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, zhengyinhe10163.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.

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-
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 $T_1, \ldots, T_N$ that are presented in $L$ different formats $F_1, \ldots, F_L$. $L \leq N$. 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 $g_0$ 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 $T_{N+1}, \ldots, T_{N+N'}$ 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 $g_\theta$ 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., $P(C, Q) = [P_g; P_f(F_j); P_i(T_i); P_m(C, Q)]$ (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 $T_i$), $P_i(T_i)$ 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** $P_i(T_i)$ is specifically allocated for the task $T_i$ and it is only shared for samples from $T_i$. We use $P_i(T_i)$ to learn task-specific knowledge. Moreover, to explicitly model samples from unseen tasks, we enlarge the set of task prompts with $L$ extra prompts $P_l(F_1), \ldots, P_l(F_L)$, in which each prompt $P_l(F_j)$ 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 $k_{l(T_i)}$ and $k_{m_i}$ to each task prompt $P_l(T_i)$ and meta prompt $P_{m_i}$, respectively, and build a fixed text en-
coder \( h \) to map a context-question pair \((C, Q)\) to a query vector \( q = h(C, Q) \). 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 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 \( P_0 \) is shared by all tasks in Diana, and the format prompt \( P_1(F_j) \) 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 \( q \) and then determine the most similar task prompt key \( k_t(T_i) \) to \( q \). The task prompt \( P_t(T_i) \) associated with \( k_t(T_i) \) is used to construct \( P(C, Q) \).

Ideally, the key vector \( k_t(T_i) \) 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:

\[
\mathcal{L}_t = \exp(||h(C, Q), k_t(T_i)||^2) + \max(1 - ||h(C, Q), k_t(T_i)||, 0),
\]  

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 \( \mathcal{M} \):

\[
(C_n, Q_n) = \arg\min_{(C', Q') \in \mathcal{M}} ||h(C', Q'), k_t(T_i)||.
\]

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

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:

\[
\mathcal{L}_m = \sum_{i \in \mathcal{S}(C, Q)} \max(0, ||k^m_i, h(C, Q)|| - \eta) + \sum_{i, j \in \mathcal{S}(C, Q)} \max(0, \gamma - ||k^m_i, k^m_j||)/M'\gamma^2,
\]

where \( \mathcal{S}(C, Q) \) is the index set of these \( M' \) meta prompt keys that are closest to \( h(C, Q) \), \( \eta \) and \( \gamma \) 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 \( c_1, \ldots, c_B \). For each sample \((C, Q)\) from \( \mathcal{M} \), the subsequent loss is optimized:

\[
\mathcal{L}'_m = \sum_{i \in \mathcal{S}(C, Q)} \max(0, ||k^m_i, c_k|| - \eta),
\]
where \( c_k \) 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 \(P_i(T_i)\). 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 \(\epsilon_k\), or use the task ID inferred based on task prompt keys with probability \(1 - \epsilon_k\) \cite{Bengio2015}. 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 \(\epsilon_k\) 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 \(\epsilon_k\) is scheduled:

\[
\epsilon_k = \max(0, \alpha - k\beta),
\]

in which \(\alpha\) and \(\beta\) 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 \(\hat{P}_i(F_1), \cdots, \hat{P}_i(F_L)\) 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 \(\hat{P}_i(F_j)\) as its task prompt in \(P_i(C, Q)\). In this way, we can capture general knowledge about all tasks for a given format in \(\hat{P}_i(F_j)\) 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 \(P_i(C, Q)\) using approaches introduced above, and then optimize \(P_i(C, Q)\) together with the encoder-decoder model \(g_\theta\) using the following LM loss:

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

The overall loss that we optimize for Diana is:

\[
\mathcal{L} = \mathcal{L}_m + \mathcal{L}'_m + \mathcal{L}_t + \mathcal{L}_{LM}.
\]

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 \(P_i(C, Q)\) for each input context \(C\) and question \(Q\), and use the learned model \(g_\theta\) 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 \(\delta_i\) is constructed following the approach proposed by Zhang et al. \cite{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:

\[
||h(C, Q), k_i(T_i)|| > \delta_i, \forall i \in [1, N].
\]

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

**Answer Prediction** is performed with a greedy decoding process:

\[
A = \arg\max_{A'} g_\theta(A'||P_i(C, Q); C, Q)].
\]

### 4 Experiments

#### 4.1 Datasets

We use two sets of tasks to evaluate Diana:

1. **decaNLP tasks**: We follow Sun et al. \cite{2019a} to select 5 tasks from the decaNLP \cite{McCann2018} 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 CJ 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:

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

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; APER (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 structured 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

<table>
<thead>
<tr>
<th>Task ID in Test</th>
<th>Methods</th>
<th>Buffer Size</th>
<th>QA Tasks</th>
<th>decaNLP Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>ProQA</td>
<td>0</td>
<td>50.69</td>
<td>12.10</td>
</tr>
<tr>
<td>ProQA+ER</td>
<td>50</td>
<td>54.00</td>
<td>7.27</td>
<td>71.26</td>
</tr>
<tr>
<td>No</td>
<td>Finetune</td>
<td>0</td>
<td>46.81</td>
<td>15.47</td>
</tr>
<tr>
<td>EWC</td>
<td>0</td>
<td>47.81</td>
<td>14.55</td>
<td>63.17</td>
</tr>
<tr>
<td>FLCB</td>
<td>0</td>
<td>47.50</td>
<td>14.98</td>
<td>63.86</td>
</tr>
<tr>
<td>AdapterCL</td>
<td>0</td>
<td>48.08</td>
<td>13.29</td>
<td>64.25</td>
</tr>
<tr>
<td>L2P</td>
<td>0</td>
<td>48.15</td>
<td>13.89</td>
<td>63.76</td>
</tr>
<tr>
<td>DualPrompt</td>
<td>0</td>
<td>48.54</td>
<td>13.66</td>
<td>64.25</td>
</tr>
<tr>
<td>ER</td>
<td>50</td>
<td>51.30</td>
<td>10.72</td>
<td>68.17</td>
</tr>
<tr>
<td>DER++</td>
<td>50</td>
<td>52.01</td>
<td>10.05</td>
<td>69.10</td>
</tr>
<tr>
<td>APER</td>
<td>50</td>
<td>52.69</td>
<td>9.28</td>
<td>69.78</td>
</tr>
<tr>
<td>Diana w/o $\mathcal{M}$</td>
<td>0</td>
<td>50.30</td>
<td>12.68</td>
<td>66.14</td>
</tr>
<tr>
<td>Diana</td>
<td>50</td>
<td>55.93</td>
<td>6.75</td>
<td>72.70</td>
</tr>
<tr>
<td>Multitask</td>
<td>-</td>
<td>59.23</td>
<td>-</td>
<td>77.97</td>
</tr>
</tbody>
</table>
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 $M$ 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 $M$ 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.

<table>
<thead>
<tr>
<th>Task ID in Test</th>
<th>Methods</th>
<th>Buffer Size</th>
<th>$A_N$ Scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>ProQA</td>
<td>0</td>
<td>35.85</td>
</tr>
<tr>
<td>ProQA+ER</td>
<td>50</td>
<td>38.00</td>
<td>30.92</td>
</tr>
<tr>
<td>No</td>
<td>Finetune</td>
<td>0</td>
<td>35.51</td>
</tr>
<tr>
<td>EWC</td>
<td>0</td>
<td>36.07</td>
<td>29.76</td>
</tr>
<tr>
<td>FLCB</td>
<td>0</td>
<td>36.68</td>
<td>31.17</td>
</tr>
<tr>
<td>AdapterCL</td>
<td>0</td>
<td>36.84</td>
<td>30.32</td>
</tr>
<tr>
<td>L2P</td>
<td>0</td>
<td>37.60</td>
<td>31.19</td>
</tr>
<tr>
<td>DualPrompt</td>
<td>0</td>
<td>36.66</td>
<td>29.71</td>
</tr>
<tr>
<td>ER</td>
<td>50</td>
<td>38.00</td>
<td>30.05</td>
</tr>
<tr>
<td>DER++</td>
<td>50</td>
<td>38.47</td>
<td>31.24</td>
</tr>
<tr>
<td>AFPER</td>
<td>50</td>
<td>36.79</td>
<td>30.22</td>
</tr>
<tr>
<td>Diana w/o $M$</td>
<td>0</td>
<td>39.22</td>
<td>33.19</td>
</tr>
<tr>
<td>Diana</td>
<td>50</td>
<td>42.12</td>
<td>34.69</td>
</tr>
<tr>
<td>Multitask</td>
<td>-</td>
<td>40.62</td>
<td>32.72</td>
</tr>
</tbody>
</table>

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 $L_m$ (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 $M$, i.e., $c_k$ in Eq. 4 is replaced with the query vector of each sample from $M$.

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
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., $A_{NY}$) 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

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

<table>
<thead>
<tr>
<th>Categories</th>
<th>Variants</th>
<th>$A_N$</th>
<th>$F_N$</th>
<th>$A_{NY}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prompt Types</td>
<td>w/o General Prompt</td>
<td>55.47</td>
<td>6.93</td>
<td>40.74</td>
</tr>
<tr>
<td></td>
<td>w/o Format Prompt</td>
<td>55.11</td>
<td>7.03</td>
<td>40.59</td>
</tr>
<tr>
<td></td>
<td>w/o Task Prompt</td>
<td>53.87</td>
<td>8.50</td>
<td>39.66</td>
</tr>
<tr>
<td></td>
<td>w/o Meta Prompt</td>
<td>53.46</td>
<td>8.56</td>
<td>40.04</td>
</tr>
<tr>
<td>Task prompt</td>
<td>w/o Sched. Sampling</td>
<td>55.15</td>
<td>7.43</td>
<td>42.00</td>
</tr>
<tr>
<td></td>
<td>w/o G.T. Identity</td>
<td>54.16</td>
<td>7.61</td>
<td>41.27</td>
</tr>
<tr>
<td></td>
<td>w/o Neg. Samples</td>
<td>54.97</td>
<td>7.66</td>
<td>41.78</td>
</tr>
<tr>
<td></td>
<td>w/o ADB</td>
<td>55.48</td>
<td>6.98</td>
<td>41.01</td>
</tr>
<tr>
<td>Meta prompt</td>
<td>w/o Sample Dive.</td>
<td>55.24</td>
<td>6.91</td>
<td>41.23</td>
</tr>
<tr>
<td></td>
<td>w/o Memory Dive.</td>
<td>55.02</td>
<td>7.41</td>
<td>41.48</td>
</tr>
<tr>
<td></td>
<td>w/o Loc.</td>
<td>54.70</td>
<td>7.54</td>
<td>41.16</td>
</tr>
<tr>
<td></td>
<td>w/o Cluster</td>
<td>54.46</td>
<td>6.99</td>
<td>41.51</td>
</tr>
<tr>
<td>Diana</td>
<td></td>
<td>55.93</td>
<td>6.75</td>
<td>42.12</td>
</tr>
</tbody>
</table>

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

We also analyze the distribution of meta prompt keys $K = \{k^j_m\}_{j=1}^M$ 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:

$$\text{Diversity} = \frac{1}{Z} \sum_{j=1}^{M} \frac{\left| N_Z(k^j_m,M) \right|}{(Z \times M)},$$  

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

$$\text{Locality} = \sum_{q \in M} \sum_{k \in N_Z(q,K)} (1 - |{q, k}|) / (Z \times |M|).$$

High locality scores are received if meta prompt keys in $K$ 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.

#### 4.7.2 Distribution of Meta Prompt Keys

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Models</th>
<th>$Z=2$</th>
<th>$Z=3$</th>
<th>$Z=5$</th>
<th>$Z=10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locality</td>
<td>w/o Sample Dive.</td>
<td>0.73</td>
<td>0.72</td>
<td>0.70</td>
<td>0.48</td>
</tr>
<tr>
<td>w/o Memory Dive.</td>
<td>0.74</td>
<td>0.72</td>
<td>0.69</td>
<td>0.63</td>
<td></td>
</tr>
<tr>
<td>Diana</td>
<td>0.74</td>
<td>0.73</td>
<td>0.70</td>
<td>0.66</td>
<td></td>
</tr>
<tr>
<td>Diversity</td>
<td>w/o Sample Dive.</td>
<td>0.63</td>
<td>0.61</td>
<td>0.59</td>
<td>0.40</td>
</tr>
<tr>
<td>w/o Memory Dive.</td>
<td>1.00</td>
<td>0.89</td>
<td>0.77</td>
<td>0.53</td>
<td></td>
</tr>
<tr>
<td>Diana</td>
<td>1.00</td>
<td>0.96</td>
<td>0.89</td>
<td>0.63</td>
<td></td>
</tr>
</tbody>
</table>

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


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


5853


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


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.

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 $P_{o}, P_{f}, P_{i}, P_{m}$ 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 $\eta$ and $\gamma$, we have a grid search between 0 and 0.5 with an interval of 0.05. For $\alpha$ and $\beta$, $\alpha$ is searched among $\{0.9, 0.7, 0.5\}$, while $\beta$ 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 $\{\hat{P}_{i}(F_{t}), \cdots, \hat{P}_{i}(F_{t})\}$ for unseen tasks, we allocate a small probability $\omega = 5\%$ for each training sample $(C, Q, A)$ to use $\hat{P}_{i}(F_{j})$ 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 $||h(C, Q), k_{i}(T_{i})|| > 0.35$, 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 $M$ 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 $k_{m}^{j}$ ($j = 1, \cdots, M$), we select top-⌈$\frac{E}{M}$⌉ samples (⌈·⌉ is the ceiling function), whose query vectors are closest to $k_{m}^{j}$. After accumulating $M[\frac{E}{M}]$ 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 $M$. In this way, the memory $M$ we constructed can expand to the whole space of prompt keys.

Note that, the memory buffer $M$ is optional in Diana. Without $M$, 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) **Extractive QA**, 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 (Kocijsky 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 (Mi- haylov 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 pre-process 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-
In this section, we analyze the task identity detection accuracy yield by the above schemes in the decaNLP task set. The above approaches are trained and evaluated on unseen tasks for 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 (fEM). 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).

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
<table>
<thead>
<tr>
<th>Task-ID in Test</th>
<th>Available</th>
<th>Unavailable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>Buffer</td>
<td>$R_{N,j}$</td>
</tr>
<tr>
<td>ProQA</td>
<td>71.09</td>
<td>37.39</td>
</tr>
<tr>
<td>ProQA+ER</td>
<td>75.57</td>
<td>50.98</td>
</tr>
</tbody>
</table>

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).

<table>
<thead>
<tr>
<th>Task-ID in Test</th>
<th>Available</th>
<th>Unavailable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>Buffer</td>
<td>$R_{N,j}$</td>
</tr>
<tr>
<td>Finetune</td>
<td>68.09</td>
<td>19.70</td>
</tr>
<tr>
<td>EWC</td>
<td>70.57</td>
<td>35.97</td>
</tr>
<tr>
<td>FLCB</td>
<td>70.96</td>
<td>33.35</td>
</tr>
<tr>
<td>AdapterCL</td>
<td>71.82</td>
<td>35.14</td>
</tr>
<tr>
<td>L2P</td>
<td>70.18</td>
<td>34.62</td>
</tr>
<tr>
<td>DualPrompt</td>
<td>70.99</td>
<td>35.33</td>
</tr>
<tr>
<td>ER</td>
<td>73.65</td>
<td>47.96</td>
</tr>
<tr>
<td>DER++</td>
<td>74.18</td>
<td>49.27</td>
</tr>
<tr>
<td>AFPER</td>
<td>75.27</td>
<td>48.90</td>
</tr>
<tr>
<td>Diana w/o $M$</td>
<td>71.94</td>
<td>36.25</td>
</tr>
<tr>
<td>Diana</td>
<td>76.93</td>
<td>51.09</td>
</tr>
<tr>
<td>Multitask</td>
<td>-</td>
<td>79.68</td>
</tr>
</tbody>
</table>

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).

<table>
<thead>
<tr>
<th>Methods</th>
<th>Buffer</th>
<th>$R_{N,j}$</th>
<th>$A_N$</th>
<th>$F_N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProQA</td>
<td>67.66</td>
<td>38.73</td>
<td>37.96</td>
<td>37.72</td>
</tr>
<tr>
<td>ProQA+ER</td>
<td>71.20</td>
<td>40.17</td>
<td>41.94</td>
<td>39.00</td>
</tr>
</tbody>
</table>

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).

Figure 4 when learning the last task $T_N$ in the put task sequence of QA task set. We can observe
Table 10: Task identity detection performance of different models under the QA tasks.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Methods</th>
<th>Scores on Seen Tasks</th>
<th>Scores on Unseen Tasks</th>
<th>Overall Scores</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>F1 Accuracy</td>
<td>F1 Accuracy</td>
<td>F1 Accuracy</td>
</tr>
<tr>
<td>Closed-world</td>
<td>Perplexity-based</td>
<td>44.92</td>
<td>52.20</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Distance-based</td>
<td>43.18</td>
<td>63.34</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Advanced distance-based</td>
<td>54.37</td>
<td>75.35</td>
<td>-</td>
</tr>
<tr>
<td>Open-world</td>
<td>Perplexity-based</td>
<td>33.15</td>
<td>58.64</td>
<td>26.14</td>
</tr>
<tr>
<td></td>
<td>Distance-based</td>
<td>38.51</td>
<td>50.53</td>
<td>21.98</td>
</tr>
<tr>
<td></td>
<td>Advanced distance-based</td>
<td>44.12</td>
<td>64.86</td>
<td>24.17</td>
</tr>
<tr>
<td></td>
<td>Diana</td>
<td>47.06</td>
<td>68.81</td>
<td>35.70</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Tunable Parameters</th>
<th>Memory Size</th>
<th>Train Time Per Batch</th>
<th>Test Time All Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower Bound</td>
<td>222.90M</td>
<td>0</td>
<td>0.55</td>
<td>523</td>
</tr>
<tr>
<td>EWC</td>
<td>222.90M</td>
<td>0</td>
<td>0.93</td>
<td>596</td>
</tr>
<tr>
<td>FLGB</td>
<td>222.90M</td>
<td>0</td>
<td>0.59</td>
<td>591</td>
</tr>
<tr>
<td>AdapterCL</td>
<td>262.25M</td>
<td>0</td>
<td>0.73</td>
<td>5852</td>
</tr>
<tr>
<td>L2P</td>
<td>223.39M</td>
<td>0</td>
<td>1.01</td>
<td>1013</td>
</tr>
<tr>
<td>DualPrompt</td>
<td>223.17M</td>
<td>0</td>
<td>0.93</td>
<td>1147</td>
</tr>
<tr>
<td>ER</td>
<td>222.90M</td>
<td>50</td>
<td>0.58</td>
<td>541</td>
</tr>
<tr>
<td>DER++</td>
<td>222.90M</td>
<td>50</td>
<td>0.68</td>
<td>604</td>
</tr>
<tr>
<td>APFPER</td>
<td>222.90M</td>
<td>50</td>
<td>0.95</td>
<td>630</td>
</tr>
<tr>
<td>ProQA</td>
<td>223.43M</td>
<td>0</td>
<td>0.86</td>
<td>863</td>
</tr>
<tr>
<td>Diana</td>
<td>223.84M</td>
<td>50</td>
<td>1.05</td>
<td>1108</td>
</tr>
<tr>
<td>Diana w/o M</td>
<td>223.84M</td>
<td>0</td>
<td>0.97</td>
<td>1123</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Method</th>
<th>AN</th>
<th>FN</th>
<th>AN'</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prompt tuning</td>
<td>46.76</td>
<td>4.71</td>
<td>32.87</td>
</tr>
<tr>
<td>Full tuning</td>
<td>55.93</td>
<td>6.75</td>
<td>46.71</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Method</th>
<th>AN</th>
<th>FN</th>
<th>AN'</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prompt tuning</td>
<td>46.76</td>
<td>4.71</td>
<td>32.87</td>
</tr>
<tr>
<td>Full tuning</td>
<td>55.93</td>
<td>6.75</td>
<td>46.71</td>
</tr>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>Format</th>
<th>Dataset</th>
<th>Case</th>
</tr>
</thead>
</table>
| 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.  
**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.
<table>
<thead>
<tr>
<th>Format</th>
<th>Dataset</th>
<th>Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extractive</td>
<td>SQuAD</td>
<td>Context: (Private_school) Private schooling in the United States has been...&lt;br&gt;Question: In what year did Massachusetts first require children to be educated in schools?&lt;br&gt;Answer: 1852</td>
</tr>
<tr>
<td></td>
<td>NewsQA</td>
<td>Context: ABECHE, Chad (CNN) – Most of the 103 children that a French charity...&lt;br&gt;Question: WHO ARE UNDER ARREST IN CHAD?&lt;br&gt;Answer: Three French journalists, a seven-member Spanish flight crew and one Belgian</td>
</tr>
<tr>
<td></td>
<td>Quoref</td>
<td>Context: (Blast of Silence) Frankie Bono, a mentally disturbed hitman from Cleveland...&lt;br&gt;Question: What is the first name of the person who follows their target to select...?&lt;br&gt;Answer: Frankie</td>
</tr>
<tr>
<td>Abstractive</td>
<td>NarQA</td>
<td>Context: The play begins with three pages disputing over the black cloak usually worn by the actor...&lt;br&gt;Question: WHO NORMALLY DELIVERS THE OPENING PROLOGUE IN THE PLAY?&lt;br&gt;Answer: THE ACTOR WEARING THE BLACK CLOAK</td>
</tr>
<tr>
<td></td>
<td>NQOpen</td>
<td>Context: cartilage - cartilage cartilage is a resilient and smooth elastic tissue, a rubber...&lt;br&gt;Question: where is each type of cartilage located in the body?&lt;br&gt;Answer: many other body components</td>
</tr>
<tr>
<td></td>
<td>Drop</td>
<td>Context: Hoping to rebound from their loss to the Patriots, the Raiders stayed at home for a Week...&lt;br&gt;Question: How many field goals did both teams kick in the first half?&lt;br&gt;Answer: 2</td>
</tr>
</tbody>
</table>
| Multiple-Choice | RACE | Context: It’s cool, and it’s hot, and everyone is doing it. People talk about it often, and friends...<br>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<br>Answer: who writes diaries online |
| | OBQA | Context: Null<br>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<br>Answer: Deep sea animals |
| | MCTest | Context: It was Jessie Bear’s birthday. She was having a party...<br>Question: Who was having a birthday?<br>Answer: Jessie Bear |
| | SIQA | Context: Tracy didn’t go home that evening and resisted Riley’s attacks<br>Question: What does Tracy need to do before this?<br>Answer: (A) make a new plan (B) Go home and see Riley (C) Find somewhere to go (D) To change her job. |
| | Dream | Context: M: How long have you been teaching in this middle school? W: For ten years...<br>Question: What’s the woman probably going to do?<br>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 $g_\theta$, 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 $\{P_m^n\}_{i=1}^{M}$, task prompt keys $\{k_t(T_i)\}_{i=1}^{N}$, meta prompt keys $\{k^i_m\}_{i=1}^{M}$

1: Initialize: $\mathcal{M} \leftarrow \emptyset$
2: for Each task $T_i$, $i = 1, \cdots, N$ do
3:     if $\mathcal{M} \neq \emptyset$ then
4:         Calculate cluster centroids $c_1, \cdots, c_B$ of $\mathcal{M}$
5:     end if
6: for number of training epochs do
7:     for Each mini-batch $I \in \{(C_j, Q_j, A_j)\}_{j=1}^{n_i} \cup \mathcal{M}$ do
8:         Obtain $\epsilon_k$ by Eq. 5
9:     end for
10:     for $(C, Q, A) \in I$ do
11:         Obtain format $F_j$ of $(C, Q, A)$
12:         Sample $\epsilon, \zeta$ from $U(0, 1)$
13:         if $\zeta < \omega$ then
14:             $P_t(C, Q) \leftarrow \hat{P}_t(F_j)$ (Use task prompt $\hat{P}_t(F_j)$ for unseen tasks)
15:         else if $\epsilon < \epsilon_k$ then
16:             $P_t(C, Q) \leftarrow P_t(T_i)$ (Use the golden truth task identity to select task prompt)
17:         else
18:             $P_t(C, Q) \leftarrow P_t(\arg\min_{T_i \in \{T_1, \cdots, T_i\}}(||q, k_t(T_i)||))$ (Use the inferred task identity to select task prompt)
19:     end if
20:     $S(C, Q) \leftarrow$ indexes of $M'$ meta prompt keys that are closest to $q$
21:     $P_m(C, Q) \leftarrow \{P_{m_j}^n\}_{j \in S(C, Q)}$
22:     $P(C, Q) \leftarrow \{P_g; P_f(F_j); P_t(C, Q); P_m(C, Q)\}$
23:     Calculate per sample loss $L_{LM}$ on $g_\theta$ and $P(C, Q)$ by Eq. 6
24:     Obtain negative sample $(C_n, Q_n)$ from $\mathcal{M}$ by Eq. 2
25:     Calculate per sample loss $L_t$ on $k_t(T_i)$ by Eq. 1
26:     Calculate per sample loss $L_m$ on $\{k_m^n\}_{s_j \in S(C, Q)}$ by Eq. 3
27:     if $(C, Q, A) \in \mathcal{M}$ then
28:         Calculate per sample loss $L'_m$ on $\{k_m^n\}_{s_j \in S(C, Q)}$ by Eq. 4
29:     end if
30: end for
31: Update $g_\theta$ and prompts with accumulated $L_{LM}$
32: Update task prompt keys $\{k_t(T_i)\}_{i=1}^{N}$ with accumulated $L_t$
33: Update meta prompt keys $\{k^i_m\}_{i=1}^{M}$ with accumulated $L_m$ and $L'_m$
34: end for
35: Update $\mathcal{M}$ with $\{(C_j, Q_j, A_j)\}_{j=1}^{n_i}$ according to details in Appendix B
36: end for
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?
  *Section 4, 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.