meta prompts.

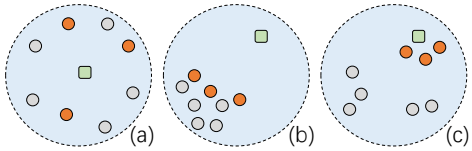

Figure 3: Illustration of the diversity and locality property. (a) The diversity property distributes key vectors to the whole space. (b) The locality property cluster similar keys to facilitate knowledge sharing. (c) Diana aims to achieve a balance between diversity and locality

When learning meta prompt keys, we expect the distribution of these keys to balance two properties: *diversity* and *locality* (Figure 3). Specifically, the diversity property aims to distribute these keys to the whole vector space so that every meta prompt can be involved in the training process. The locality property aims to cluster similar meta prompts keys so that the knowledge of each sample can be better shared. For each input C and Q, we propose the following loss to enforce the above two properties:

$$\begin{aligned}\mathcal{L}_m = & \sum_{i \in \mathcal{S}(C,Q)} \max(0, ||\boldsymbol{k}_m^i, h(C, Q)|| - \eta)+ \\ & \sum_{i,j \in \mathcal{S}(C,Q)} \max(0, \gamma - ||\boldsymbol{k}_m^i, \boldsymbol{k}_m^j||)/M'^2,\end{aligned} \tag{3}$$

where $\mathcal{S}(C, Q)$ is the index set of these M' meta prompt keys that are closest to $h(C, Q)$, η and γ are scalar hyper-parameters for the distance margin. Specifically, the first term in Eq. 3 enforces the locality property by pulling these M' meta prompt keys around the query vector. The second term enforces the diversity property by pushing these meta prompt keys away from each other to occupy the whole vector space.

Note that Eq. 3 only involves a single query $h(C, Q)$ from the current task. This may limit the learned meta prompt keys since samples from previously learned tasks are not considered. In this study, we extend Eq. 3 to better shape the distributions of meta prompt keys with the help of the memory buffer $\mathcal{M}$, in which samples from previously learned tasks are contained. Specifically, when learning the task T_i, we first calculate query vectors for samples in $\mathcal{M}$ and then group these query vectors into B clusters (we set $B = 5 \times i$ in our experiments, where i is the number of received tasks). Centroids of these B clusters are denoted as $\boldsymbol{c}_1, \cdots, \boldsymbol{c}_B$. For each sample (C, Q) from $\mathcal{M}$, the subsequent loss is optimized:

$$\mathcal{L}'_m = \sum_{i \in S(C,Q)} \max(0, ||\boldsymbol{k}_m^i, \boldsymbol{c}_k|| - \eta), \tag{4}$$

where $\boldsymbol{c}_k$ is the centroid to which (C, Q) belong. The above loss enforces the global diversity by scattering meta prompt keys to each centroid.

3.4 Model Training

Scheduled Sampling of Task Prompts When training Diana, the task ID of each sample (C, Q) is given so that we can directly get the task prompt $P_t(T_i)$. However, naively using golden truth task IDs leads to an exposure bias issue, i.e., task IDs inferred in testing may not always be correct.

In this study, we introduce a scheduled sampling process to tackle the exposure bias issue. Specifically, for a given sample (C, Q, A) in the k-th training step, we toss a coin and use the golden truth task ID with probability ϵ_k, or use the task ID inferred based on task prompt keys with probability $1 - \epsilon_k$ (Bengio et al., 2015). Note that when starting to learn each task, prompt keys are not well optimized, and thus the selected task ID is not accurate. Therefore, we set the value of ϵ_k to favor the golden truth task ID at the beginning (i.e., when k is small) and gradually switch to the inferred task ID as the training proceeds (i.e., when k is large), i.e., a linear decrement of ϵ_k is scheduled:

$$\epsilon_k = \max(0, \alpha - k\beta), \tag{5}$$

in which α and β are scalar hyper-parameters.

Note that LL models may encounter another source of exposure bias since we may receive inputs from unseen tasks in the testing phase. In this study, we use these L extra prompts $\hat{P}_t(F_1), \cdots, \hat{P}_t(F_L)$ to explicitly model unseen tasks. Specifically, for each training sample (C, Q, A), we first determine its task format F_j based on (C, Q), and allocate a small probability to use $\hat{P}_t(F_j)$ as its task prompt in $P(C, Q)$. In this way, we can capture general knowledge about all tasks for a given format in $\hat{P}_t(F_j)$ and expect the knowledge to facilitate handling unseen tasks.

Train with LM Loss For each training sample (C, Q, A), we first construct the prompt $P(C, Q)$ using approaches introduced above, and then optimize $P(C, Q)$ together with the encoder-decoder model g_θ using the following LM loss:

$$\mathcal{L}_{LM} = -log\ g_\theta(A|[P(C, Q); C; Q]). \tag{6}$$

The overall loss that we optimize for Diana is:

$$\mathcal{L} = \mathcal{L}_m + \mathcal{L'}_m + \mathcal{L}_t + \mathcal{L}_{LM}. \tag{7}$$

After learning each task T_i, we select a small number of samples from T_i based on the query vector of each sample to update the memory $\mathcal{M}$. This selection process aims to maintain diverse samples in $\mathcal{M}$. More details are in Appendix B.

See summarized training process in Algorithm 1.

3.5 Model Inference

When testing, we determine the prompt $P(C, Q)$ for each input context C and question Q, and use the learned model g_θ to predict the answer A.

Adaptive Decision Boundaries (ADB) are used to select proper task prompts in the testing phase. Specifically, for each task T_i, a scalar boundary δ_i is constructed following the approach proposed by Zhang et al. (2021). An input (C, Q) is regarded as a sample from unseen tasks if its query vector $h(C, Q)$ falls outside the boundary of every task:

$$||h(C, Q), \boldsymbol{k}_t(T_i)|| > \delta_i, \forall i \in [1, N]. \tag{8}$$

For samples from unseen tasks, we use the prompt $\hat{P}_t(F_j)$ as its task prompt in $P(C, Q)$, where F_j is the format of (C, Q).

Answer Prediction is performed with a greedy decoding process:

$$A = \underset{A'}{\text{argmax}}\, g_\theta(A'|[P(C, Q); C, Q]). \tag{9}$$

4 Experiments

4.1 Datasets

We use two sets of tasks to evaluate Diana:

1. decaNLP tasks: We follow Sun et al. (2019a) to select 5 tasks from the decaNLP (McCann et al., 2018) to train Diana. These tasks cover 3 different formats: Span Extraction, Sequence Generation, and Text Classification. We also collect $N' = 3$ additional tasks for each of these 3 format from decaNLP to serve as unseen tasks in the testing phase, i.e., our model is trained on $N = 5$ seen tasks while tested on 8 tasks;

2. QA tasks: The second set focuses on question answering (QA) benchmarks. Specifically, we use 8 QA datasets over 3 QA formats, i.e., Extractive QA, Abstractive QA and Multiple-Choice QA to train Diana. We also collect $N' = 3$ additional QA datasets for each of these three formats as unseen tasks, i.e., our model is trained on $N = 8$ seen tasks while tested on 11 tasks.

Note that task IDs for all testing samples are not available in our experiments. See Appendix C,J for more details of our dataset settings.

4.2 Evaluation Metrics

Individual tasks from above two task sets are evaluated following McCann et al. (2018) and Zhong et al. (2022a), respectively (see Appendix C). To evaluate the LL performance of Diana, we build a performance matrix $R \in \mathbb{R}^{N\times(N+N')}$, where $R_{i,j}$ is the model performance on task T_j after learning task T_i. The following LL metrics are computed:

1. Average Performance A_N and $A_{N'}$ is defined as the average performance of the final model on N seen tasks and N' unseen tasks, respectively:

$$A_N = \frac{1}{N}\sum_{j=1}^{N} R_{N,j}, \quad A_{N'} = \frac{1}{N'}\sum_{j=N+1}^{N+N'} R_{N,j}. \quad (10)$$

2. Average Forget F_N is defined as the average performance decrease of each task after it is learned:

$$F_N = \frac{1}{N-1}\sum_{j=1}^{N-1} \max_{i\in\{1,\cdots,N-1\}} (R_{i,j} - R_{N,j}). \quad (11)$$

In our experiments, we perform five runs with different random seeds and task orders. All reported metric scores are averages of these five runs. Ideally, we expect a strong LL model to yield high A_N and $A_{N'}$ scores, and low F_N scores.

4.3 Implementation Details

We use T5-base (Raffel et al., 2020) to initialize our encoder-decoder model, and set the lengths of soft prompts P_g, P_f, P_t, P_m to 20, 40, 40, 20, respectively. We maintain totally $M = 30$ meta prompts, and for each sample (C, Q) we choose $M' = 5$ meta prompts to construct $P_m(C, Q)$. We use the AdamW (Loshchilov and Hutter, 2017) optimizer with a learning rate of 1e-4 and batch size of 64. Each task is trained for five epochs. We set $\eta = 0.15$ and $\gamma = 0.3$ in Eq. 3 and $\alpha = 0.9$ and $\beta = 3e-4$ in Eq. 5. We maintain 50 samples from each learned task in the memory $\mathcal{M}$. All experiments are performed on 4 V100 GPUs, and the computational cost of our model is analyzed in Appendix G. See more details in Appendix A.

4.4 Baselines

We use the following competitive baselines covering all three types of LL models:

1. *Regularization-based methods*: **EWC** (Kirkpatrick et al., 2017) adopts the elastic weight consolidation approach to add regularization on parameter changes; **FLCB** (Gao et al., 2022) uses knowledge learned from previous tasks to guide future task learning; 2. *Rehearsal-based methods*: **ER** (Chaudhry et al., 2019b) replays memory samples from previous tasks to consolidate learned knowledge; **DER++** (Buzzega et al., 2020) augments ER with a L_2 loss on the soft labels; **AFPER** (Mi et al., 2020) combines ER with an adaptive elastic weight consolidation mechanism; 3. *Architecture-based methods*: **AdapterCL** (Madotto et al., 2021a) allocates separate adapters for different tasks; **L2P** (Wang et al., 2022b) attaches a group of prompts on a pre-trained model to share fine-grained knowledge; **DualPrompt** (Wang et al., 2022a) uses different prompts to encode task-invariant and task-specific knowledge; **ProQA** (Zhong et al., 2022a) uses a unified structural prompt to implement LL models. Note that ProQA is designed for task incremental learning that requires accessing task IDs in the testing phase.

We combine ProQA and ER to implement a stronger baseline **ProQA+ER**, in which samples from previous tasks are replayed for the ProQA model, and we also implement a variant of Diana by removing the memory buffer **Diana w/o** $\mathcal{M}$. We further report the performance for sequentially fine-tuning the LL model on all tasks (**Finetune**) and multi-task learning (**Multitask**). Note that the performance of Multitask is generally regarded as the upper bound of LL models when only seen tasks are considered.

All the above baselines are implemented following the same settings of our model, including using

Task ID in Test	Methods	Buffer Size	QA Tasks		decaNLP Tasks	
			A_N	F_N	A_N	F_N
Yes	ProQA	0	50.69	12.10	66.70	10.54
	ProQA+ER	50	54.00	7.27	71.26	5.33
No	Finetune	0	46.81	15.47	57.92	18.41
	EWC	0	47.81	14.55	63.17	13.58
	FLCB	0	47.50	14.98	63.86	13.36
	AdapterCL	0	48.08	13.29	64.25	12.38
	L2P	0	48.15	13.89	63.76	13.47
	DualPrompt	0	48.54	13.66	64.47	12.49
	ER	50	51.30	10.72	68.17	7.42
	DER++	50	52.01	10.05	69.10	6.86
	AFPER	50	52.69	9.28	69.78	6.17
	Diana w/o $\mathcal{M}$	0	50.30	12.68	66.14	10.61
	Diana	50	**55.93**	**6.75**	**72.70**	**4.25**
	Multitask	-	59.23	-	77.97	-

Table 1: Model performance on seen tasks. Best results (except the upper bound Multitask) are bolded. Our model Diana significantly outperforms other baselines on all metrics with p-value$<$0.05 (t-test).

the same backbone PLM, prompt size, and memory size used for replay. Note that for the ProQA baseline, we follow its original setting to provide task IDs for testing samples when evaluating.

4.5 Experiment Results

Results on Seen Tasks Table 1 shows the result on seen tasks from our two task sets. It can be seen that Diana outperforms all competitive baselines. Specifically, in the more general domain incremental learning scenario, i.e., when task IDs are unavailable in testing, Diana outperforms the best-performing baseline AFPER by a large margin. On QA tasks, Diana achieves 6.15% relative improvement on the A_N score and 27.26% relative decrease on the F_N score. Similar trend is also observed on decaNLP tasks. This means that Diana obtains higher performance with less forgetting in the LL process compared with other baselines.

We can also observe that: (1) Diana even outperforms the ProQA+ER baseline, which leaks task IDs in testing. This proves the superiority of our model design. (2) When task IDs are unavailable, Diana w/o $\mathcal{M}$ outperforms all baselines that do not use the memory buffer. This demonstrates that Diana's hierarchical prompts help to improve the LL performance even without the memory buffer.

Results on Unseen Tasks Table 2 shows the result on unseen tasks from our two task sets. Note that we cannot compute the average forget score for unseen tasks since these tasks are never learned. Diana yields the best performances on all settings. It also achieves a relative improvement of 9.49% and 11.04% on the $A_{N'}$ score compared with the best baseline DER++ on these two task sets.

We can also observe that: (1) When $\mathcal{M}$ is unavailable, models that share knowledge through fine-grained components (i.e., Diana and L2P) generally obtain high performance, and our model that allocates extra prompts for unseen tasks achieves the best performance. This validates our approach of using hierarchical prompts to explicitly model unseen tasks. (2) It is interesting to see that Diana even outperforms Multitask, which is usually regarded as the upper bound of traditional LL models when only seen tasks are considered. This indicates that traditional LL models have limited generalization ability to unseen tasks and it also proves that our model is effective in modeling unseen tasks.

See Appendix D for detailed experimental results of all tasks.

Task ID in Test	Methods	Buffer Size	$A_{N'}$	
			QA Tasks	decaNLP Tasks
Yes	ProQA	0	35.85	30.08
	ProQA+ER	50	38.00	30.92
No	Finetune	0	35.51	28.08
	EWC	0	36.07	29.76
	FLCB	0	36.68	31.17
	AdapterCL	0	36.84	30.32
	L2P	0	37.60	31.19
	DualPrompt	0	36.66	29.71
	ER	50	37.80	30.05
	DER++	50	38.47	31.24
	AFPER	50	36.79	30.22
	Diana w/o $\mathcal{M}$	0	39.22	33.19
	Diana	50	**42.12**	**34.69**
	Multitask	-	40.62	32.72

Table 2: Model performance on unseen tasks. Best results are bolded. Diana significantly outperforms other baselines on all metrics with p-value$<$0.05 (t-test).

4.6 Ablation Studies

We conduct ablation studies on different components of Diana. Specifically, three types of variants are implemented:

1. Each of these four prompt types is ablated: **w/o general prompt**, **w/o format prompt**, **w/o task prompt**, **w/o meta prompt**.

2. Schemes to enhance task prompts are ablated: **w/o Sched. Sampling** removes the scheduled sampling scheme and only uses the ground truth task IDs in training; **w/o G.T. Identity** is similar to the above variant. Instead, it only uses predicted task IDs in training; **w/o Neg. Samples** only uses positive samples to train task prompt keys, i.e., the second term in Eq. 1 is removed; **w/o ADB** uses fixed decision boundaries instead of ADBs to detect unseen tasks.

3. Schemes to enhance meta prompts are ablated: **w/o Sample Dive.** does not enforce the diversity property of the meta prompt keys, i.e., the second term in Eq. 3 is removed; **w/o Memory Dive.** does not use samples from previous tasks to enhance the diversity property, i.e., the loss $\mathcal{L}'_m$ (Eq. 4) is removed; **w/o Loc.** does not enforce the locality property of the meta prompt keys, i.e., the first term in Eq. 3 is removed; **w/o Cluster** does not cluster samples in $\mathcal{M}$, i.e., $\boldsymbol{c}_k$ in Eq. 4 is replaced with the query vector of each sample from $\mathcal{M}$.

Table 3 shows the performance of the above variants on QA tasks. It can be observed that Diana outperforms all the above variants. We can also see that: (1) "w/o Meta Prompt" lowers the LL performance by a large margin. This indicates that these

Categories	Variants	A_N	F_N	$A_{N'}$
Prompt Types	w/o General Prompt	55.47	6.93	40.74
	w/o Format Prompt	55.11	7.03	40.59
	w/o Task Prompt	53.87	8.50	39.66
	w/o Meta Prompt	53.46	8.56	40.04
Task prompt	w/o Sched. Sampling	55.15	7.43	42.00
	w/o G.T. Identity	54.16	7.61	41.27
	w/o Neg. Samples	54.97	7.66	41.78
	w/o ADB	55.48	6.98	41.01
Meta prompt	w/o Sample Dive.	55.24	6.91	41.23
	w/o Memory Dive.	55.02	7.41	41.48
	w/o Loc.	54.70	7.54	41.16
	w/o Cluster	55.46	6.99	41.51
Diana		**55.93**	**6.75**	**42.12**

Table 3: Ablation studies of model components and training strategies on QA tasks. Each result is an average of 5 random runs.

fine-grained meta prompts are important in building lifelong models. (2) The scheduled sampling scheme helps to learn better task prompts and thus improves the LL performance. (3) ADB improves model performance on unseen tasks (i.e., $A_{N'}$) by a large margin. (4) Enforcing the diversity property of meta prompt keys is important to obtain good key representations and facilitates the learning of each task.

4.7 More Analysis

4.7.1 Task ID Detection Performance

Diana needs to detect task IDs of input samples when determining the task prompt to be used. To verify the performance of the task ID detector implemented in Diana (Section 3.3 and 3.5), we compare the approach used in Diana with other task ID detectors: (1) Perplexity-based detector implemented in baseline "AdapterCL" determines the task IDs based on the perplexity of the PLM when different adapter modules are activated. (2) Distance-based detector implemented in our variant "w/o Neg. Samples" determines the task identity based on the distance between each key and query vectors. (3) Advanced distance-based detector implemented in our variant "w/o ADB" utilizes negative samples based on the above detector. Note that we do not apply ADB in the above two distance-based detectors. On our testing data, the above three approaches achieve a task ID detection accuracy of 59.84%, 52.72%, and 63.43%, respectively, while Diana reaches a task ID detection accuracy of 66.97%. This verifies the effectiveness of our approaches to optimize task prompt keys in detecting task IDs. More detailed comparisons of these task ID detectors can be found in Appendix E.

Criteria	Models	Z=2	Z=3	Z=5	Z=10
Locality	w/o Sample Dive.	0.73	0.72	**0.70**	0.48
	w/o Memory Dive.	0.74	0.72	0.69	0.63
	Diana	**0.74**	**0.73**	**0.70**	**0.66**
Diversity	w/o Sample Dive.	0.63	0.61	0.59	0.40
	w/o Memory Dive.	**1.00**	0.89	0.77	0.53
	Diana	**1.00**	**0.96**	**0.89**	**0.63**

Table 4: Quantitative analysis of the locality and diversity for meta prompt keys on QA tasks.

4.7.2 Distribution of Meta Prompt Keys

We also analyze the distribution of meta prompt keys $\mathcal{K} = \{\boldsymbol{k}_m^j\}_{j=1}^M$ constructed in Diana, which are expected to balance the locality and diversity property. Specifically, we introduce two metrics to quantify these two properties. For the diversity property, we follow Mansoury et al. (2020) to measure whether these meta prompt keys cover the whole vector space:

$$Diversity = |\bigcup_{j=1}^{M} \mathcal{N}_Z(\boldsymbol{k}_m^j, \mathcal{M})|/(Z \cdot M), \quad (12)$$

where $\mathcal{N}_Z(\boldsymbol{k}_m^j, \mathcal{M})$ represents the set of top-Z nearest samples in $\mathcal{M}$ around $\boldsymbol{k}_m^j$, and $|\cdot|$ returns the sample count of a set. High diversity scores are received if we can scatter meta prompt keys near every query vector from $\mathcal{M}$. For the locality property, we follow Scellato et al. (2010) to measure whether there are keys clustered around each query vector $\boldsymbol{q}$ in $\mathcal{M}$:

$$Locality = \sum_{\boldsymbol{q} \in \mathcal{M}} \sum_{\boldsymbol{k} \in \mathcal{N}_Z(\boldsymbol{q}, \mathcal{K})} (1 - ||\boldsymbol{q}, \boldsymbol{k}||)/(Z \cdot |\mathcal{M}|). \quad (13)$$

High locality scores are received if meta prompt keys in $\mathcal{K}$ are tightly clustered.

On the QA tasks, we compare the above two metrics between Diana and our ablation variants for meta prompts under different values of Z. As can be seen from Table 4, the strategies we introduced in Diana (Section 3.3) help to enforce the locality and diversity properties of meta prompt keys.

5 Conclusion

We propose Diana, a novel LL model for the domain incremental learning scenario. Diana converts different NLP tasks into a unified sequence generation format and uses a prompt-enhanced PLM to learn these tasks. We introduce four types of hierarchically organized prompts in Diana to capture knowledge in different granularities. These

prompts are shared between different scopes of samples and are dynamically combined based on a set of key vectors. The space of key vectors is learned with several distance-based regularization terms. Dedicated components are also allocated in Diana to model samples from unseen tasks. Experiments and empirical analysis on two sets of tasks show that Diana outperforms SOTA LL models, especially in handling samples from unseen tasks.

Limitations

One major limitation of this study is its input modality. Specifically, our model is limited to textual inputs and ignores other modalities (e.g., vision and audio). Open and domain incremental lifelong learning across modalities is more realistic and challenging. Fortunately, we can obtain robust features of different modalities via multi-modal pre-training models (Xu et al., 2021; Huo et al., 2021). For future work, we will try to tackle multi-modal tasks in an open (including out of distribution data (Lang et al., 2022, 2023a,b)) and domain incremental lifelong learning scenario with better approaches.

Ethics Statement

This work does not raise any direct ethical issues. In the proposed work, we seek to develop a model for domain incremental lifelong learning in an open world, and we believe this work leads to intellectual merits that benefit from a realistic and efficient lifelong learning model. All experiments are conducted on open datasets.

References

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and SIMONE CALDERARA. 2020. Dark experience for general continual learning: a strong, simple baseline. In *Advances in Neural Information Processing Systems*, volume 33, pages 15920–15930. Curran Associates, Inc.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2019a. Efficient lifelong learning with a-gem. In *Proceedings of ICLR*.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. 2019b. On tiny episodic memories in continual learning.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. 2016. Net2Net: Accelerating learning via knowledge transfer. In *Proceedings of ICLR*.

Pradeep Dasigi, Nelson F. Liu, Ana Marasović, Noah A. Smith, and Matt Gardner. 2019. Quoref: A reading comprehension dataset with questions requiring coreferential reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5925–5932, Hong Kong, China. Association for Computational Linguistics.

Thomas G Dietterich. 2017. Steps toward robust artificial intelligence. *Ai Magazine*, 38(3):3–24.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. 2020. Adversarial continual learning. In *European Conference on Computer Vision*, pages 386–402. Springer.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. 2020. Orthogonal gradient descent for continual learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3762–3773. PMLR.

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.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Jiaqi Gao, Jingqi Li, Hongming Shan, Yanyun Qu, James Z. Wang, and Junping Zhang. 2022. Forget less, count better: A domain-incremental self-distillation learning benchmark for lifelong crowd counting.

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.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Yuqi Huo, Manli Zhang, Guangzhen Liu, Haoyu Lu, Yizhao Gao, Guoxing Yang, Jingyuan Wen, Heng Zhang, Baogui Xu, Weihao Zheng, Zongzheng Xi, Yueqian Yang, Anwen Hu, Jinming Zhao, Ruichen Li, Yida Zhao, Liang Zhang, Yuqing Song, Xin Hong, Wanqing Cui, Dan Yang Hou, Yingyan Li, Junyi Li, Peiyu Liu, Zheng Gong, Chuhao Jin, Yuchong Sun, Shizhe Chen, Zhiwu Lu, Zhicheng Dou, Qin Jin, Yanyan Lan, Wayne Xin Zhao, Ruihua Song, and Ji-Rong Wen. 2021. Wenlan: Bridging vision and language by large-scale multi-modal pre-training. *CoRR*, abs/2103.06561.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

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 NAS*, pages 3521–3526.

Tomas Kocisky, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gabor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6(0):317–328.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7(0):452–466.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Hao Lang, Yinhe Zheng, Binyuan Hui, Fei Huang, and Yongbin Li. 2023a. Out-of-domain intent detection considering multi-turn dialogue contexts. *arXiv preprint arXiv:2305.03237*.

Hao Lang, Yinhe Zheng, Yixuan Li, Jian Sun, Fei Huang, and Yongbin Li. 2023b. A survey on out-of-distribution detection in nlp. *arXiv preprint arXiv:2305.03236*.

Hao Lang, Yinhe Zheng, Jian Sun, Fei Huang, Luo Si, and Yongbin Li. 2022. Estimating soft labels for out-of-domain intent detection. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 261–276, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics.

Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *TPAMI*, 40(12):2935–2947.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. Gpt understands, too.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization.

Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul Crook, Bing Liu, Zhou Yu, Eunjoon Cho, Pascale Fung, and Zhiguang Wang. 2021a. 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.

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

Arun Mallya and Svetlana Lazebnik. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773.

Davide Maltoni and Vincenzo Lomonaco. 2019. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116:56–73.

Massimiliano Mancini, Elisa Ricci, Barbara Caputo, and Samuel Rota Bulo. 2018. Adding new tasks to a single network with weight transformations using binary masks. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0.

Masoud Mansoury, Himan Abdollahpouri, Mykola Pechenizkiy, Bamshad Mobasher, and Robin Burke. 2020. Fairmatch: A graph-based approach for improving aggregate diversity in recommender systems. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, UMAP '20, page 154–162, New York, NY, USA. Association for Computing Machinery.

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

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2019. The natural language decathlon: Multitask learning as question answering.

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.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Nan Pu, Wei Chen, Yu Liu, Erwin M Bakker, and Michael S Lew. 2021. Lifelong person re-identification via adaptive knowledge accumulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7901–7910.

Chengwei Qin and Shafiq Joty. 2022. LFPT5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5. In *International Conference on Learning Representations*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

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 *Proceedings of CVPR*, pages 2001–2010.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, Washington, USA. Association for Computational Linguistics.

Hippolyt Ritter, Aleksandar Botev, and David Barber. 2018. Online structured laplace approximations for overcoming catastrophic forgetting. In *Proceedings of NIPS*, pages 3738–3748.

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

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China. Association for Computational Linguistics.

Salvatore Scellato, Cecilia Mascolo, Mirco Musolesi, and Vito Latora. 2010. Distance matters: geo-social metrics for online social networks. In *3rd Workshop on Online Social Networks (WOSN 2010)*.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. In *Proceedings of NIPS*, 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. 2019a. Lamol: Language modeling for lifelong language learning.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019b. Dream: A challenge data set and models for dialogue-based reading comprehension. *Transactions of the Association for Computational Linguistics*, 7:217–231.

Sebastian Thrun and Tom M Mitchell. 1995. Lifelong robot learning. *Robotics and autonomous systems*, 15(1-2):25–46.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada. Association for Computational Linguistics.

Gido M. van de Ven and Andreas S. Tolias. 2019. Three scenarios for continual learning.

Gido M van de Ven, Tinne Tuytelaars, and Andreas S Tolias. 2022. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou', and Daniel Cer. 2022. SPoT: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5039–5059, Dublin, Ireland. Association for Computational Linguistics.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022a. Dualprompt: Complementary prompting for rehearsal-free continual learning.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022b. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 139–149.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 438–449, Valencia, Spain. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Chayut Wiwatcharakoses and Daniel Berrar. 2020. Soinn+, a self-organizing incremental neural network for unsupervised learning from noisy data streams. *Expert Systems with Applications*, 143:113069.

Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. 2020. Supermasks in superposition. In *Advances in Neural Information Processing Systems*, volume 33, pages 15173–15184. Curran Associates, Inc.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang, et al. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *arXiv preprint arXiv:2201.05966*.

Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou. 2021. LayoutLMv2: Multi-modal pre-training for visually-rich document understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2579–2591, Online. Association for Computational Linguistics.

Hanrong Ye, Hong Liu, Fanyang Meng, and Xia Li. 2021. Bi-directional exponential angular triplet loss for rgb-infrared person re-identification. *IEEE Transactions on Image Processing*, 30:1583–1595.

Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *Proceedings of ICML*, pages 3987–3995.

Hanlei Zhang, Hua Xu, and Ting-En Lin. 2021. Deep open intent classification with adaptive decision boundary. In *AAAI*, pages 14374–14382.

Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. 2020. Class-incremental learning via deep model consolidation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1131–1140.

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

Wanjun Zhong, Yifan Gao, Ning Ding, Yujia Qin, Zhiyuan Liu, Ming Zhou, Jiahai Wang, Jian Yin, and Nan Duan. 2022a. Proqa: Structural prompt-based pre-training for unified question answering.

Wanjun Zhong, Yifan Gao, Ning Ding, Yujia Qin, Zhiyuan Liu, Ming Zhou, Jiahai Wang, Jian Yin, and Nan Duan. 2022b. ProQA: Structural prompt-based pre-training for unified question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4230–4243, Seattle, United States. Association for Computational Linguistics.

A More Implementation Details

We use T5-base (Raffel et al., 2020) to initialize our encoder-decoder model (12 layers, 768 dimensional hidden size, and 12 attention heads), and set the lengths of soft prompts P_g,P_f,P_t,P_m to 20, 40, 40, 20, respectively. We use a fixed T5-base encoder with an average pooling layer to obtain the query vector. We maintain a pool of $M = 30$ meta prompts, and for each sample (C, Q) we choose $M' = 5$ meta prompts to construct $P_m(C, Q)$. We use the AdamW (Loshchilov and Hutter, 2017) optimizer for training. All hyper-parameters are tuned according to the average score on validation datasets of NarQA, RACE, OBQA, SIQA and Dream. We tried epoch number of $\{2, 3, 4, 5, 6, 7, 8\}$ and learning rate of $\{1e{-}5, 5e{-}5, 1e-4, 5e-4, 1e-3\}$. We finally set the learning rate to 1e-4 and the number of training epochs to 5. We set $\eta = 0.15$ and $\gamma = 0.3$ in Eq. 3 and $\alpha = 0.9$ and $\beta = 3e-4$ in Eq. 5. For η and γ, we have a grid search between 0 and 0.5 with an interval of 0.05. For α and β, α is searched among $\{0.9, 0.7, 0.5\}$, while β is searched among $\{1e-5, 3e-5, 1e-4, 3e-4, 1e-3\}$. All experiments are performed on 4 V100 GPUs (32GB). The batch size is set to 64. In each set of tasks, We perform 5 runs with different task orders by setting the random seed to $\{42, 43, 44, 45, 46\}$ respectively. In this way, we report the average score of each method. Note that we only use the random seed 42 for tuning hyper-parameters.

In order to train extra task prompts $\{\hat{P}_t(F_1), \cdots, \hat{P}_t(F_L)\}$ for unseen tasks, we allocate a small probability $\omega = 5\%$ for each training sample (C, Q, A) to use $\hat{P}_t(F_j)$ as its task prompt in $P(C, Q)$, where F_j is the task format of (C, Q, A). To implement variant "w/o ADB" for ablation study, we use a fixed decision boundary instead of ADB. If for any task T_i, the distance $||h(C, Q), \boldsymbol{k}_t(T_i)|| > 0.35$, we regard the sample is from unseen tasks.

The adaptive decision boundary for each task is determined following the approach proposed by Zhang et al. (2021). We use AdamW optimizer with a learning rate of 0.02 to learn each decision boundary. To obtain the ROUGE-L score, we use the NLTK package for sentence tokenization, and python rouge-score package for evaluation.

B Memory Update

After learning task T_i, we select E diverse samples (we set $E = 50$ in our experiments) from T_i to update the memory $\mathcal{M}$ based on the query vector of each sample. Specifically, our selection criteria are built based on the distance of these prompt keys and query vectors. For each meta prompt key $\boldsymbol{k}_m^j$ $(j = 1, \cdots, M)$, we select top-$\lceil \frac{E}{M} \rceil$ samples ($\lceil \cdot \rceil$ is the ceiling function), whose query vectors are closest to $\boldsymbol{k}_m^j$. After accumulating $M\lceil \frac{E}{M} \rceil$ memory samples selected by M meta prompt keys, we rank these samples based on their distance to the corresponding meta prompt keys, and choose top-E samples with the smallest distance to be fed into $\mathcal{M}$. In this way, the memory $\mathcal{M}$ we constructed can expand to the whole space of prompt keys.

Note that, the memory buffer $\mathcal{M}$ is optional in Diana. Without $\mathcal{M}$, the loss in Eq. 4 is not optimized, and the second term in Eq. 1 is removed.

C Detailed Dataset Setting and Evaluation Metrics

For the decaNLP task set, 8 benchmarks over 3 formats are covered, i.e., (1) *Span Extraction*, including **SQuAD** (Rajpurkar et al., 2016), **QA-ZRE** (Levy et al., 2017), **QA-SRL** (He et al., 2015); (2) *Sequence Generation*, including **WOZ** (Wen et al., 2017), **WikiSQL** (Zhong et al., 2017), **CNN/DM** (Hermann et al., 2015); (3) *Text Classification*, including **SST** (Socher et al., 2013) and **MNLI** (Williams et al., 2018). For the QA task set, 11 QA benchmarks over 3 QA formats are covered, i.e., : (1) *Extractive* QA, including **SQuAD** (Rajpurkar et al., 2016), **NewsQA** (Trischler et al., 2017), and **Quoref** (Dasigi et al., 2019); (2) *Abstractive* QA, including **NarQA** (Kocisky et al., 2018), **NQOpen** (Kwiatkowski et al., 2019), and **Drop** (Dua et al., 2019); (3) *Multiple-Choice* QA, including **RACE** (Lai et al., 2017), **OBQA** (Mihaylov et al., 2018), **MCTest** (Richardson et al., 2013), **SIQA** (Sap et al., 2019), and **Dream** (Sun et al., 2019b). The statistics of the above datasets are summarized in Table 5. We follow the preprocess scheme released by Khashabi et al. (2020) to tackle these datasets. Some of these datasets do not contain a validation set, thus we only use the validation sets of NarQA, RACE, OBQA, SIQA and Dream in the QA task set to search hyper-parameters.

The evaluation for each single task follows Mc-

Task set	Dataset	Train set size	Val set size	Test set size
decaNLP	SQuAD	87k	-	10k
	QA-ZRE	-	-	12k
	QA-SRL	6.4k	-	2.2k
	WikiSQL	56k	-	15k
	WOZ	2.5k	-	1.6k
	CNN/DM	-	-	11k
	SST	6.9k	-	1.8k
	MNLI	-	-	20k
QA	SQuAD	87k	-	10k
	NewsQA	76k	-	4.3k
	Quoref	-	-	2.7k
	NarQA	65k	6.9k	21k
	NQOpen	9.6k	-	10k
	Drop	-	-	9.5k
	RACE	87k	4.8k	4.9k
	OBQA	4.9k	500	500
	MCTest	1.4k	-	320
	SIQA	33k	1.9k	2.2k
	Dream	-	2.0k	2.0k

Table 5: Dataset Statistics of the decaNLP task set and the QA task set.

Cann et al. (2018); Zhong et al. (2022b). Among the decaNLP tasks, we compute F1 score for QA-SRL and QA-ZRE, Exact Match (EM) score for SQuAD, MNLI and SST, ROUGE-L for CNN/DM. For WOZ, we adopt turn-based dialogue state exact match (dsEM). For WikiSQL, we use exact match of logical forms (lfEM). For the QA task set, we compute the accuracy of option selection for all Multi-Choice QA tasks and use EM score for all Extractive QA tasks. Among Abstractive QA tasks, we use F1 score for Drop and NQOpen, and ROUGE-L (Lin, 2004) for NarQA.

D Detailed Experimental Results

We provide the detailed performance of Diana under each single task compared with competitive baselines. The results under five seen tasks of the decaNLP task set, and eight seen tasks of the QA task set are shown in Table 6 and Table 7. The results of unseen tasks for the decaNLP task set and the QA task set are shown in Table 8 and Table 9.

E More Analysis of Task Identity Detection Performance

Architecture-based LL models need to detect task identities of input samples when these identities are unavailable in the testing phase. To verify the performance of the task identity detector implemented in Diana, we compare our approach with other task identity detectors: (1) Perplexity-based detector implemented in baseline "AdapterCL" determines the task identities based on the perplexity of the PLM when different adapter modules are activated. (2) Distance-based detector implemented in our variant "w/o Neg. Samples" determines the task identity based on the distance between each key and query vectors. (3) Advanced distance-based detector implemented in our variant "w/o ADB" utilizes negative samples based on the above detector. Note that we do not apply ADB in the above two distance-based detectors.

The above approaches are trained and evaluated with the QA tasks under two scenarios: (1) In **Closed-world**: detectors are only required to detect samples from seen tasks. Note that in this setting, the Advanced distance-based detector used in "w/o ADB" is the same as the task identity detector implemented in Diana. (2) In **Open-world**: detectors are required to handle unseen task samples as well. When tested in the open-world scenario, these two distance-based detectors adopt a fixed decision boundary of 0.35 (see Appendix A). The perplexity-based detector adopts a perplexity threshold of 4, i.e., samples with a perplexity score above 4 are regarded as unseen task samples. This perplexity threshold is selected based on the model performance on the validation set.

We report the task identity detection accuracy and Marco F1 scores for seen samples and unseen samples separately in Table 10. we can observe that: (1) The task identity detector used in Diana achieves the best performance in both scenarios. This proves the effectiveness of our task prompt keys in detecting task identities. (2) Negative samples used in Advanced distance-based detector significantly improve the task identity detection performance on seen tasks. (3) ADB is effective in improving the task identity detection performance on unseen tasks.

F More Analysis of Scheduled Sampling

We perform a more detailed analysis of the scheduled sampling scheme introduced in Diana. Specifically, in the ablation variant "w/o G.T. Identity", the model only uses predicted task identities in training. This scheme helps to alleviate the discrepancy between training and testing with the cost of the model coverage speed. In the ablation variant "w/o Sched. Sampling", the model only uses golden truth task identities in the training process. This scheme leads to the discrepancy between training and testing. The above two schemes under-perform our model Diana.

In this section, we analyze the task identity detection accuracy yield by the above schemes in

Task-ID in Test	Methods	Buffer Size	$R_{N,j}$ SQuAD	WikiSQL	SST	QA-SRL	WOZ	A_N	F_N
Available	ProQA	0	71.09	37.39	92.16	75.68	57.17	66.70	10.54
	ProQA+ER	50	75.57	50.98	91.67	76.74	61.33	71.26	5.33
Unavailable	Finetune	0	68.09	19.70	90.45	69.43	41.91	57.92	18.41
	EWC	0	70.57	35.97	89.79	71.19	48.34	63.17	13.58
	FLCB	0	70.96	33.35	90.03	74.71	50.23	63.86	13.36
	AdapterCL	0	71.82	35.14	90.95	72.83	50.53	64.25	12.38
	L2P	0	70.18	34.62	90.39	72.57	51.02	63.76	13.47
	DualPrompt	0	70.99	35.33	90.91	73.92	51.18	64.47	12.49
	ER	50	73.65	47.96	92.20	74.17	52.88	68.17	7.42
	DER++	50	74.18	49.27	92.34	75.11	54.61	69.10	6.86
	AFPER	50	75.27	48.90	91.56	76.34	56.82	69.78	6.17
	Diana w/o $\mathcal{M}$	0	71.94	36.25	91.03	74.59	56.90	66.14	10.61
	Diana	50	**76.93**	**51.09**	**92.74**	**77.69**	**65.06**	**72.70**	**4.25**
	Multitask	-	79.68	53.65	93.59	80.38	82.57	77.97	-

Table 6: Model performance on seen tasks in decaNLP. Best results (except the upper bound Multitask) are bold. Our model Diana significantly outperforms other baselines on all metrics with p-value$<$0.05 (t-test).

Task-ID in Test	Methods	Buffer Size	$R_{N,j}$ SQuAD	NewsQA	NarQA	NQOpen	RACE	OBQA	MCTest	SIQA	A_N	F_N
Available	ProQA	0	67.66	38.73	37.96	37.72	53.75	43.73	68.27	57.73	50.69	12.10
	ProQA+ER	50	71.20	40.17	41.94	39.00	57.09	47.00	77.94	57.67	54.00	7.27
Unavailable	Finetune	0	57.58	35.84	33.74	34.49	50.28	42.20	65.67	54.72	46.81	15.47
	EWC	0	59.84	36.44	34.88	35.14	50.54	43.43	66.52	55.68	47.81	14.55
	FLCB	0	58.73	36.97	34.27	34.90	51.63	41.53	66.60	55.39	47.50	14.98
	AdapterCL	0	59.64	37.31	37.42	36.70	49.57	41.80	66.67	55.54	48.08	13.29
	L2P	0	62.98	36.23	35.79	36.49	49.00	41.93	66.98	55.77	48.15	13.89
	DualPrompt	0	62.60	36.36	34.35	36.53	52.10	42.67	67.57	56.26	48.54	13.66
	ER	50	65.08	38.72	39.07	36.48	55.90	43.53	74.31	57.29	51.30	10.72
	DER++	50	67.08	39.03	39.91	36.93	56.42	44.13	74.77	57.77	52.01	10.05
	AFPER	50	68.14	40.79	40.16	38.89	55.08	46.60	75.33	56.52	52.69	9.28
	Diana w/o $\mathcal{M}$	0	65.51	37.78	37.35	37.41	54.14	46.27	68.50	57.41	50.30	12.68
	Diana	50	**74.44**	**42.91**	**43.16**	**40.05**	**59.08**	**48.47**	**78.44**	**60.92**	**55.93**	**6.75**
	Multitask	-	80.22	44.74	47.30	41.72	64.05	51.00	83.44	61.41	59.23	-

Table 7: Model performance on seen QA tasks. Best results (except the upper bound Multitask) are bold. Our model Diana significantly outperforms other baselines on all metrics with p-value$<$0.05 (t-test).

Methods	Buffer Size	$R_{N,j}$ CNN/DM	QA-ZRE	MNLI	$A_{N'}$
ProQA	0	13.25	37.58	39.42	30.08
ProQA+ER	50	14.18	38.42	40.17	30.92
Finetune	0	10.61	36.50	37.12	28.08
EWC	0	11.78	37.62	39.88	29.76
FLCB	0	12.98	40.02	40.52	31.17
AdapterCL	0	13.23	37.88	39.84	30.32
L2P	0	13.09	40.16	40.31	31.19
DualPrompt	0	12.92	37.04	39.18	29.71
ER	50	13.04	38.06	39.04	30.05
DER++	50	14.67	39.74	39.32	31.24
AFPER	50	12.14	38.66	39.85	30.22
Diana w/o $\mathcal{M}$	0	14.94	43.95	40.69	33.19
Diana	50	**15.80**	**44.74**	**43.53**	**34.69**
Multitask	-	15.98	42.12	40.07	32.72

Table 8: Model performance on unseen tasks in decaNLP. Best results (except Multitask) are bold. Diana significantly outperforms other baselines on all metrics with p-value$<$0.05 (t-test).

Methods	Buffer Size	$R_{N,j}$ Quoref	Drop	Dream	$A_{N'}$
ProQA	0	33.40	18.29	55.85	35.85
ProQA+ER	50	35.87	19.78	58.35	38.00
Finetune	0	33.08.	18.10	55.36	35.51
EWC	0	33.43	18.14	56.65	36.07
FLCB	0	34.85	18.31	56.88	36.68
AdapterCL	0	35.47	17.83	57.21	36.84
L2P	0	36.22	19.18	57.40	37.60
DualPrompt	0	35.22	18.52	56.25	36.66
ER	50	35.14	18.56	59.71	37.80
DER++	50	36.15	19.08	60.17	38.47
AFPER	50	35.26	18.83	56.29	36.79
Diana w/o $\mathcal{M}$	0	37.95	20.32	59.39	39.22
Diana	50	**40.42**	**22.91**	**63.03**	**42.12**
Multitask	-	36.27	22.99	62.60	40.62

Table 9: Model performance on unseen QA tasks. Best results (except Multitask) are bold. Diana significantly outperforms other baselines on all metrics with p-value$<$0.05 (t-test).

Figure 4 when learning the last task T_N in the input task sequence of QA task set. We can observe

Scenario	Methods	Scores on Seen Tasks		Scores on Unseen Tasks		Overall Scores	
		F1	Accuracy	F1	Accuracy	F1	Accuracy
Closed-world	Perplexity-based	44.92	52.20	-	-	44.92	52.20
	Distance-based	43.18	63.34	-	-	43.18	63.34
	Advanced distance-based	**54.37**	**75.35**	-	-	**54.37**	**75.15**
Open-world	Perplexity-based	33.15	58.64	26.14	**62.98**	32.37	59.84
	Distance-based	38.51	50.53	21.98	58.48	36.67	52.72
	Advanced distance-based	44.12	64.86	24.17	59.67	41.90	63.43
	Diana	**47.06**	**68.81**	**35.70**	62.16	**45.80**	**66.97**

Table 10: Task identity detection performance of different models under the QA tasks.

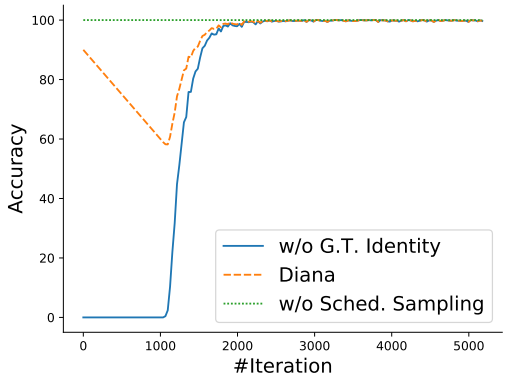

Figure 4: The task identity detection accuracy for samples from the last task T_N when learning T_N of the QA task set.

Methods	Tunable Parameters	Memory Size	Train Time Per Batch	Test Time All Tasks
Lower Bound	222.90M	0	0.55	523
EWC	222.90M	0	0.93	596
FLCB	222.90M	0	0.59	591
AdapterCL	262.25M	0	0.73	5852
L2P	223.39M	0	1.01	1013
DualPrompt	223.17M	0	0.93	1147
ER	222.90M	50	0.58	541
DER++	222.90M	50	0.68	604
AFPER	222.90M	50	0.95	630
ProQA	223.43M	0	0.86	863
Diana	223.84M	50	1.05	1108
Diana w/o $\mathcal{M}$	223.84M	0	0.97	1123

Table 11: Computational cost of Diana and baselines for the QA task set. "Train Time" is the average time cost for each batch. "Test Time" is the total time cost to evaluate all 11 tasks. Both train and test times are in seconds.

that the task identity detection accuracy achieved by "w/o G.T. Identity" is extremely low in earlier iterations, which hinders task prompts from sharing task-specific knowledge in the early training stage. The scheduled sampling process introduced in Diana effectively compromises between detecting correct task identities and alleviating the train-test discrepancy, and thus it results in the best LL performance among these variants. Note that the task identity detection accuracy in "w/o Sched. Sampling" is almost zero in the first 1,000 iterations when learning task T_N. This is because the task prompt keys for previous $N-1$ tasks are already well learned. The randomly initialized prompt key for task T_N needs to be pulled to the query vector space before starting to be functional.

G More Analysis of Computational Cost

We analyze the computational cost of Diana when learning the QA tasks, including the number of tunable parameters, time used for training and testing, and size of required memories retained from previous tasks. As indicated in Table 11, Diana does not introduce too much computation overhead.

PLM Size	Method	A_N	F_N	$A_{N'}$
T5-small	DER++	41.78	15.69	26.62
	Diana	**46.50**	**10.42**	**31.95**
T5-base	DER++	52.01	10.05	38.47
	Diana	**55.93**	**6.75**	**42.12**
T5-large	DER++	59.97	9.50	46.71
	Diana	**64.19**	**6.85**	**51.28**

Table 12: Performance with different sized PLMs on QA tasks.

Method	A_N	F_N	$A_{N'}$
Prompt tuning	46.76	**4.71**	32.87
Full tuning	**55.93**	6.75	**42.12**

Table 13: Performance with different training methods on QA tasks.

H Effect of PLM Size

We evaluate Diana and the best-performing baseline DER++ on different sized PLM using QA datasets. As shown in Table 12, Diana obtains better performance with larger PLM size, and consistently outperforms the baseline.

I Analysis of Training Method

During training, we follow a full tuning scheme that updates parameters of the backbone language models (T5) along with prompts. We also investigate the performance of prompt tuning, which fixes the backbone language model and only updates the prompts. As indicated in Table 13, prompt tuning dramatically degenerates the performance of Diana.

J Cases

We list some samples for tasks we modeled from the decaNLP task set and the QA task set respectively, shown in Table 14 and Table 15.

K Training Process

Details about the training process of Diana are shown in Algorithm 1.

Format	Dataset	Case
Span Extraction	SQuAD	**Context**: (Private_school) Private schooling in the United States has been... **Question**: In what year did Massachusetts first require children to be educated in schools? **Answer**: 1852
	QA-SRL	**Context**:the race is in mixed eights , and usually held in late february / early march. **Question**:when is something held ? **Answer**:in late february / early march
	QA-ZRE	**Context**:travis hamonic (born august 16 , 1990) is a canadian professional ice hockey... **Question**:what team does travis hamonic belong to ? **Answer**:new york islanders
Sequence Generation	CNN/DM	**Context**:(cnn) governments around the world are using the threat of terrorism... **Question**:what is the summary ? **Answer**:amnesty ' s annual death penalty report catalogs encouraging signs...
	WOZ	**Context**:what is the phone number and postcode of a cheap restaurant in the east part of town ?... **Question**:what is the change in state ? **Answer**:price range : cheap , area : east ; phone , postcode
	WikiSQL	**Context**:the table has columns player , no . , nationality , position , years in toronto... **Question**:what is the translation from english to sql ? **Answer**:select nationality from table where player = terrence ross
Text Classification	SST	**Context**:no movement , no yuks , not much of anything . **Question**:is this review negative or positive ? **Answer**: negative
	MNLI	**Context**:premise:yeah i i think my favorite restaurant is always been the one closest you... **Question**:hypothesis:i like him for the most part , but would still enjoy seeing someone beat him. - - entailment , neutral , or contradiction ? **Answer**: entailment

Table 14: Samples extracted from different decaNLP tasks. Each task contains a context, a question and an answer. Note that SQuAD is in the QA task set as well.

Format	Dataset	Case
Extractive	SQuAD	**Context**: (Private_school) Private schooling in the United States has been... **Question**: In what year did Massachusetts first require children to be educated in schools? **Answer**: 1852
	NewsQA	**Context**:ABECHE, Chad (CNN) – Most of the 103 children that a French charity... **Question**:WHO ARE UNDER ARREST IN CHAD? **Answer**:Three French journalists, a seven-member Spanish flight crew and one Belgian
	Quoref	**Context**:(Blast of Silence) Frankie Bono, a mentally disturbed hitman from Cleveland... **Question**:What is the first name of the person who follows their target to select...? **Answer**:Frankie
Abstractive	NarQA	**Context**:The play begins with three pages disputing over the black cloak usually worn by the actor... **Question**:WHO NORMALLY DELIVERS THE OPENING PROLOGUE IN THE PLAY? **Answer**:THE ACTOR WEARING THE BLACK CLOAK
	NQOpen	**Context**:- cartilage - cartilage cartilage is a resilient and smooth elastic tissue , a rubber... **Question**:where is each type of cartilage located in the body? **Answer**:many other body components
	Drop	**Context**:Hoping to rebound from their loss to the Patriots, the Raiders stayed at home for a Week... **Question**:How many field goals did both teams kick in the first half? **Answer**:2
Multiple-Choice	RACE	**Context**:It's cool, and it's hot, and everyone is doing it. People talk about it often, and friends... **Question**:A blogger is a person _ . (A) who teaches kids bad words (B) who posts songs from the latest bands (C) who got drunk last weekend (D) who writes diaries online **Answer**: who writes diaries online
	OBQA	**Context**:Null **Question**:Frilled sharks and angler fish live far beneath the surface of the ocean, which is why they are known as (A) Deep sea animals (B) fish (C) Long Sea Fish (D) Far Sea Animals Deep sea animals Answer:Deep sea animals
	MCTest	**Context**:It was Jessie Bear's birthday. She was having a party... **Question**:Who was having a birthday? (A) Jessie Bear (b) no one (C) Lion (D) Tiger **Answer**:Jessie Bear
	SIQA	**Context**:Tracy didn't go home that evening and resisted Riley's attacks **Question**:What does Tracy need to do before this? (A) make a new plan (B) Go home and see Riley (C) Find somewhere to go **Answer**:Find somewhere to go
	Dream	**Context**:M: How long have you been teaching in this middle school? W: For ten years... **Question**:What's the woman probably going to do? (A) To teach a different textbook. (B) To change her job. (C) To learn a different textbook. **Answer**:To change her job.

Table 15: Samples extracted from different QA tasks. Each task contains a context, a question and an answer.

Algorithm 1 Training process of Diana

Input: prompt-enhanced model g_θ, datasets $\{(C_j, Q_j, A_j)\}_{j=1}^{n_i}$ for each task T_i (i=1, $\cdots$, N), memory buffer $\mathcal{M}$, general prompt P_g, format prompts $\{P_f(F_j)\}_{j=1}^{F}$, task prompts $\{P_t(T_i)\}_{i=1}^{N} \cup \{\hat{P}_t(F_j)\}_{j=1}^{F}$, meta prompts $\{P_m^i\}_{i=1}^{M}$, task prompt keys $\{\boldsymbol{k}_t(T_i)\}_{i=1}^{N}$, meta prompt keys $\{\boldsymbol{k}_m^i\}_{i=1}^{M}$

Initialize: $\mathcal{M} \leftarrow \emptyset$
for Each task T_i, $i = 1, \cdots, N$ **do**
  if $\mathcal{M} \neq \emptyset$ **then**
    Calculate cluster centroids $\boldsymbol{c}_1, \cdots, \boldsymbol{c}_B$ of $\mathcal{M}$
  end if
  for number of training epochs **do**
    for Each mini-batch $I \in \{(C_j, Q_j, A_j)\}_{j=1}^{n_i} \cup \mathcal{M}$ **do**
      Obtain ϵ_k by Eq. 5
      for $(C, Q, A) \in I$ **do**
        Obtain format F_j of (C, Q, A)
        Sample ϵ, ζ from $U(0, 1)$
        if $\zeta < \omega$ **then**
          $P_t(C, Q) \leftarrow \hat{P}_t(F_j)$ {Use task prompt $\hat{P}_t(F_j)$ for unseen tasks}
        else if $\epsilon < \epsilon_k$ **then**
          $P_t(C, Q) \leftarrow P_t(T_i)$ {Use the golden truth task identity to select task prompt}
        else
          $P_t(C, Q) \leftarrow P_t(\underset{T_\tau \in \{T_1, \cdots, T_i\}}{\operatorname{argmin}} (||\boldsymbol{q}, \boldsymbol{k}_t(T_\tau)||))$ {Use the inferred task identity to select task prompt}
        end if
        $\mathcal{S}(C, Q) \leftarrow$ indexes of M' meta prompt keys that are closest to $\boldsymbol{q}$
        $P_m(C, Q) \leftarrow \{P_m^j\}_{j \in \mathcal{S}(C,Q)}$
        $P(C, Q) \leftarrow [P_g; P_f(F_j); P_t(C, Q); P_m(C, Q)]$
        Calculate per sample loss $\mathcal{L}_{LM}$ on g_θ and $P(C, Q)$ by Eq. 6
        Obtain negative sample (C_n, Q_n) from $\mathcal{M}$ by Eq. 2
        Calculate per sample loss $\mathcal{L}_t$ on $\boldsymbol{k}_t(T_i)$ by Eq. 1
        Calculate per sample loss $\mathcal{L}_m$ on $\{\boldsymbol{k}_m^{s_j}\}(s_j \in \mathcal{S}(C, Q))$ by Eq. 3
        if $(C, Q, A) \in \mathcal{M}$ **then**
          Calculate per sample loss $\mathcal{L'}_m$ on $\{\boldsymbol{k}_m^{s_j}\}(s_j \in \mathcal{S}(C, Q))$ by Eq. 4
        end if
      end for
      Update g_θ and prompts with accumulated $\mathcal{L}_{LM}$
      Update task prompt keys $\{\boldsymbol{k}_t(T_i)\}_{i=1}^{N}$ with accumulated $\mathcal{L}_t$
      Update meta prompt keys $\{\boldsymbol{k}_m^i\}_{i=1}^{M}$ with accumulated $\mathcal{L}_m$ and $\mathcal{L'}_m$
    end for
  end for
  Update $\mathcal{M}$ with $\{(C_j, Q_j, A_j)\}_{j=1}^{n_i}$ according to details in Appendix B
end for

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section Limitations

☑ A2. Did you discuss any potential risks of your work?
Section Ethics Statement

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 4

☑ B1. Did you cite the creators of artifacts you used?
Section 4, Appendix C

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Appendix C

C ☑ Did you run computational experiments?

Section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section4, Appendix A, Appendix G

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4, Appendix A

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4, Appendix A

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix A

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.