ProQA: **Structural Prompt-based Pre-training for Unified Question Answering**

Anonymous ACL submission

Abstract

Question Answering (QA) is a longstanding challenge in natural language processing. Existing QA works mostly focus on specific question types, knowledge domains, or reasoning skills. The specialty in QA research hinders systems from modeling commonalities between tasks and generalization for wider applications. To address this issue, we present ProQA, a unified QA paradigm that solves various tasks through a single model. ProQA takes a unified *structural prompt* as the bridge and improves the QA-centric ability by *structural prompt-based pre-training*. Through a structurally designed prompt-based input schema, ProQA concurrently models the knowledge generalization for all QA tasks while keeping the knowledge customization for every specific QA task. Furthermore, ProQA is pre-trained with structural prompt-formatted large-scale synthesized corpus, which empowers the model with the commonly-required QA ability. Experimental results on 11 QA benchmarks demonstrate that ProQA consistently boosts performance on both full data fine-tuning, few-shot learning, and zero-shot testing scenarios. Furthermore, ProQA exhibits strong ability in both continual learning and transfer learning by taking the advantages of the structural prompt.[1]

1 Introduction

Question Answering has long been an inspirational challenge in NLP research, and is viewed as the next-generation search engine and an essential tool for human beings to obtain knowledge (Etzioni, 2011). Many distinct datasets (Rajpurkar et al., 2016; Lai et al., 2017; Kwiatkowski et al., 2019) have been proposed along with the research trend on QA, involving very diverse question types (e.g., *extractive QA, abstractive QA, multiple-choice QA*), domains (e.g., *finance, daily events*), and answer types (e.g., *free-formed text, selected option*). The majority of previous works focus on tasks with specific question types (Lai et al., 2017; Yang et al., 2018) or specific domains (Trischler et al., 2017; Kwiatkowski et al., 2019). Recent research on large pre-trained language models (Brown et al., 2020; Bommasani et al., 2021) indicates that there may be tight connections among various tasks, which sheds light on a unified paradigm that can be potentially applied to solve various QA tasks to model their commonality.

This observation motivates us to develop a unified QA model, which can model both the commonly-required QA ability and the difference between various QA tasks within a same paradigm. To achieve this goal, there are several key challenges needed to be addressed: (1) How to model commonalities and enhance transferability among different QA tasks in various domains/formats while reducing the conflict between them? (2) How to construct large-scale QA corpus as the high-quality QA-centric data is scarce for pre-training?

In light of this, we conceive ProQA, a unified QA paradigm, which builds up a general model to solve different QA tasks utilizing a **structural prompt** and improves commonly-required QA ability via **structural prompt-based pre-training**.

Firstly, to model the commonalities and distinguish task differences, we adopt a **structural prompt** to organize the inputs with a unified structurally designed input schema. As illustrated in Fig. 1, given the complex components (e.g., *"Domain"*, *"Format"*, *"Task"*, *"Question"*, *"Passage"*) as inputs, ProQA divides components into multiple key-value pairs, in which a specific component like *"Question"* denotes a key, and the specific instance in this component is taken as the value. In this way, the model can discriminate different input components by key indicators and model the speciality of each task via task-specific values (learnable prompts).

Secondly, to alleviate data sparsity problem and

[1] We will release the code, pre-training corpus and pre-trained models to facilitate future research along this line.

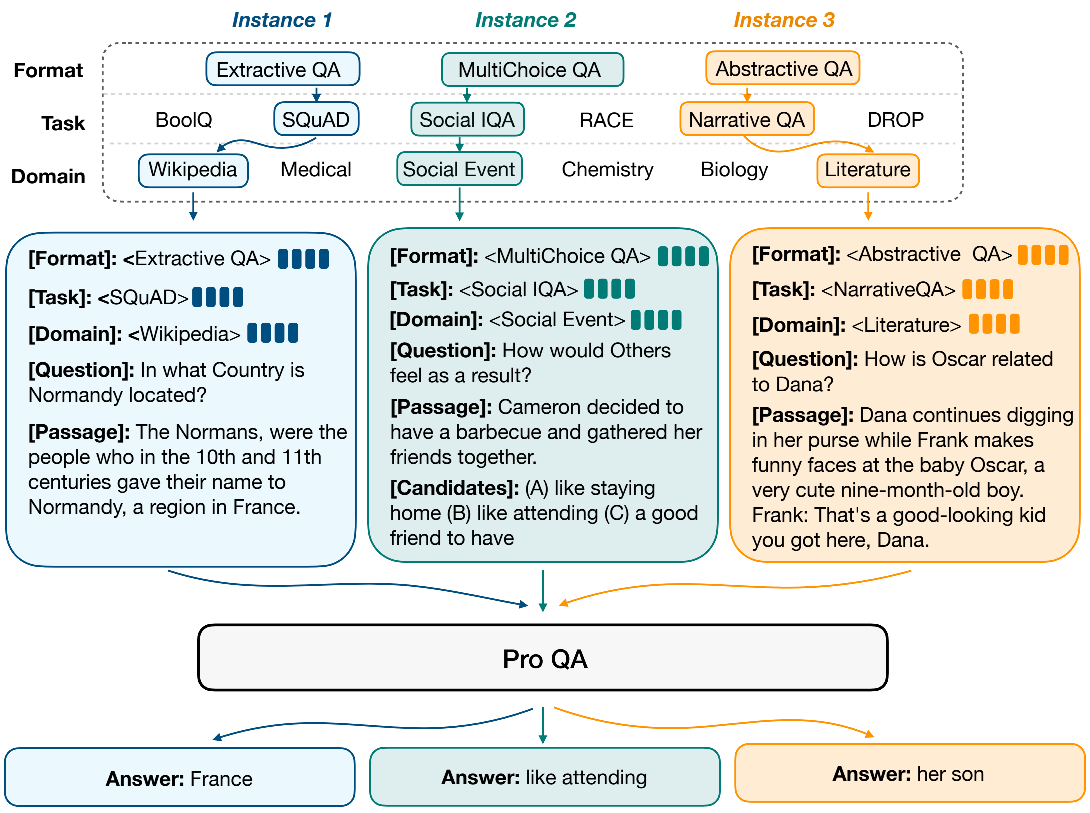
Instance 1
Instance 2
Instance 3
Format
Task
Domain
Extractive QA
MultiChoice QA
Abstractive QA
BoolQ
SQuAD
Social IQA
RACE
Narrative QA
DROP
Wikipedia
Medical
Social Event
Chemistry
Biology
Literature
[Format]: <Extractive QA>
[Task]: <SQuAD>
[Domain]: <Wikipedia>
[Question]: In what Country is Normandy located?
[Passage]: The Normans, were the people who in the 10th and 11th centuries gave their name to Normandy, a region in France.
[Format]: <MultiChoice QA>
[Task]: <Social IQA>
[Domain]: <Social Event>
[Question]: How would Others feel as a result?
[Passage]: Cameron decided to have a barbecue and gathered her friends together.
[Candidates]: (A) like staying home (B) like attending (C) a good friend to have
[Format]: <Abstractive QA>
[Task]: <NarrativeQA>
[Domain]: <Literature>
[Question]: How is Oscar related to Dana?
[Passage]: Dana continues digging in her purse while Frank makes funny faces at the baby Oscar, a very cute nine-month-old boy. Frank: That's a good-looking kid you got here, Dana.
Pro QA
Answer: France
Answer: like attending
Answer: her son

Figure 1: Approach overview of ProQA. Each box represents a specific instance formulated with the **structural prompt**, and ProQA is pre-trained with **structural prompt-based pre-training**. [ ] indicates *special key indicator*, < > denotes *hard prompt*, and colored squares denote *continuous learnable soft prompts*.

empower the model with transferability to the adaptation of new tasks, we conduct **structural prompt-based pre-training**. We first build a large-scale synthetic QA corpus automatically from Wikipedia, utilizing only a few seed datasets as the prior supervisions for pre-training corpus construction and finally covering primary QA formats. Then we format the pre-training data with the structural prompt, and teach the model to learn the general purpose QA-centric ability and the functionality of each component in the structural prompt via pre-training.

We evaluate the effectiveness of ProQA on 11 downstream QA benchmarks, and the results show that our system achieves consistent performance boost in full data fine-tuning, few-shot learning, and zero-shot learning settings. Experiments demonstrate that ProQA can better mitigate the catastrophic forgetting issue during continual learning by restoring the task-specific soft prompts residing in the structural prompt. Further analyses illustrate that our model has better transferability as it can be more quickly adapted to a newly involved task. Ablation studies verify the effectiveness of both the soft prompt and prompt-based pre-training.

The contributions are summarized as follows:

- We propose ProQA, a unified QA framework for solving various tasks within a single paradigm, taking an extensible and learnable structural prompt as the bridge.
- We enhance general QA-centric capabilities via structural prompt-based pre-training.
- Comprehensive experiments show that our model consistently improves the performance on 11 QA tasks especially in low-resource settings and exhibits better effectiveness in continual learning and few-shot transfer learning.

2 Related Work

Unifying QA formats. Despite vast diversity of current QA tasks in question type, answer type, answer source, and data domain (Zeng et al., 2020), there have been efforts in exploring a *unified* format for various QA tasks. Some pioneered to demonstrate the generalization and transferability among different QA tasks (Talmor and Berant, 2019; Dua et al., 2019a; Fisch et al., 2019). Another line of works investigate multi-task learning for QA (McCann et al., 2018; Shen et al., 2019; Deng et al., 2019) by jointly training a single encoder to promote knowledge sharing. However, these methods typically require deploying distinct prediction heads for different tasks, which lead to poor scalability and flexibility when confronted with emerging QA tasks of new types.

To this end, inspired by the success of casting multiple tasks into the same text-to-text format (Lewis et al., 2020; Raffel et al., 2020), researchers propose to learn a single model to unify various QA formats, alleviating the labor of task-specific designs (Khashabi et al., 2020b; Tafjord and Clark, 2021). However, these models (1) do not explicitly model the task or component characteristics, thus failing to properly disentangle the difference among QA tasks; and (2) overly rely on supervised data from specific tasks, which may not be available under data-scarce scenarios.

QA-centric pre-training. Numerous efforts have been spent on improving PLMs' reasoning abilities with an intermediate pre-training stage before fine-tuning on target QA tasks, including (1) language modeling adaptation with salient span masking, which trains PLMs to recover randomly chosen (Guu et al., 2020; Wang et al., 2021) or machine-generated (Kang et al., 2020) masked named entities in the raw corpus; (2) training data augmentation with synthetic question-answer-context triples, such as generating (a) pseudo questions through adversarial training (Hosking and Riedel, 2019), knowledge bases (Hu et al., 2021) or machine translation (Lewis et al., 2019), (b) pseudo answers exploiting recurring spans (Ram et al., 2021) or rules based on heuristics (Bian et al., 2021) and (c) pseudo contexts via information retrieval (Glass et al., 2019). Nevertheless, these works largely target at improving a certain reasoning ability for PLMs, and thus cannot be easily generalized to other QA tasks.

Prompts for PLMs. To effectively stimulate the knowledge acquired through pre-training, prompt-oriented fine-tuning is receiving increasing attention (Liu et al., 2021; Ding et al., 2021), which re-formulates the objective of downstream tasks similar to that of pre-training by inserting manually designed (Schick and Schütze, 2021a,b) or automatically searched (Jiang et al., 2020; Shin et al., 2020) hard *prompt tokens* into the input text. Considering that discrete prompts may not be an optimal solution in the continuous embedding space, recent works (Li and Liang, 2021; Hambardzumyan et al., 2021) proposed tunable soft prompts. It achieves satisfying performance especially when the model size grows extremely large (Lester et al., 2021). Compared with the cumbersome parameters in PLMs, soft prompts are lightweight and pluggable, which paves the way for our goal of flexible adaptation to a new QA task.

[Format]: <Format Prompt**>** ▮▮▮▮

[Task]: <Task Prompt**>** ▮▮▮▮

[Domain]: <Domain Prompt**>** ▮▮▮▮

[Question]: Question Text

[Passage]: Passage Text

[Key$_i$]: Value$_i$

…

[Key$_k$]: Value$_k$

Figure 2: An illustration of the structural prompt. [ ] indicates special key indicator, < > denotes hard prompt, and grey squares indicate continuous soft prompts.

3 ProQA

In this section, we detailedly describe the whole framework of ProQA for general purpose QA, which solves various QA tasks within the same paradigm.

3.1 Overview

As shown in Fig. 2, we first organize the inputs of various QA tasks with a unified **structural prompt** (§ 3.2), and adopt a unified text-to-text model for question answering. Then, to enhance the model in learning the general QA-centric ability and the semantics of the structural prompt, we conduct **structural prompt-based pre-training** with synthetic pre-training corpus formatted with the structural prompt (§ 3.3).

Inspired by Khashabi et al. (2020b) and T5 (Raffel et al., 2020), we solve all downstream QA tasks with a unified text-to-text model. In this work, we mainly adopt T5 as the model backbone. Taking the structural prompt-based model input, the unified model generates the answer of the question.

3.2 Structural Prompt

Here we detailedly illustrate the design of the structural prompt and its formatted input to the model.

Definition. We organize complex QA task inputs with the structural prompt. As shown in Fig. 2, the structural prompt consists of multiple $\{key : value\}$ pairs, where the key represents a specific component[2] (e.g., "*Task*", "*Format*", "*Question*", etc.), and the $value$ has two possible types: (1) textual content (e.g., *question*, *passage*, *options*) of the data instance; (2) task attributes (e.g., *format, domain*) represented as the combination of a discrete *hard prompt* and continuous *soft prompts*. The hard prompt is a predefined discrete description (we adopt a special token here), and the soft prompts are lightweight learnable and pluggable continuous embeddings that are proven to be parameter-effective in task adaptation (Lester et al., 2021). The structural prompt-formatted examples are illustrated in Fig. 1. In the case of the SQuAD dataset, "⟨Format Prompt⟩", "⟨Task Prompt⟩", "⟨Domain Prompt⟩" will be "⟨Extractive QA⟩", "⟨SQuAD⟩", "⟨Wikipedia⟩", respectively.

To enhance the model in discriminating the functional difference between components, we adopt a **special key indicator** with learnable representation to represent each key. Furthermore, to model the difference between several tasks/domains/formats, we also adopt learnable and storable **specific soft prompts** as the $value$ to represent their customized characteristics, which makes the model more flexible for task adaptation.

As a result, the structural prompt can empower the model in the following aspects: (1) modeling knowledge generalization of various tasks utilizing a unified input schema; (2) discriminating different components with the special key indicator; (3) customizing the speciality of each task/format/domain with learnable and storable soft prompts as the $value$ under corresponding keys.

Input Representation. Specifically, given a structural prompt-formatted instance, we describe the specific representation of the model input. We firstly translate k^{th} key to a *key indicator* D_k (a special token), which is attached by the tokens V_k of the specific value to form a token sequence. It is further represented as $\boldsymbol{E}_k = Embedding([D_k; V_k])$. The representation of D_k is initialized and updated during training. Since we use soft prompts $\boldsymbol{P}_{\text{task}}/\boldsymbol{P}_{\text{format}}/\boldsymbol{P}_{\text{domain}}$ as the value of the corresponding key and they are commonly required for all the tasks, we prepend them to the input for convenience and concatenate all the $\boldsymbol{E}_k$ to form the final model input $\boldsymbol{X}$:

$$\boldsymbol{X} = [\boldsymbol{P}_{\text{domain}}; \boldsymbol{P}_{\text{format}}; \boldsymbol{P}_{\text{task}}; \boldsymbol{E}_1; ...; \boldsymbol{E}_k] \quad (1)$$

It is also worth noting that the representations $\boldsymbol{D}$ of key indicators and the soft prompts $\boldsymbol{P}$ are jointly trained with the main model parameters during pre-training for learning the semantics of the structural prompt. Moreover, after being tuned by various tasks, the soft prompts $\boldsymbol{P}$ can be stored to record the customized task-specific characteristics.

3.3 Structural Prompt-based Pre-training

In this part, we introduce how we conduct structural prompt-based pre-training to help the model in learning commonly-required QA ability and the semantics of the structural prompt during pre-training to facilitate the adaption of the structural prompt to downstream tasks.

Task Formulation. Along with the structural prompt-based paradigm, we manifest various exemplary QA format types (i.e., *Extractive QA*, *Abstractive QA*, *Multiple-choice QA* and *Yes/No QA*) for pre-training to inject the general QA-centric ability. Given the multi-format QA pre-training corpus, we transform all QA formats according to the proposed structural prompt, which enables joint pre-training while keeping the differences among various formats. Taking a structural prompt-formatted instance as the input and a free-form answer as the output, the task is further tailored to a QA task with the encoder-decoder model.

Pre-training Corpus Construction. When we prepare the QA pre-training corpus, data sparsity problem is extremely severe because (1) it is impractical and laborious to obtain a large-scale high-quality annotated data for pre-training and (2) it is hard to generate QA-centric self-supervised data using rule-based methods (e.g., token masking or sentence reordering). In this work, inspired by Lewis et al. (2021), we adopt a **generation-filtering based corpus construction method** to synthesize a large-scale pre-training corpus, based on a large-scale unlabeled Wikipedia corpus with almost 6 million passages.

[2]It is worth noting that "*Format*" key denotes the format type (e.g., "*MultiChoice QA*") of the task while the "*Task*" key denotes a specific dataset (e.g., "*SQuAD*").

Typically, the general generation-filtering process consists of the following components:

1. A QA-pair generation model $g_{qa}(q, a|c)$: Given a passage c as input, $g_{qa}(q, a|c)$ generates q `[SEP]` a as the output sequence including a pair of question q and its answer a.
2. A filtering QA language model $f(a|q, c)$ for filtering the generated QA-pairs to ensure the quality and consistency of the question and the answer. $f(a|q, c)$ is a conditional-probability-based approach to filter out QA pairs softly. It scores a QA pair (q, a) with the likelihood of the answer a conditioned on the passage c and question q. The QA-pairs with scores higher than a threshold will be kept for pre-training.

We adopt the same text-to-text pre-trained model T5 described in § 3.1 as the model backbone of both the generation and filtering model.

To ensure the reliability of the generation and filtering models, we inevitably select a few seed datasets (typically one for each QA format type) as the prior supervisions to train these models. It is worth mentioning that, we avoid using more supervised data for corpus construction, because we expect the whole paradigm to have better expandability. In other words, if we want to extend the paradigm for a newly-involved QA format type but with limited supervised data, we can utilize these data to automatically create a synthetic large-scale pre-training corpus.

More specifically, the construction method has little variance for different formats according to their input components. For ***Extractive QA*** and ***Abstractive QA***, we adopt the aforementioned general method to synthesize QA-pairs. We also tried to first extract answers using rule-based method (extracted named-entities or key phrases), and only generate questions. We empirically find that this method performs much worse as it involves simple bias of the rule-based method. As the inputs for ***Multiple-Choice QA*** involve a new component "*Candidate Answers*", we adopt a distractor (negative options) generation model $g_{neg}(o|c, q, a)$ to generate three negative options o. For ***Yes/No QA***, we simply generate questions by taking *True/False* as the corresponding answers. Further details are described in Appendix A.

Format	Dataset	#Train	#Dev	QA Skills
Extractive QA	SQuAD*	87k	10k	Word Matching
	Quoref	22k	2k	Coreference Reasoning
	NewsQA	76k	4k	Word Matching
Abstractive QA	NarQA*	65k	21k	Story Understanding
	DROP	77k	9k	Discrete Reasoning
	NQOpen	79k	3.6k	Multi-passage Understanding
MultiChoice QA	RACE*	87k	4k	Multi-sentence Reasoning
	DREAM	6k	2k	Dialog Reasoning
	MCTest	1.4k	320	Multi-sentence Reasoning
	OBQA	4k	501	Common Knowledge
	SIQA	33.4k	2.2k	Commonsense Reasoning

Table 1: Dataset statistics and required language understanding skills. Datasets with * denote seed datasets for preparing pretraining data.

4 Experimental Setup

4.1 Datasets and Evaluation Metrics

We consider three formats of QA datasets in our experiments: Extractive QA, Abstractive QA and Multiple-Choice QA[3]. For each QA format, we select one *seed* dataset for preparing the large-scale pre-training data. The seed dataset is used to train the question-answer generation and filtering models in the process of pre-training corpus construction. In total, the experiments are conducted on 11 QA datasets with three different formats and various language understanding abilities. An overview of datasets used in the experiments and their required QA skills are summarized in Table 1.

Extractive QA. We take SQuAD 1.1 (Rajpurkar et al., 2016) as the seed dataset for extractive style QA. In addition, we consider NewsQA (Trischler et al., 2017) and Quoref (Dasigi et al., 2019) to evaluate the generalization ability of models. The EM (Exact Match) score between the extracted span and the gold answer span is used as the evaluation metric for extractive QA.

Abstractive QA. Narrative QA (NarQA) (Kočiský et al., 2018) is taken as the seed dataset for Abstractive QA. DROP (Dua et al., 2019b) and the open-domain version of NaturalQuestions (NQOpen) (Kwiatkowski et al., 2019) are also considered. Passages for each question in NQOpen are retrieved by the dense passage retriever (Karpukhin et al., 2020) and are concatenated into a sequence. We use ROUGE-L (Lin, 2004) metric for NarQA and F1 score for DROP and NQOpen.

[3]We also include Yes/No QA in our pilot study. We do not consider it in our main experiments because datasets in this format are extremely rare. Results on this QA formats are shown in Appendix B.

Setting	Dataset	ExtractiveQA			AbstractiveQA			MultiChoiceQA					Avg
		SQuAD	Quoref	NewsQA	NarQA	DROP	NQOpen	RACE	DREAM	MCTest	OBQA	SIQA	
Full-Data	T5	83.4	64.9	45.2	49.3	45.0	42.3	67.9	54.8	44.4	49.6	64.1	55.5
	UnifiedQA	84.4	74.8	45.3	49.6	45.1	42.5	71.6	67.6	83.1	57.6	64.9	62.4
	ProQA (qapair)	84.9	76.6	**50.8**	49.8	**55.0**	43.2	**73.6**	72.9	85.0	**61.6**	**67.5**	64.9
	ProQA (paq)	**85.3**	**76.8**	50.4	**50.1**	52.5	**43.9**	73.2	**73.3**	**85.9**	61.4	67.2	**65.0**
Few-Shot	T5	6.7	14.6	20.5	3.4	5.8	11.9	26.2	34.7	38.1	29.0	32.4	20.3
	UnifiedQA	82.0	38.2	34.2	49.1	22.2	31.6	53.0	57.4	73.8	41.2	42.8	48.1
	ProQA (qapair)	82.9	44.2	41.1	49.1	24.9	33.3	63.4	64.5	82.5	**46.2**	49.1	52.8
	ProQA (paq)	**84.4**	**52.2**	**42.1**	**49.2**	**27.1**	**36.0**	**66.5**	**66.0**	**84.1**	44.8	**49.4**	**54.7**
Zero-Shot	T5	0.0	0.0	0.0	3.5	2.0	1.5	24.1	34.2	27.5	21.9	33.2	13.5
	UnifiedQA	80.7	27.9	31.4	48.3	18.0	30.9	53.0	57.0	73.4	35.9	40.3	45.2
	ProQA (qapair)	80.4	30.5	30.7	48.1	17.0	33.0	62.6	64.3	**81.3**	36.0	**47.2**	48.3
	ProQA (paq)	**81.3**	**42.1**	**31.8**	**48.4**	**19.7**	**36.0**	**65.9**	**65.2**	**81.3**	**38.6**	46.7	**50.6**

Table 2: Main results on 11 downstream QA datasets under full-data fine-tuning, few-show learning, and zero-shot learning settings. Since the supervisions of seeds datasets are used in the pre-training corpus construction which may introduce bias in few-shot and zero-shot settings, results on these corresponding entries are underlined.

Multiple-Choice QA. For multiple choice QA, the following datasets are considered: RACE (Lai et al., 2017) (seed dataset), DREAM (Sun et al., 2019), MCTest (Richardson et al., 2013), OpenBookQA (OBQA) (Mihaylov et al., 2018), Social IQa (SIQA) (Sap et al., 2019). OBQA does not have contexts (reading comprehension passages). The context for DREAM is in the dialogue style and we concatenate them into a sequence as the passage input. We select the option with the highest textual similarity with the generated answer as the final answer. We compute the accuracy of the correct options for all multiple choice QA datasets.

4.2 Approaches

T5 (Raffel et al., 2020) is a unified text-to-text pre-training framework that covers all text-based language problems. We use `google/t5-v1_1-base` from HuggingFace Transformers (Wolf et al., 2020) that is only pre-trained on C4 excluding any supervised training dataset (e.g., QA datasets).

UnifiedQA (Khashabi et al., 2020b) crosses the format boundaries of different QA tasks by formulating them into text-to-text tasks under T5. It directly concatenates all inputs via `\n` into a sequence and feeds it into T5 for predicting the answer. We train our own UnifiedQA model on the combination of three aforementioned seed datasets, namely SQuAD, NarQA, and RACE.

ProQA is our proposed structural prompt-based pre-training approach. ProQA is pre-trained jointly on three formats of pre-training corpus: Extractive QA, Abstractive QA, and Multiple-Choice QA. This approach using corpus prepared from QA-pair generation-filtering model described in § 3.3 is named as ProQA (qapair). Additionally, we leverage the off-the-shelf large-scale QA pairs from Probably-Asked Questions/PAQ (Lewis et al., 2021), and replace our extractive QA pre-training corpus by a subset of PAQ (abstractive QA and multiple-choice QA corpus remains unchanged). PAQ provides a refined pipeline that introduces learned models on every step of QA pair generation, i.e., passage selection, answer identification, question generation, and filtering. We name this variant as ProQA (paq).

For every downstream QA dataset, we start from the above pre-trained models and conduct experiments under full-data fine-tuning, few-shot learning, and zero-shot learning settings. For few-shot learning, we randomly sample 32 instances from the training set.

5 Results and Analyses

5.1 Main Results

Main results are shown in Table 2, and we have the following observations:

- QA-centric pre-trained models, namely UnifiedQA and ProQA, outperform T5 by a large margin on both seed datasets and non-seed datasets. This is because there is some transferable knowledge across different QA tasks. Once the model is pre-trained by any QA task, the learned knowledge can be generalized to any other datasets.
- ProQA demonstrates better knowledge customization ability than UnifiedQA – ProQA beats UnifiedQA by a large margin in few-shot and zero-shot settings. This is because (1) the

Methods	Task A→Task B									
	EX→EX	EX→AB	EX→MC	AB→EX	AB→AB	AB→MC	MC→EX	MC→AB	MC→MC	Avg
Task B Model	20.5%	26.2%	13.0%	8.9%	6.3%	6.5%	4.6%	4.9%	0.9%	9.9%
Task B Model (w/ Task A Prompt)	17.1%	10.6%	6.7%	3.1%	3.1%	2.2%	0.4%	0.7%	-0.5%	4.3%

Table 3: Continual learning results for averaged performance drops compared with the original task A results under different task learning orders (**lower is better**). Negative number means the performance improves compared with the original task A results. EX: Extractive QA; AB: Abstractive QA; MC: Multiple-chioce QA.

hard and soft prompts in the structural prompt enable better knowledge customization for every QA task, especially the "Task" key-value pair that is different for every QA task; (2) structural prompt-based pretraining empowers ProQA to adapt faster (§ 5.3) and better (Table 2) to these non-seed datasets.

- Comparing ProQA (qapair) and ProQA (paq), we find that ProQA (paq) performs better in most scenarios. Presumably, PAQ provides high quality pre-training corpus through its pipelined approach – there are in total four BERT-sized models to be prepared for generating PAQ corpus. Instead, our proposed QA pair generation approach is simple and can be applied to not only Extractive QA but also Abstractive QA and Multiple-choice QA in the pre-training corpus construction process.

5.2 Continual Learning via Soft Prompt

One benefit of introducing soft prompt in ProQA is that it can potentially mitigate the catastrophic forgetting issue when adapting to a new task. If ProQA is **sequentially fine-tuned on task A and task B under few-shot setting**, it can load task A soft prompt back when it is evaluated again on the task A. The plug-in flexibility of ProQA brings huge improvements compared with its counterpart that keeps the task B soft prompt.

We conduct continual learning by setting task A and B as different combinations among datasets with formats[4]: Extractive QA (EX), Abstractive QA (AB), and Multiple-choice QA (MC). Formally, we first adapt ProQA to task A by few-shot learning to obtain the model A: f_θ^A with performance s^A. Then we sequentially adapt f_θ^A to task B and receive task B model f_θ^{AB}. We evaluate performance of the model f_θ^{AB} on task A under two settings: (1) direct testing (task-B prompt) (2) first restoring the learned task-A prompt from f_θ^A to the model f_θ^{AB} and then testing. Performance of the two settings are denoted as s^{AB} and $s^{AB'}$, respectively. We evaluate the continual learning performance under these two settings with the **percentage of the performance drop** on the task A: "*Task B Model*"= $\frac{s^A - s^{AB}}{s^A}$, and "*Task B Model (w/ Task A Prompt)*" = $\frac{s^A - s^{AB'}}{s^A}$.

[4]Note that we consider two tasks in continual learning because we also want to directly investigate the task adaptation to-and-fro the same format (e.g., MC → MC) or different formats (e.g., AB → EX).

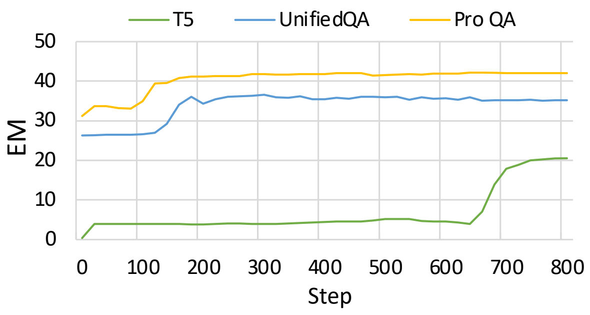

Figure 3: The few-shot learning curves of EM scores on the validation set of the NewsQA task.

As shown in Table 3, the catastrophic forgetting issue does exist when evaluating task A with task B model ("*Task B Model*") directly. The performance drops as large as 26.2% for EX→AB. However, restoring task A prompt brings huge improvements across all task combinations ("*Task B Model w/ Task A Prompt*"). It is surprising to see that restoring task A prompt could sometimes even improve task A performance (MC→MC = −0.5%). Presumably, sequential learning two tasks under the same question format (MC) makes the model learn the transferable knowledge while restoring task A prompt brings task-specific knowledge. Detailed experimental results on the 33 combinations of datasets can be found in Appendix C.

5.3 Convergence Analysis

We investigate the effectiveness of pre-training by compare the step-wise performance under few-shot learning setting. The learning curves of EM scores on the validation set of the NewsQA task is shown in Figure 3. Out of the three models, T5 convergences slowest because it does not have any QA-centric knowledge while our proposed ProQA

Model	NewsQA	NQOpen	DREAM
ProQA	42.1	36.0	66.0
w/o soft prompt	38.5	32.9	64.5
w/o pretraining	20.5	10.7	35.1
UnifiedQA + Pre-train Corpus	37.3	32.4	59.6

Table 4: Ablation study results on three non-seed datasets under different QA formats (extractive, abstractive, multiple-choice).

adapts fastest and best. Moreover, we find that UnifiedQA EM score rapidly saturates and eventually degrades slightly, suggesting that the model overfits under the few-shot setting. On the counterpart, our ProQA continues to improve and never degrades because the hard and soft prompt inside the structural prompt balance the knowledge generalization and knowledge customization well.

5.4 Ablation Study

An ablation study is conducted to unveil the effectiveness of every component in ProQA. We consider three variants of ProQA: (1) ProQA without the soft prompt in its structural prompt; (2) ProQA further without prompt-based pre-training. (3) UnifiedQA + Pre-train Corpus is the UnifiedQA model pre-trained on our prepared large-scale synthetic QA corpus. Few-shot learning results on three non-seed datasets under different QA formats are shown in Table 4. We find that removing the soft prompt from the model disables the task-specific knowledge learned during pre-training. Moreover, removing the prompt-based pretraining drastically hurts the performance as the equivalent model (T5 + hard structural prompt) does not have any QA knowledge. Finally, UnifiedQA + Pre-train Corpus could not compete with ProQA, showing that our proposed structural prompt earns better balance between knowledge generalization and knowledge customization than UnifiedQA.

6 Discussion

In this section, we discuss on how to extend the ProQA to a new task even with a new schema, and sheds light on potential future directions.

1) Task Adaptation with Structural Prompt: The design of structural prompt empowers ProQA with better expandability. In our main experiments, we adopt 3 format types and 11 QA tasks. In the future, we can adapt ProQA to more tasks, formats, domains, and new input schema. Intuitively, when being adapted to a new task with unseen format/domain, ProQA can initialize the specific soft prompts and learn the characteristic of the new domain/task through model training. Moreover, if we encounter a new input schema that involves new keys (e.g., "*extracted entities or commonsense knowledge*"), we can add a new key-value pair in the input schema and learns the functionality of the new key indicator through training.

2) Unified QA Systems: We think further studies on unified QA systems could target on a better pre-training schema for general purpose QA, or optimizing the modeling strategy for the structural prompt to process more complex input, or output formats (e.g., adding extracted entities or retrieved knowledge).

3) Unification with Structural Prompt: The application of the structural prompt is not limited only on the QA task. Intuitively, task inputs/outputs with various formats or components can also be organized with the structural prompt, like *dialog* or *aspect-based sentiment analysis*. In this way, we can integrate multiple tasks with carefully organized structural input, and improve the uniformity and expandability of the whole paradigm.

7 Conclusion

We introduce ProQA, a unified QA paradigm that adopts a single model for solving various QA tasks with the bridge of a structural prompt. Structural prompt simultaneously models the common ability required for various tasks and keeps the speciality of each task, through a structurally designed learnable input schema. We further conduct structural prompt-based pre-training, seeking to empower the model with general QA-centric ability and injects the semantic knowledge of the structural prompt into the pre-training model. Experimental results on 11 QA benchmarks demonstrate that ProQA can significantly boost performance on all settings. Further analyses show that our method can better mitigate the catastrophic forgetting issue during continual learning, and our method can be adapted to a newly involved task more quickly, by taking the advantages of the structural prompt. In the future, we hope our analysis could inspire more explorations on the unified QA methods, or the unification of distinct tasks with complex inputs modeling by the structural prompt. We also hope structural prompt can be further utilized into the unification of more tasks with complex inputs.

References

Ning Bian, Xianpei Han, Bo Chen, Hongyu Lin, Ben He, and Le Sun. 2021. Bridging the gap between language model and reading comprehension: Unsupervised mrc via self-supervision. *arXiv preprint arXiv:2107.08582*.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

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

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Yang Deng, Yuexiang Xie, Yaliang Li, Min Yang, Nan Du, Wei Fan, Kai Lei, and Ying Shen. 2019. Multi-task learning with multi-view attention for answer selection and knowledge base question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6318–6325.

Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Hai-Tao Zheng, and Maosong Sun. 2021. Openprompt: An open-source framework for prompt-learning. *arXiv preprint arXiv:2111.01998*.

Dheeru Dua, Ananth Gottumukkala, Alon Talmor, Sameer Singh, and Matt Gardner. 2019a. Orb: An open reading benchmark for comprehensive evaluation of machine reading comprehension. *arXiv preprint arXiv:1912.12598*.

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

Oren Etzioni. 2011. Search needs a shake-up. *Nature*, 476(7358):25–26.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. Mrqa 2019 shared task: Evaluating generalization in reading comprehension. *arXiv preprint arXiv:1910.09753*.

Michael Glass, Alfio Gliozzo, Rishav Chakravarti, Anthony Ferritto, Lin Pan, GP Bhargav, Dinesh Garg, and Avirup Sil. 2019. Span selection pre-training for question answering. *arXiv preprint arXiv:1909.04120*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: Word-level Adversarial ReProgramming. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4921–4933, Online. Association for Computational Linguistics.

Tom Hosking and Sebastian Riedel. 2019. Evaluating rewards for question generation models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2278–2283, Minneapolis, Minnesota. Association for Computational Linguistics.

Ziniu Hu, Yizhou Sun, and Kai-Wei Chang. 2021. Relation-guided pre-training for open-domain question answering. *arXiv preprint arXiv:2109.10346*.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Minki Kang, Moonsu Han, and Sung Ju Hwang. 2020. Neural mask generator: Learning to generate adaptive word maskings for language model adaptation. *arXiv preprint arXiv:2010.02705*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Daniel Khashabi, Tushar Khot, and Ashish Sabharwal. 2020a. More bang for your buck: Natural perturbation for robust question answering. In *Proceedings of* *the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 163–170, Online. Association for Computational Linguistics.

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

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

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

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

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick Lewis, Ludovic Denoyer, and Sebastian Riedel. 2019. Unsupervised question answering by cloze translation. *arXiv preprint arXiv:1906.04980*.

Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021. PAQ: 65 million probably-asked questions and what you can do with them. *Transactions of the Association for Computational Linguistics*, 9:1098–1115.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

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

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pretrain, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

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

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

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

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

Ori Ram, Yuval Kirstain, Jonathan Berant, Amir Globerson, and Omer Levy. 2021. Few-shot question answering by pretraining span selection. *arXiv preprint arXiv:2101.00438*.

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

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

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Tao Shen, Xiubo Geng, Tao Qin, Daya Guo, Duyu Tang, Nan Duan, Guodong Long, and Daxin Jiang. 2019. Multi-task learning for conversational question answering over a large-scale knowledge base. *arXiv preprint arXiv:1910.05069*.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

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

Oyvind Tafjord and Peter Clark. 2021. General-purpose question-answering with macaw. *arXiv preprint arXiv:2109.02593*.

Alon Talmor and Jonathan Berant. 2019. MultiQA: An empirical investigation of generalization and transfer in reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4911–4921, Florence, Italy. Association for Computational Linguistics.

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

Cunxiang Wang, Pai Liu, and Yue Zhang. 2021. Can generative pre-trained language models serve as knowledge bases for closed-book qa? *arXiv preprint arXiv:2106.01561*.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Changchang Zeng, Shaobo Li, Qin Li, Jie Hu, and Jianjun Hu. 2020. A survey on machine reading comprehension—tasks, evaluation metrics and benchmark datasets. *Applied Sciences*, 10(21):7640.

A Implementation Details

A.1 Corpus Preparation

In this part, we describe the details of corpus construction.

The current pre-training corpus contains almost 4 million pre-training instances formulated with the structural prompt, including 1 million Multiple-choice QA instances, 2 million Extractive QA instances, and 2 million Abstractive QA instances. When generating questions and answers, we take the *context* as the input, and the sequence "*question* `[SEP]` *answer*" as the output. In order to train the filtering model, we take the *context* and *question* as the inputs, and the *answer* as the output. During the inference process of QA-pairs filtering, we take the context and the generated question as the model input of the QA model and set generated answer as the label. Then we compute the final soft score with the cross-entropy loss between the label and the answer generated by the QA model, Next, we rerank all the generated QA-pairs according to the soft scores in an ascending order to select the most consistent QA-pairs as the pre-training instances.

Specifically, we employ AdamW as the optimizer for model training. We adopt *T5-Large* as the model backbone and the seed datasets as supervisions for training both the question-answer pairs generation, and the filtering QA model. We set learning rate as 1e-5, warmup step as 0, batch size as 2 per GPU, and training epochs as 10.

A.2 Details on Pre-training and Task Adaptation.

Pre-training. During pre-training, we jointly train the main model parameters with the representations of the *special key indicators* and the *task/format-specific soft prompts*.

Initially, we don't have any specific tasks during pre-training, so we take the three pre-training corpus (i.e., "*MultiChoiceQA, Extractive QA, and Abstractive QA*") as the three initial tasks, and randomly initialize the task and format specific soft prompts.

Specifically, we use *T5-Base* as the model backbone, and set learning rate as 1e-4, batch size as 8 per GPU and gradient accumulation steps as 10. We adopt 8 V100 GPUs for pre-training.

Fine-tuning. During fine-tuning, we need to initialize the task/format-specific soft prompts for a specific downstream task. If the task corresponds to a specific format participating in the pre-training stage, we use the corresponding soft prompts of this format type to initialize the soft prompts for the current tasks to transfer the learned knowledge. If the task corresponds to a new format, we can randomly initialize the task/format prompts.

Specifically, we use *T5-Base* as the model backbone, and set learning rate as 1e-4, batch size as 2 per GPU, gradient accumulation steps as 2, and training epochs as 5. We adopt 8 V100 GPUs for fine-tuning.

Few-shot Learning. We adopt a similar way to initialize the task-specific soft prompts for few-shot learning. We use the standard setting which utilizes 32 randomly selected instances for few-shot learning. Specifically, we adopt *T5-Base* as the model backbone, and set learning rate as 1e-5, batch size as 1 per GPU, gradient accumulation steps as 1, and training steps as 800 for few-shot learning.

Zero-shot Learning Since zero-shot learning does not involve training stage, we just need to initialize the task-specific prompt for inference. Therefore, we initialize the task-specific prompt with the pre-trained task prompts of its corresponding format type.

Setting	Dataset	BoolQ
Full-Data	T5	62.2
	Pro QA	80.6
Few-Shot	T5	0.0
	Pro QA	55.4
Zero-Shot	T5	0.0
	Pro QA	62.1

Table 5: Result on two Yes/No QA tasks under full-data fine-tuning, few-shot learning, and zero-shot learning settings.

B Results on Yes/No Pre-training

During our pilot study, we take the BoolQ (Clark et al., 2019) as the seed dataset to construct a large-scale pre-training corpus, and test the full-data, few-shot, zero-shot on top of the pre-trained ProQA. We also take the naturally-perturbed version of this dataset BoolQ-NP (Khashabi et al., 2020a) into account for evaluation. Results are shown in Table 5. We find that the ProQA significantly outperforms T5 baseline on all settings. Note that we take a strict evaluation towards the model's output.

Task A	Task A Model (Few-Shot Results)	Task B	Task B Model (Evaluation on Task A)	Task B Model (w/ Task A Prompt) (Evaluation on Task A)
EX/NewsQA (EM)	42.1	EX/Quoref (EM)	33.7	35.3
		AB/DROP (F1)	31.3	33.7
		AB/NQOpen (F1)	27.6	34.2
		MC/DREAM (Acc)	35.2	37.9
		MC/MCTest (Acc)	34.5	36.8
		MC/OBQA (Acc)	34.4	35.9
EX/Quoref (EM)	52.2	EX/NewsQA (EM)	41.1	42.8
		AB/DROP (F1)	43.0	51.4
		AB/NQOpen (F1)	38.0	51.0
		MC/DREAM (Acc)	47.2	51.3
		MC/MCTest (Acc)	47.8	52.0
		MC/OBQA (Acc)	48.5	51.6
AB/NQOpen (F1)	36.0	EX/NewsQA (EM)	32.8	33.9
		EX/Quoref (EM)	36.2	35.4
		AB/DROP (F1)	34.3	35.8
AB/DROP (F1)	27.1	EX/NewsQA (EM)	22.8	26.3
		EX/Quoref (EM)	24.0	26.6
		AB/NQOpen (F1)	25.0	25.6
		MC/DREAM (Acc)	25.1	26.2
		MC/MCTest (Acc)	25.4	26.7
		MC/OBQA (Acc)	25.6	26.7
MC/MCTest (Acc)	84.1	EX/NewsQA (EM)	81.6	83.1
		AB/DROP (F1)	82.8	83.8
		MC/DREAM (Acc)	83.2	83.8
		MC/OBQA (Acc)	82.2	82.5
MC/DREAM (Acc)	66.0	EX/NewsQA (EM)	64.0	65.4
		AB/DROP (F1)	63.5	65.7
		MC/MCTest (Acc)	65.5	65.8
		MC/OBQA (Acc)	65.2	65.5
MC/OBQA (Acc)	44.8	EX/NewsQA (EM)	41.4	45.2
		AB/DROP (F1)	40.6	44.2
		MC/MCTest (Acc)	44.8	45.8
		MC/DREAM (Acc)	44.2	46.8

Table 6: Full results on continual learning. For each task, we provide the task format (EX, AB, MC) and its evaluation metrics (EM, F1, Acc). EX: Extractive QA; AB: Abstractive QA; MC: Multiple-Choice QA.

In other words, if the output is not any format of "yes", "no", "true", "false", that prediction will be classified as wrong.

C Details on Continual Learning

Table 6 provides the full results for the continual learning experiment. The model is firstly trained on task A under few-shot setting, and then fine-tuned on task B. Afterwards, we evaluate the trained "Task B Model" and "Task B Model (w/ Task A Prompt)" on task A to test its continual learning capability. Detailed results on every Task A/Task B combination (33 reported in total) are shown in Table 6. Note that we consider two tasks in continual learning because we also want to investigate the task adaptation to-and-fro the same format (e.g., MC $\rightarrow$ MC) or different formats (e.g., AB $\rightarrow$ EX). The results shed light on how could we arrange the order of training on tasks to achieve the best overall performance when a bunch of tasks arrive.