Continuous QA Learning with Structured Prompts

Yinhe Zheng
Lingxin AI, Beijing, China
zhengyinhe1@163.com

Abstract

QA models with lifelong learning (LL) abilities are important for practical QA applications, and architecture-based LL methods are reported to be an effective implementation for these models. However, it is non-trivial to extend previous approaches to QA tasks since they either require access to task identities in the testing phase or do not explicitly model samples from unseen tasks. In this paper, we propose Diana: a dynamic architecture-based lifelong QA model that tries to learn a sequence of QA tasks with a prompt enhanced language model. Four types of hierarchically organized prompts are used in Diana to capture QA 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 different 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 lifelong QA models, especially in handling unseen tasks.

1 Introduction

Question Answering (QA) is important for advanced AI applications (Yang et al. 2019), and current QA models generally obtain high performance when learning fixed QA tasks (Soares and Parreiras 2020). However, these models may not be practical in real applications, where models are usually required to support new QA tasks. Training a learned QA model on new tasks will lead to the catastrophic forgetting issue, i.e., models forget previously learned knowledge when learning new tasks (French 1999). Without resorting to the expensive re-training process, Lifelong Learning (LL) enables the QA model to continuously learn new tasks while preserving previously learned knowledge (Thrun and Mitchell 1995). Therefore, developing QA models with LL abilities in practical applications is important.

An effective implementation of LL is the architecture-based approach (Chen, Goodfellow, and Shlens 2016; Rusu et al. 2016; Fernando et al. 2017; Wistawarakoses and Berrar 2020; Qin and Joty 2022), in which task-specific components are maintained to preserve learned knowledge (Mancini et al. 2018). Recently, some studies proposed to convert QA tasks into a unified language modeling (LM) format (Khashabi et al. 2020), and these studies have achieved promising QA performances with the help of prompts (Brown et al. 2020; Jiang et al. 2020) and adapter (Houlsby et al. 2019) modules. Some works have also tried to extend these approaches to build lifelong QA models by sharing the same underlying pre-trained language model (PLM) between different tasks (Su et al. 2020). These models can learn a sequence of QA tasks with different QA formats (Zhong et al. 2022).

However, applying the above approaches in building lifelong QA models has two major problems. First, most above architecture-based LL approaches need to access the task identity of the testing sample (Wang et al. 2022a,b). This setting limits the application of lifelong QA models because task identities for input questions are unavailable in most practical scenarios (Geigle et al. 2021). Second, most previous LL models are tested only on seen tasks and do not consider testing samples from unseen tasks (Boult et al. 2019; Mundt et al. 2020). This setting is violated in practical QA applications because users usually pose questions that do not belong to previously seen tasks.

There are two kinds of methods to tackle above problems:

(1) Some models organize their components at the task-level and use a classifier to determine the task identity for each testing sample (Wortsman et al. 2020; Madotto et al. 2021), i.e., each task is assigned with a separate component that will be activated based on the predicted task identity. This scheme generally yields high LL performances if the task identity is correctly predicted because each task is modeled by a dedicated component. However, these models usually exhibit poor generalization performance in QA applications because it is infeasible to determine task identities for questions that do not belong to any previously seen QA tasks. Moreover, the previously learned knowledge is isolated in these task-specific components and cannot be shared to tackle questions from unseen tasks (Rogers, Gardner, and Augenstein 2021).

(2) Some models are organized at the instance-level and use a dynamic architecture for each input sample (Wistawarakoses and Berrar 2020), i.e., a pool of fine-grained components are maintained and dynamically combined in each forward pass based on the input instance. This scheme avoids the inconvenience of explicitly determining the task identity for each testing sample (Wang et al. 2020).
2022b) and helps LL models tackle unseen tasks since the learned knowledge is distributed into different model components (Träuble et al. 2022). However, these models usually yield sub-optimal 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 categories and propose Diana: a dynamic architecture-based lifelong QA model. We follow previous approaches to convert QA tasks into a unified LM format and propose to learn these tasks using a prompt-enhanced PLM. Four types of hierarchically organized prompts are maintained in Diana (i.e., General Prompt, Format Prompt, Task Prompt, and Meta Prompt) to capture QA knowledge from different granularities. Specifically, the general prompt is used for all QA tasks, and the format prompts are shared between tasks in the same QA format. Moreover, a task prompt is assigned for each incoming QA task, and a pool of meta prompts are maintained and dynamically combined when handling each sample. In this way, Diana can better generalize to unseen tasks while achieving high LL performances since its components are organized at both task-level and instance-level.

Further, we allocate separate prompts for unseen tasks and learn a key vector for each task prompt and meta prompt to better share knowledge between different tasks so that samples from unseen tasks can be explicitly modeled.

We perform extensive experiments on 11 benchmark QA tasks across three different formats and further test the generalization performance of our model on three unseen tasks. Results indicate that Diana outperforms state-of-the-art (SOTA) baselines on all benchmarks, especially when generalizing to unseen tasks. Our main contributions are:

- We propose Diana: a novel architecture-based lifelong QA model that uses four types of hierarchically organized prompts to capture knowledge in different granularities. Both task-level and instance-level components are maintained in Diana so that it can better generalize to unseen tasks while achieving high LL performance.
- We are the first to explicitly model unseen tasks in lifelong QA models. Specifically, we dedicate separate prompts for unseen tasks, and build prompt keys to facilitate knowledge sharing between different tasks.
- Extensive experiments show that Diana outperformed SOTA baselines in building lifelong QA models.

2 Related Work

2.1 Lifelong Learning

LL aims at incrementally acquiring new knowledge without catastrophically forgetting previously learned ones. Generally, three categories of methods are proposed: 1. Rehearsal-based methods (Rebuffi et al. 2017; Shin et al. 2017; Sun, Ho, and Lee 2019; 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, Poole, and Ganguli 2017; Li and Hoiem 2017; Ritter, Botev, and Barber 2018; Farajtabar et al. 2020) consolidate model parameters that are important to previous tasks by introducing additional regularization terms; 3. Architecture-based methods (Chen, Goodfellow, and Shlens 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 categories based on whether the task identity is provided for each testing sample 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 identity is not provided for each testing sample. One line of methods in this category attempt to detect the task identity for each input sample (Madotto et al. 2021). 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.

2.2 Domain Generalization

Domain Generalization (DG) aims to learn a model from several seen domains that will generalize well on unseen testing domains (Muandet, Balduzzi, and Schölkopf 2013; Xu et al. 2014; Ghifary et al. 2015; Li et al. 2018; Zhao et al. 2020). An effective approach for DG maintains separate components to model domain-specific knowledge and open domain knowledge, respectively (Khosla et al. 2012; Li et al. 2017, 2019). Our method allocates additional prompts for unseen tasks to improve LL generalization performance.

2.3 Unified Question Answering

Question Answering (QA) tasks have been posed in different formats (Sun et al. 2019), such as Extractive, Abstractive, and Multiple-Choice. To encourage knowledge sharing across these formats, existing approaches attempt to build a unified QA model by casting different QA tasks into a unified text-to-text format (Khashabi et al. 2020; McCann et al. 2019). The most similar previous work compared to ours is ProQA (Zhong et al. 2022), where different QA tasks are unified and a set of structured prompts are used to learn these tasks. However, LL experiments in ProQA only consider two tasks and assume task identities for testing samples. In this work, we focus on the domain-incremental learning setting, where task identity is not provided for each testing sample (Muandet, Balduzzi, and Schölkopf 2013; Xu et al. 2014; Ghifary et al. 2015; Li et al. 2018; Zhao et al. 2020). An effective approach for DG maintains separate components to model domain-specific knowledge and open domain knowledge, respectively (Khosla et al. 2012; Li et al. 2017, 2019). Our method allocates additional prompts for unseen tasks to improve LL generalization performance.

3 Method

3.1 Task Formulation

In this study, we aim to sequentially learn \(N\) QA 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 QA format \(F_j\), 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
an input context and 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. 2022) and use a prompt-enhanced encoder-decoder model \(g_0\) to learn each task \(T_1\) in Diana. We use soft prompts (Liu et al. 2021; Lester, Al-Rfou, and Constant 2021; Vu et al. 2022) in our study, i.e., each prompt is a sequence of trainable embeddings that are randomly initialized and optimized when learning each incoming task. For each training sample \((C, Q, A)\) from \(T_i\), we first construct a prompt \(P(C, Q)\) based on \(C\) and \(Q\). Then the encoder takes in the concatenation of \(P(C, Q), C\), and \(Q\) and the decoder predicts \(A\), i.e., \(A = g_0[[P(C, Q); C, Q]]\), in which “[:]” denotes the sequence concatenation operation.

A hierarchical prompt structure is maintained in Diana to construct \(P(C, Q)\), i.e., \(P(C, Q)\) is a concatenation of four kinds of prompts \([P_g; P_f(F_j); P_i(T_i); P_m(C, Q)]\), in which \(F_j\) is the format of \(T_i\), \(P_g\) is a general prompt, \(P_f(F_j)\) is a format prompt, \(P_i(T_i)\) is a task prompt and \(P_m(C, Q)\) is a combined meta prompt. Specifically, To encode task-agnostic knowledge of QA, we use one general prompt \(P_g\) for all incoming tasks; To capture knowledge that is shared by tasks in the same format, we assign a format-specific prompt \(P_f(F_j)\) for tasks that are in the same format \(F_j\), \((j=1, \cdots, L)\); To learn knowledge associated with each task, we allocate a separate task prompt \(P_i(T_i)\) for each task \(T_i\), \((i=1, \cdots, N)\); To capture fine-grained knowledge distributed in each input sample \((C, Q, A)\), we maintain \(M\) meta prompts \([P_m]_{m=1}^M\) and dynamically combine these prompts based on \(C\) and \(Q\) to obtain \(P_m(C, Q)\). Moreover, to explicitly model samples from unseen tasks, we enlarge the set of task prompts with \(L\) extra prompts \(P_i(F_1), \cdots, P_i(F_L)\), in which each prompt \(P_i(F_j)\) models the unseen task for a particular format \(F_j\).

To tackle the problem of lacking task identities in the testing phase, we associate a key vector \(k_i(T_i)\) and \(k_{m,i}\) to each task prompt \(P_i(T_i)\) and meta prompt \(P_m\), respectively. A fixed query function \(h\) is built to map a context \(C\) and question \(Q\) to a query vector \(q = h(C, Q)\). Specifically, \(h\) is initialized by a fixed PLM and not tuned in the training process so that the semantic encoded in the query vector \(q\) does not drift when learning different tasks. For a given input \(C\) and \(Q\), the corresponding task prompt \(P_i(T_i)\) in \(P(C, Q)\) can be determined by retrieving the most similar task key vector \(k_i(T_i)\) using \(q\), and the combined meta prompt \(P_m(C, Q)\) in \(P(C, Q)\) can be constructed by based on the similarities between meta key vectors \(k_{m,i}\) and \(q\). Note that we do not allocate key vectors for these \(L\) extra task prompts \(P_i(F_1), \cdots, P_i(F_L)\) that are designed to model unseen tasks. These prompts are used in Diana with an explicit unseen task determination process (see more details in Section 3.4).

A two-stage learning process is introduced in Diana when handling each training sample \((C, Q, A)\). The first stage focuses on learning a representation space for prompt keys so that we can determine proper prompts to construct \(P(C, Q)\) by matching the query vector and each prompt key. 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 in Diana so that we can construct the concatenated prompt \(P(C, Q)\) based on the input context \(C\) and question \(Q\). 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 corresponding to a given input context \(C\) and question \(Q\). Specifically, we first calculate the query vector \(q = h(C, Q)\) and determine the most similar task prompt key \(k_i(T_i)\) to \(q\). The task prompt \(P_i(T_i)\) associated with \(k_i(T_i)\) is used to construct \(P(C, Q)\). Ideally, the key vector \(k_i(T_i)\) for a task prompt \(P_i(T_i)\) should be located near the query vectors of samples from task \(T_i\) and distant to the query vectors of samples from other tasks \(T_j\) \((j \neq i)\). Therefore, when learning each task \(T_i\), we maintain a small memory buffer \(M\) for samples from previously learned tasks \(T_j\) \((j < i)\), and design the following exponential angular triplet loss (Ye et al. 2021) for each training sample \((C, Q, A)\) from \(T_i\):

\[
\mathcal{L}_1 = \exp(||h(C, Q), k_i(T_i)|| + \max(1 - ||h(C_n, Q_n), k_i(T_i)||, 0)),
\]

in which the operator \(||., ||\) 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 \(M\):

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

**Meta Prompt Keys** help to combine each meta prompt \(P_m\) to produce \(P_m(C, Q)\). Specifically, for each input context \(C\) and question \(Q\), we select \(M'\) meta prompt keys that are closest to the query vector \(q = h(C, Q)\), and concatenate these \(M'\) associated meta prompts to obtain \(P_m(C, Q)\). 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*. Specifically, the diversity property aims to distribute meta prompt keys to the whole vector space so that every meta prompt can be involved in the training process. The locality property aims to group similar prompt keys to clusters 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:

\[ L_m = \sum_{i \in S(C, Q)} \max(0, \|k_{m, i} - h(C, Q)\| - \eta) + \sum_{i, j \in S(C, Q)} \max(0, \gamma - \|k_{m, i} - k_{m, j}\|/M^2), \]  

where \( S(C, Q) \) is the index set of these \( M' \) selected meta prompt keys that are closest to \( h(C, Q) \). \( \eta \) and \( \gamma \) are scalar hyper-parameters to control the distance margin. Specifically, the first term in the above equation enforces the locality property by pulling these selected \( M' \) 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 the above loss in Eq. 3 only involves individual query vectors. Thus meta prompt keys learned using this loss may not be diverse enough 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 \( M \), in which samples from previously learned tasks are contained. Specifically, when learning the task \( T_i \), we first calculate query vectors for samples in \( M \) and then group these query vectors into \( B \) clusters (we set \( B = 5i \) in our experiments, where \( i \) is the number of received tasks). Centroids of these \( B \) clusters are calculated as \( c_1, \ldots, c_B \). For each \( C \) and \( Q \) from \( M \), the subsequent loss is optimized:

\[ L'_m = \sum_{i \in S(C, Q)} \max(0, \|k_{i, c_k}\| - \eta), \]  

where \( c_k \) is the cluster centroid to which \( C \) and \( Q \) belong. The above loss enforces the global diversity by scattering these meta prompt keys to each centroid.

### 3.4 Model Training

**Scheduled Sampling of Task Prompts** When training Diana, we are given the task identity of each training sample to directly obtain the corresponding task prompt \( P_t(T_i) \) for \( P(C, Q) \). However, naively using these golden truth task identities leads to an exposure bias issue, i.e., task prompts used in the testing phase may not be correct since we need to infer the task identity for each testing sample.

In this study, we introduce a scheduled sampling process to tackle the above exposure bias issue when learning each task \( T_i \). Specifically, for a given sample \((C, Q, A)\) in the \( k \)-th training step, we toss a coin and use the golden truth task identity with probability \( \epsilon_k \), or use the task identity inferred based on task prompt keys with probability \( 1 - \epsilon_k \) (Bengio et al. 2015). Note that when starting to learn each task, the corresponding prompt key is not well optimized, and thus the selected task identity is not accurate. Therefore, we set the value of \( \epsilon_k \) to favor the golden truth task identity at the beginning (i.e., when \( k \) is small) and gradually switch to the inferred task identity 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 lifelong QA 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 \( F_t \) based on \( C \) and \( Q \), 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 \( P_t(F_j) \) and expect this knowledge can better generalize to unseen tasks.

**Train with QA Loss** For each training sample \((C, Q, A)\), we first construct the prompt \( P(C, Q) \) for \( C \) and \( Q \), and then optimize \( P(C, Q) \) together with the encoder-decoder model \( g_{\theta} \) using the following loss:

\[ L_{QA} = -\log g_{\theta}(A|[P(C, Q); C, Q]). \]  

The overall loss that we optimize for Diana is:

\[ L = L_m + L'_m + L_t + L_{QA}. \]  

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 \( M \). This selection process aims to maintain diverse samples in \( M \). More details are in Appendix B.

### 3.5 Model Inference

In the testing phase, we first determine the prompt \( P(C, Q) \) for each input context \( C \) and question \( Q \), and then predict the answer sequence \( A \) using the learned model \( g_{\theta} \).

**Adaptive Decision Boundary for Each Task** When selecting the proper task prompt in the testing phase, we propose constructing an adaptive decision boundary (ADB) for each task \( T_i \) to determine whether the input belongs to unseen tasks. Specifically, for \( T_i \), a scalar boundary \( \delta_i \) is constructed following the approach proposed by Zhang, Xu, and Lin (2021). An input context \( C \) and question \( Q \) are regarded as a sample from unseen tasks if its query vector \( h(C, Q) \) falls outside the boundary of every task, i.e.,

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

For samples from unseen tasks, we use the prompt \( \hat{P}_t(F_j) \) to construct \( P(C, Q) \), where \( F_j \) is the format of \( C \) and \( Q \).

**Answer Prediction** The answer sequence \( A \) is predicted from the prompt-enhanced encoder-decoder model with a greedy decoding process:

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

### 4 Experiments

**4.1 Datasets and Metrics**

**Datasets** We carry out experiments on 11 benchmark datasets covering 3 QA formats: (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 (Kocijany 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, Burges, and Renshaw 2013), SIQA (Sap et al. 2019), and Dream (Sun et al. 2019). We regard each dataset as an individual QA task and reserve $N' = 3$ tasks as unseen tasks (Quoref, Drop, and Dream). Our model is trained on the rest of $N = 8$ seen tasks while tested on all 11 tasks. The task identity for each sample is not available when testing.

**Evaluation Metrics** The evaluation of the above tasks follows Zhong et al. (2022). Specifically, we compute the accuracy of option selection for all Multi-Choice QA tasks and use Exact Match (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.

When learning each task, 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 $T_i$. Based on $R$, we compute the following metrics to evaluate the LL performance:

**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:

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

**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 scores of $A_N$ and $F_N$ are averages of these five runs. Moreover, we also report the averaged **Final Performance** $R_{N,j}$ of each task $T_j$ over these 5 runs. Ideally, we expect a strong lifelong QA model to yield high $A_N$ and $R_{N,j}$, while obtaining low $F_N$.

**4.2 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 use a fixed T5-base encoder with an average pooling layer to obtain the query vector. 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. See more details in Appendix A.

**4.3 Baselines**

We use the following competitive baselines:

- **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; 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; Architecture-based methods: AdapterCL (Madotto et al. 2021) allocates separate adapter modules 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. 2022) uses a unified structural prompt to implement lifelong QA models. Note that ProQA requires obtaining task identities 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 QA 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.

All above baselines are implemented following same settings of our model, including the backbone PLM, prompt size, and memory size used for replay. Note that for the ProQA baseline, we follow its original setting to provide task identities for testing samples when evaluating.

**4.4 Experiment Results**

**Results on Seen Tasks** Table 1 shows the result on eight seen tasks. Diana outperforms all competitive baselines. When task identities are unavailable, Diana outperforms the best performing baseline AFPER with a large margin, i.e., 6.15% relative improvement on the $A_N$ score and 27.26% relative decrease on the $F_N$ score. We can also observe that: (1) Diana even outperforms the ProQA+ER baseline, which lacks task identities in testing. This proves the superiority of our model design. (2) When task identities 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 three unseen tasks. Diana yields the best performances on all metrics. It also achieves a relative improvement of 9.49% on the $A_N$ score compared with the best baseline DER++. 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 the effectiveness of using hierarchical prompts to explicitly model unseen tasks. (2) It is interesting to see that our model even outperforms Multitask when the $\mathcal{M}$ is available. This further proves that our model is effective in modeling unseen tasks.
### 4.5 Ablation Studies

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

1. Each prompt type 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 identities in training; w/o G.T. Identity is similar to the above variant. Instead it only uses predicted task identities 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 on each sample by removing the second term in Eq.3; w/o Memory Dive, does not exploit memory samples to enhance the diversity property by removing the loss $L_m$ (Eq.4); 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 that Diana outperforms all above variants. We can also observe that: (1) “w/o Meta Prompt” achieves low performance. This indicates that these fine-grained meta prompts are more important in building lifelong QA 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., $A_{N,j}$) 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.

### 4.6 More Analysis

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

Table 3: Ablation studies of model components and training strategies. Each result is an average of 5 random runs.
Table 4: Quantitative analysis of the locality and diversity for meta prompt keys.

| 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  |

The help of several distance-based regularization terms. Dedicated components are also allocated in Diana to model samples from unseen tasks. Experiments and empirical analysis on benchmark QA tasks show that Diana outperforms SOTA lifelong QA models, especially in handling samples from unseen tasks.

References

Bengio, S.; Vinyals, O.; Jaitly, N.; and Shazeer, N. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. Advances in neural information processing systems, 28.

Boul, T. E.; Cruz, S.; Dhamija, A. R.; Gunther, M.; Hendry, J.; and Scheirer, W. J. 2019. Learning and the unknown: Surveying steps toward open world recognition. In Proceedings of the AAAI conference on artificial intelligence, volume 33, 9801–9807.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., Advances in Neural Information Processing Systems, volume 33, 1877–1901. Curran Associates, Inc.

Buzzeza, P.; Boschini, M.; Porrello, A.; Abati, D.; and Calderara, S. 2020. Dark Experience for General Continual Learning: a Strong, Simple Baseline. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., Advances in Neural Information Processing Systems, volume 33, 15920–15930. Curran Associates, Inc.

Chaudhry, A.; Ranzato, M.; Rohrbach, M.; and Elhoseiny, M. 2019a. Efficient lifelong learning with a-gem. In Proceedings of ICLR.

Chaudhry, A.; Rohrbach, M.; Elhoseiny, M.; Ajanthan, T.; Dokania, P. K.; Torr, P. H. S.; and Ranzato, M. 2019b. On Tiny Episodic Memories in Continual Learning. arXiv:1902.10486.

Chen, T.; Goodfellow, I.; and Shlens, J. 2016. Net2Net: Accelerating Learning via Knowledge Transfer. In Proceedings of ICLR.

Dasigi, P.; Liu, N. F.; Marasović, A.; Smith, N. A.; and Gardner, M. 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), 5925–5932. Hong Kong, China: Association for Computational Linguistics.

Dua, D.; Wang, Y.; Dasigi, P.; Stanovsky, G.; Singh, S.; and Gardner, M. 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.
Maltoni, D.; and Lomonaco, V. 2019. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116: 56–73.

Mancini, M.; Ricci, E.; Caputo, B.; and Rota Bulo, S. 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, 0–0.

Mansoury, M.; Abdollahpouri, H.; Pechenizkiy, M.; Mobasher, B.; and Burke, R. 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, 154–162. New York, NY, USA: Association for Computing Machinery. ISBN 9781450368612.

McCann, B.; Keskar, N. S.; Xiong, C.; and Socher, R. 2019. The Natural Language Decathlon: Multitask Learning as Question Answering.

Mi, F.; Chen, L.; Zhao, M.; Huang, M.; and Faltings, B. 2020. Continual Learning for Natural Language Generation in Task-oriented Dialog Systems. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 3461–3474. Online: Association for Computational Linguistics.

Mihaylov, T.; Clark, P.; Khot, T.; and Sabharwal, A. 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*, 2381–2391. Brussels, Belgium: Association for Computational Linguistics.

Muandet, K.; Balduzzi, D.; and Schölkopf, B. 2013. Domain Generalization via Invariant Feature Representation. In Dasgupta, S.; and McAllester, D., eds., *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, 10–18. Atlanta, Georgia, USA: PMLR.

Mundt, M.; Hong, Y. W.; Plischke, I.; and Ramesh, V. 2020. A Wholistic View of Continual Learning with Deep Neural Networks: Forgotten Lessons and the Bridge to Active and Open World Learning. arXiv:2009.01797.

Pu, N.; Chen, W.; Liu, Y.; Bakker, E. M.; and Lew, M. S. 2021. Lifelong person re-identification via adaptive knowledge accumulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7901–7910.

Qin, C.; and Joty, S. 2022. LFPTS: A Unified Framework for Lifelong Few-shot Language Learning Based on Prompt Tuning of T5. In *International Conference on Learning Representations*.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21: 1–67.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2383–2392. Austin, Texas: Association for Computational Linguistics.

Rebuffi, S.-A.; Kolesnikov, A.; Sperl, G.; and Lampert, C. H. 2017. iCaRL: Incremental classifier and representation learning. In *Proceedings of CVPR*, 2001–2010.

Richardson, M.; Burges, C. J.; and Renshaw, E. 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*, 193–203. Seattle, Washington, USA: Association for Computational Linguistics.

Ritter, H.; Botev, A.; and Barber, D. 2018. Online structured laplace approximations for overcoming catastrophic forgetting. In *Proceedings of NIPS*, 3738–3748.

Rogers, A.; Gardner, M.; and Augenstein, I. 2021. QA Dataset Explosion: A Taxonomy of NLP Resources for Question Answering and Reading Comprehension. arXiv:2107.12708.

Rusu, A. A.; Rabinowitz, N. C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; and Hadsell, R. 2016. Progressive Neural Networks. arXiv:1606.04671.

Sap, M.; Rashkin, H.; Chen, D.; Le Bras, R.; and Choi, Y. 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)*, 4463–4473. Hong Kong, China: Association for Computational Linguistics.

Scellato, S.; Mascolo, C.; Musolesi, M.; and Latora, V. 2010. Distance matters: geo-social metrics for online social networks. In *3rd Workshop on Online Social Networks (WOSN 2010)*.

Shin, H.; Lee, J. K.; Kim, J.; and Kim, J. 2017. Continual learning with deep generative replay. In *Proceedings of NIPS*, 2990–2999.

Soares, M. A. C.; and Parreiras, F. S. 2020. A literature review on question answering techniques, paradigms and systems. *Journal of King Saud University-Computer and Information Sciences*, 32(6): 635–646.

Su, L.; Guo, J.; Zhang, R.; Fan, Y.; Lan, Y.; and Cheng, X. 2020. Continual domain adaptation for machine reading comprehension. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 1395–1404.

Sun, F.-K.; Ho, C.-H.; and Lee, H.-Y. 2019. LAMOL: Language MOdeling for Lifelong Language Learning. arXiv:1909.03329.

Sun, K.; Yu, D.; Chen, J.; Yu, D.; Choi, Y.; and Cardie, C. 2019. Dream: A challenge data set and models for dialogue-based reading comprehension. *Transactions of the Association for Computational Linguistics*, 7: 217–231.

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

Trischler, A.; Wang, T.; Yuan, X.; Harris, J.; Sordoni, A.; Bachman, P.; and Suleman, K. 2017. NewsQA: A Machine Comprehension Dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, 191–200. Vancouver, Canada: Association for Computational Linguistics.
IEEE/CVF Winter Conference on Applications of Computer Vision, 1131–1140.

Zhao, S.; Gong, M.; Liu, T.; Fu, H.; and Tao, D. 2020. Domain Generalization via Entropy Regularization. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., Advances in Neural Information Processing Systems, volume 33, 16096–16107. Curran Associates, Inc.

Wu, W.; Gao, Y.; Ding, N.; Qin, Y.; Liu, Z.; Zhou, M.; Wang, J.; Yin, J.; and Duan, N. 2022. ProQA: Structural Prompt-based Pre-training for Unified Question Answering. arXiv:2205.04040.

Appendix

A 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 $P_0, P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_{10}, P_{11}$ 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 SQuAD, 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 our experiments, 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.

To train extra task prompts $\{\hat{P}_1(F_1), \ldots, \hat{P}_9(F_9)\}$ for unseen tasks, we allocate a small probability $\omega = 5\%$ for each training sample $(C, Q, A)$ to use $P_j(F_j)$ as its task prompt in $P(C, Q, A)$. We 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, Xu, and Lin (2021). We use AdamW optimizer with a learning rate of 0.02 to learn each decision boundary.

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. Specifi-
cally, our selection criteria are built based on the distance of these prompt keys and query vectors. For each meta prompt key \( k^j_m \) \((j = 1, \cdots, M)\), we select top-\([\lceil \frac{E}{N} \rceil] \) samples (\( \lceil \cdot \rceil \) is the ceiling function), whose query vectors are closest to \( k^j_m \). After accumulating \( M \lceil \frac{E}{N} \rceil \) 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 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 evaluated 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 5. 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.

D 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 Figure 1 when learning the last task \( T_N \) in the input task sequence. We can observe 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.

E More Analysis of Computational Cost

We also analyze the computational cost of Diana, including the number of tunable parameters, time used for training and testing, and size of required memories retained from previous tasks. As can be seen from Table 6, Diana does not introduce too much computation overhead.

F Training Process

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

G Dataset Statistics

We perform experiments on 11 benchmark QA tasks across three different formats. The statistics of these 11 datasets
| 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          |
|               | Distance-based        | 43.18                | 63.34                  | 43.18          |
|               | Advanced distance-based | -                   | -                      | 54.37          |
|               |                       | 75.35                |                        | 75.15          |
| Open-world    | Perplexity-based      | 33.15                | 58.64                  | 26.14          |
|               | Distance-based        | 38.51                | 50.53                  | 21.98          |
|               | Advanced distance-based | -                   | -                      | 44.12          |
|               | Diana                 | 47.06                | 68.81                  | 35.70          |

Table 5: Task identity detection performance in different models.

| 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                 |
| AFFER                 | 222.90M            | 50          | 0.95                 | 630                 |
| ProQA                 | 223.43M            | 0           | 0.86                 | 863                 |
| Diana                 | 223.84M            | 50          | 1.05                 | 1108                |
| Diana w/o M           | 223.84M            | 0           | 0.97                 | 1123                |

Table 6: Computational cost of Diana and baselines. “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.

| Format               | Dataset | Train set size | Val set size | Test set size |
|----------------------|---------|----------------|--------------|---------------|
| Extractive           | SQuAD  | 80k            | 7k           | 10k           |
|                      | NewsQA | 76k            | -            | 4.3k          |
|                      | Quoref  | 22k            | -            | 2.7k          |
| Abstractive          | NarQA  | 65k            | 6.9k         | 21k           |
|                      | NQOpen  | 9.6k           | -            | 10k           |
|                      | Drop   | 77k            | -            | 9.5k          |
| Multiple-Choice      | RACE   | 87k            | 4.8k         | 4.9k          |
|                      | OBQA   | 4.9k           | 500          | 500           |
|                      | SIQA   | 33k            | 1.9k         | 2.2k          |
|                      | Dream  | 6.1k           | 2.0k         | 2.0k          |

Table 7: Dataset Statistics.

...are summarized in Table 7. Note that we follow the pre-process scheme released by Khashabi et al. (2020) to tackle these 11 datasets. Some of these datasets do not contain a validation set. We only use the validation sets of SQuAD, NarQA, RACE, OBQA, SIQA and Dream to search hyperparameters.

### H Cases

We list some samples for each task we modeled in Table 8.
| 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 the best possible location?  
**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 16 duel...  
**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 8: Samples extracted from different tasks. Each task contains a context, a question and an answer.
Algorithm 1: Training process of Diana

**Input:** QA model $g_θ$, datasets $\{(C_j, Q_j, A_j)\}_{j=1}^n$ for each task $T_i$ ($i=1, \cdots, N$), memory buffer $M$, general prompt $P_g$, format prompts $\{P_f(F_j)\}_{j=1}^m$, task prompts $\{P_t(T_i)\}_{i=1}^N \cup \{\hat{P}_t(F_j)\}_{j=1}^m$, meta prompts $\{P_m\}_{i=1}^M$, task prompt keys $\{k_t(T_i)\}_{i=1}^N$, meta prompt keys $\{k_m\}_{i=1}^M$.

1: **Initialize:** $M \leftarrow \emptyset$
2: **for** Each task $T_i$, $i=1, \cdots, N$ **do**
3:     if $M \neq \emptyset$ **then**
4:         Calculate cluster centroids $c_1, \cdots, c_B$ of $M$
5:     **end if**
6:     for number of training epochs **do**
7:         for Each mini-batch $t \in \{(C_j, Q_j, A_j)\}_{j=1}^n \cup M$ **do**
8:             Obtain $\varepsilon_k$ by Eq.5
9:             for $(C, Q, A) \in I$ **do**
10:                Obtain format $F_j$ of $(C, Q, A)$
11:                Sample $\epsilon, \zeta$ from $U(0, 1)$
12:                if $\zeta < \omega$ **then**
13:                   $P_t(C, Q) \leftarrow \hat{P}_t(F_j)$ \{Use task prompt $\hat{P}_t(F_j)$ for unseen tasks\}
14:                **else if** $\epsilon < \epsilon_k$ **then**
15:                   $P_t(C, Q) \leftarrow P_t(T_i)$ \{Use the golden truth task identity to select task prompt\}
16:                **else**
17:                   $P_t(C, Q) \leftarrow P_t(\text{argmin}_{T_r \in \{T_1, \cdots, T_i\}}(||q, k_t(T_r)||))$ \{Use the inferred task identity to select task prompt\}
18:             **end if**
19:             $S(C, Q) \leftarrow$ indexes of $M'$ meta prompt keys that are closest to $q$
20:             $P_m(C, Q) \leftarrow \{P_{m_j}\}_{j \in S(C, Q)}$
21:             $P(C, Q) \leftarrow [g_θ; P_f(F_j); P_t(C, Q); P_m(C, Q)]$
22:             Calculate per sample loss $L_{QA}$ on $g_θ$ and $P(C, Q)$ by Eq.6
23:             Obtain negative sample $(C_n, Q_n)$ from $M$ by Eq.2
24:             Calculate per sample loss $L_{t}$ on $k_t(T_i)$ by Eq.1
25:             Calculate per sample loss $L_m$ on $\{k_m\}_{s_j \in S(C, Q)}$ by Eq.4
26:             if $(C, Q, A) \in M$ **then**
27:                 Calculate per sample loss $L'_m$ on $\{k_m\}_{s_j \in S(C, Q)}$ by Eq.4
28:             **end if**
29:         **end for**
30:     **end for**
31:     Update $g_θ$ and prompts with accumulated $L_{QA}$
32:     Update task prompt keys $\{k_t(T_i)\}_{i=1}^N$ with accumulated $L_t$
33:     Update meta prompt keys $\{k_m\}_{i=1}^M$ with accumulated $L_m$ and $L'_m$
34: **end for**
35: Update $M$ with $\{(C_j, Q_j, A_j)\}_{j=1}^n$ according to details in Appendix B
36: **end for**
Open Domain Incremental Lifelong Learning for QA with Hierarchical Prompts

Written by AAAI Press Staff†

AAAI Style Contributions by Pater Patel Schneider, Sunil Issar,
J. Scott Penberthy, George Ferguson, Hans Guesgen, Francisco Cruz†, Marc Pujol-Gonzalez‡

†Association for the Advancement of Artificial Intelligence
1900 Embarcadero Road, Suite 101
Palo Alto, California 94303-3310 USA
publications23@aaai.org

Implementation Details

All the experiments use T5-base as model backbone. We create 5 learning curriculums by randomly permuting 8 seen tasks with random seeds of \{42, 43, 44, 45, 46\}, and report average score of each method. To implement query function \( h \), we adopt an additional T5-base encoder and keep its parameters frozen. Meanwhile, the score function \( \text{sim} \) is implemented cosine similarity, which ranks the match between sample query and prompt keys.

During training and testing, we adopt 4 V100 GPUs. For training, AdamW is employed as the optimizer. We tried epoch number of \{3, 4, 5, 6, 7, 8\} and learning rate of \{1e−5, 5e−5, 1e−4, 5e−4, 1e−3\}. All hyper-parameters are tuned according to averaged score on validation dataset of NewsQA, NarQA and RACE. We set batch size as 8 per GPU, gradient accumulation steps as 2. We set learning rate as 1e−4 and training epoch as 5. For meta prompt key training, we adopt two loss margins \( \eta = 0.85 \) and \( \gamma = 0.7 \). For scheduled sampling in task prompt selection, we choose \( \epsilon = 0 \), \( \alpha = 0.1 \) and \( \beta = 3e−4 \). We have \( N = 30 \) meta prompts as a pool, and for each instance we choose \( M = 5 \) meta prompts. When clustering memory samples to generate prototypes, we set \( R = 5 \). Hence, there are \( 5t \) prototypes after learning \( t \) tasks. For “task0” prompts, we set \( n_0 = 5000 \), training “task0” prompt \( \mathcal{P}_{task} \) with 5000 samples each task of format \( f \). The lengths of prompt values \( \mathcal{V}_{gen}, \mathcal{V}_{form}, \mathcal{V}_{task}, \mathcal{V}_{meta} \) are set as 20, 40, 40 and 20.

When training unseen task prompt \( \hat{P}_t(F_j) \) in format \( F_j \), 5000 samples from each task in \( F_j \) are accumulated to train extra unseen task prompt.

To enhance the diversity of samples stored in memory buffer, we select memory samples from each task with meta prompt keys. After the training process of each task, we have meta prompt keys already updated. For limited memory size \( B \) per task, we select top-\( \lfloor B/M \rfloor \) samples for each meta prompt key, using similarity score function \( \text{sim} \). After accumulating \( M\lfloor B/M \rfloor \) memory samples for all prompt keys, we rank these samples with their similarity to their corresponding meta prompt keys, and choose top-\( B \) samples as task memory. Yet, memory buffer is optional in Diana. Without memory buffer, the training of meta prompt keys changes from Equation 4 to Equation 3. Meanwhile, without memorized negative query samples from memory buffer, the training of task prompt keys for sample \( x \) of task \( t \) changes from Equation 1 to a simpler form:

\[
\mathcal{L}_{task}^{\Omega} = 1 - \text{sim}(\text{enc}(x), \mathcal{K}_t) \tag{1}
\]

Analysis of scheduled sampling

At training time, only using task prompt with ground truth task identity results in a discrepancy between training and testing, as is implemented by Diana-SS. An extreme solution is to only use task identity predicted, as is implemented by variant Diana-GT. However, as shown in Figure 1, task identity classification accuracy is extremely low for earlier iterations, which hinders task prompts from sharing task-specific knowledge. We make a compromise between two extremes, adopt a linear scheduled sampling when choosing task prompts. When closing the gap between training and testing, it still remains a larger probability of using true task prompt. As shown in ablation study, Diana achieves a better performance than two extremes Diana-SS and Diana-GT.

Figure 1: Change in probability of using corresponding task prompt (SIQA) at training time for different strategies. We train SIQA as the last task.
| Methods    | Param. | Mem. | Training Time. | Testing Time. |
|------------|--------|------|----------------|---------------|
| Lower Bound | 222.90M | ∅   | 751            | 523           |
| EWC        | 222.90M | ∅   | 1274           | 596           |
| FLCB       | 222.90M | ∅   | 813            | 591           |
| AdapterCL  | 262.25M | ∅   | 1000           | 5852          |
| L2P        | 223.39M | ∅   | 1382           | 1013          |
| DualPrompt | 223.17M | ∅   | 1274           | 1147          |
| ER         | 222.90M | 50   | 799            | 541           |
| DER++      | 222.90M | 50   | 930            | 604           |
| AFPER      | 223.43M | 50   | 1299           | 630           |
| ProQA      | 223.43M | ∅   | 1172           | 863           |
| Diana      | 223.84M | 50   | 1432           | 1108          |

Table 1: Comparison of computational cost. Param shows number of parameters (Million), and Mem the episodic memory size need per task. Training Time shows average training time (second) for each epoch on the SQuAD dataset, and Testing Time shows time (second) for model to be evaluated on all 11 datasets.

**Analysis of computational cost**

Table 1 shows total parameters, computation and memory overhead introduced by different methods. As we can see, our method improves lifelong learning for QA without introducing too much computation overhead.