KECP: Knowledge Enhanced Contrastive Prompting for Few-shot Extractive Question Answering

Jianing Wang[1]*, **Chengyu Wang**[2]*, **Minghui Qiu**[2], **Qiuhui Shi**[3], **Hongbin Wang**[3], **Jun Huang**[2], **Ming Gao**[1,4]†

[1] School of Data Science and Engineering, East China Normal University, Shanghai, China
[2] Alibaba Group, Hangzhou, China [3] Ant Group, Hangzhou, China
[4] KLATASDS-MOE, School of Statistics, East China Normal University, Shanghai, China
lygwjn@gmail.com, chengyu.wcy@alibaba-inc.com
minghui.qmh@alibaba-inc.com, qiuhui.sqh@antgroup.com
hongbin.whb@@antgroup.com, huangjun.hj@alibaba-inc.com
mgao@dase.ecnu.edu.cn

Abstract

Extractive Question Answering (EQA) is one of the most essential tasks in Machine Reading Comprehension (MRC), which can be solved by fine-tuning the span selecting heads of Pre-trained Language Models (PLMs). However, most existing approaches for MRC may perform poorly in the few-shot learning scenario. To solve this issue, we propose a novel framework named **K**nowledge **E**nhanced **C**ontrastive **P**rompt-tuning (*KECP*). Instead of adding pointer heads to PLMs, we introduce a seminal paradigm for EQA that transforms the task into a non-autoregressive Masked Language Modeling (MLM) generation problem. Simultaneously, rich semantics from the external knowledge base (KB) and the passage context support enhancing the query's representations. In addition, to boost the performance of PLMs, we jointly train the model by the MLM and contrastive learning objectives. Experiments on multiple benchmarks demonstrate that our method consistently outperforms state-of-the-art approaches in few-shot settings by a large margin. [1]

1 Introduction

Span-based Extractive Question Answering (EQA) is one of the most challenging tasks of Machine Reading Comprehension (MRC). A majority of recent approaches (Wang and Jiang, 2019; Yang et al., 2019; Dai et al., 2021) add pointer heads (Vinyals et al., 2015) to Pre-trained Language Models (PLMs) to predict the start and the end positions of the answer span (shown in Figure 1(a)). Yet, these conventional fine-tuning frameworks heavily depend on the time-consuming and labor-intensive process of data annotation. Additionally, there is a large gap between the pre-training objective of Masked Language Modeling (MLM) (i.e., predicting the distribution over the entire vocabularies) and the fine-tuning objective of span selection (i.e., predicting the distribution of positions), which hinders the transfer and adaptation of knowledge in PLMs to downstream MRC tasks (Brown et al., 2020). A straightforward approach is to integrate the span selection process into pre-training (Ram et al., 2021). However, it may cost a lot of computational resources during pre-training.

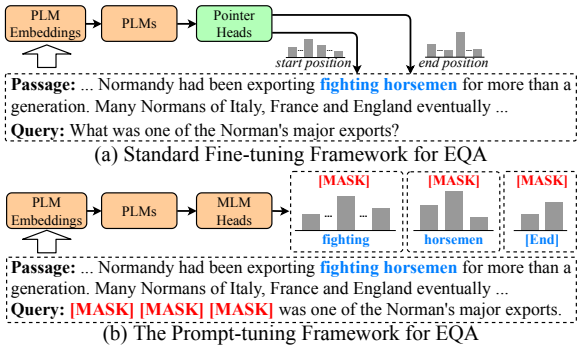

Figure 1: The comparison of the standard fine-tuning and prompt-tuning framework. The blocks in orange and green denote the modules of PLMs and newly initialized modules, respectively. (Best viewed in color.)

Recently, a branch of prompt-based fine-tuning paradigm (i.e. prompt-tuning) arises to transform the downstream tasks into the cloze-style problem (Schick and Schütze, 2021; Han et al., 2021; Li and Liang, 2021a; Gao et al., 2021; Liu et al., 2021a; Assem et al., 2021; Chada and Natarajan, 2021). To specify, task-specific prompt templates with `[MASK]` tokens are added to input texts (`[MASK]` denotes the masked language token in

* J. Wang and C. Wang contributed equally to this work.
† Corresponding author.
[1]All datasets are publicly available. Source codes will be released in EasyNLP (Wang et al., 2022). URL: https://github.com/alibaba/EasyNLP

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 3152 - 3163
December 7-11, 2022

PLMs). The results of the masked positions generated by the MLM head are used for the prediction[2]. By prompt-tuning, we can use few training samples to fast adapt the prior knowledge in PLMs to downstream tasks. A natural idea is that we can transform EQA into the MLM task by adding a series of masked language tokens. As shown in Figure 1(b), the query is transformed into a prompt template containing multiple `[MASK]` tokens, which can be directly used for the answer tokens prediction. However, we observe that two new issues for vanilla PLMs: 1) the MLM head, which is based on single-token non-autoregressive prediction, has a poor inference ability to understand the task paradigm of EQA ; 2) there are many confusing span texts in the passage have similar semantics to the correct answer, which can unavoidably make the model produce negative answers. Therefore, a natural question arises: *how to employ prompt-tuning over PLMs for EQA to achieve high performance in the few-shot learning setting?*

In this work, we introduce *KECP*, a novel **K**nowledge **E**nhanced **C**ontrastive **P**rompting framework for the EQA task. We view EQA as an MLM generation task that transform the query to a prompt with multiple masked language tokens. In order to improve the inference ability, for each given example, we inject related knowledge base (KB) embeddings into context embeddings of the PLM, and enrich the representations of selected tokens in the query prompt. To make PLMs better understand the span prediction task, we further propose a novel span-level contrastive learning objective to boost the PLM to distinguish the correct answer with the negatives with similar semantics. During the inference time, we implement a highly-efficient model-free prefix-tree decoder and generate answers by beam search. In the experiments, we evaluate our proposed framework over seven EQA benchmarks in the few-shot scenario. The results show that our method consistently outperforms state-of-the-art approaches by a large margin. Specifically, we achieve a 75.45% F1 value on SQuAD2.0 with only 16 training examples.

To sum up, we make the following contributions:

- We propose a novel *KECP* framework for few-shot EQA task based on prompt-tuning.
- In *KECP*, EQA is transformed into the MLM generation problem, which alleviates model over-fitting and bridges the gap between pre-training and fine-tuning. We further employ knowledge bases to enhance the token representations and design a novel contrastive learning task for better performance.
- Experiments show that *KECP* outperforms all the baselines in few-shot scenarios for EQA.

[2]For example, in sentiment analysis, a prompt template (e.g., "It was `[MASK]`.") is added to the review text (e.g., "This dish is very attractive."). We can obtain the result tokens of masked position for label prediction (e.g., "delicious" for the positive label and "unappetizing" for the negative label).

2 Related Work

In this section, we summarize the related work on EQA and prompt-tuning for PLMs.

2.1 Extractive Question Answering

EQA is one of the most challenging MRC tasks, which aims to find the correct answer span from a passage based on a query. A variety of benchmark tasks on EQA have been released and attracted great interest (Rajpurkar et al., 2016; Fisch et al., 2019; Rajpurkar et al., 2018; Lai et al., 2017; Trischler et al., 2017; Levy et al., 2017; Joshi et al., 2017; Chada and Natarajan, 2021). Early works utilize attention mechanism to capture rich interaction information between the passage and the query (Wang et al., 2017; Wang and Jiang, 2017). Recently, benefited from the powerful modeling abilities of PLMs, such as GPT (Brown et al., 2020), BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and SpanBERT (Joshi et al., 2020), etc., we have witnessed the qualitative improvement of MRC based on fine-tuning PLMs. However, this standard fine-tuning paradigm may cause over-fitting in the few-shot settings. To solve the problem, (Ram et al., 2021) propose Splinter for few-shot EQA by pre-training over the span selection task, but it costs a lot of time and computational resources to pre-train these PLMs. On the contrary, we leverage prompt-tuning for few-shot EQA without any additional pre-training steps.

2.2 Prompt-tuning for PLMs

Prompt-tuning is one of the flourishing research in the past two years. GPT-3 (Brown et al., 2020) enables few/zero-shot learning for various NLP tasks without fine-tuning, which relies on handcraft prompts and achieves outstanding performance. To facilitate automatic prompt construction, AutoPrompt (Shin et al., 2020) and LM-BFF (Gao et al., 2021) automatically generate discrete prompt tokens from texts. Recently, a series of methods learn

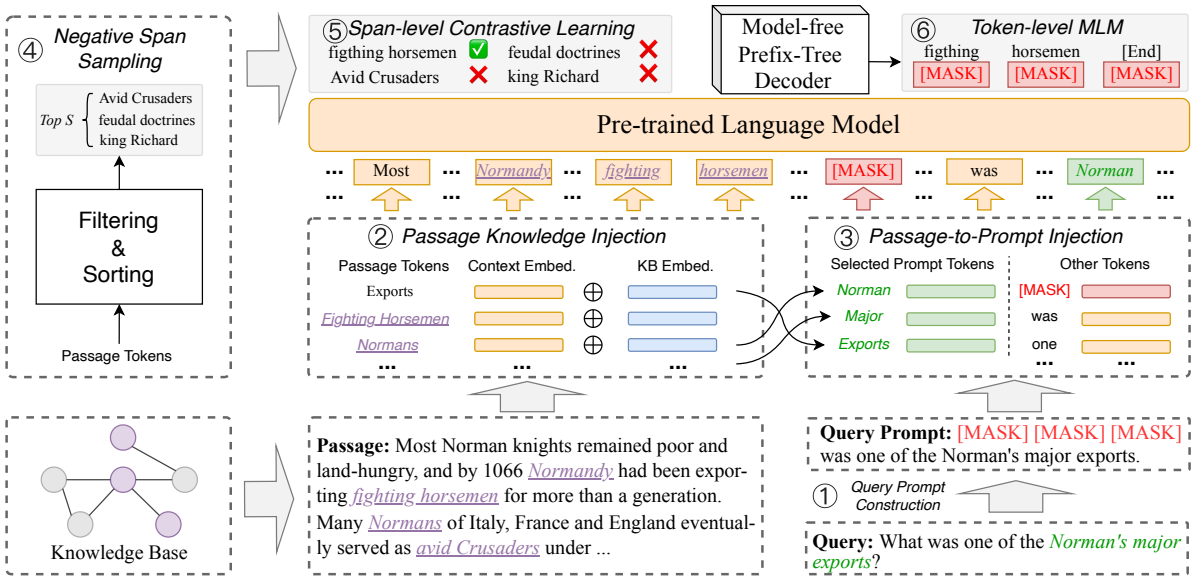

Figure 2: The *KECP* framework. Given a passage and a query, we first construct the query prompt by heuristic rules (①). Next, we capture the knowledge both from passage text and external KB to enhance the representations of selected prompt tokens (② ③). To improve the accuracy of answer prediction, we sample negative span texts with similar and confused semantics (④), and train the model with contrastive learning (⑤). During the inference stage, the answer span text can be generated by MLM and a model-free prefix-tree decoder (⑥). (Best viewed in color).

continuous prompt embeddings with differentiable parameters for natural language understanding and text generation task, such as Prefix-tuning (Li and Liang, 2021b), P-tuning V2 (Liu et al., 2021a), PTR (Han et al., 2021), and many other related works (Li and Liang, 2021b; Qin and Eisner, 2021). Different from previous work (Ram et al., 2021), we focus on prompt-based learning for the challenging low-resource EQA.

3 The *KECP* Framework

In this section, we formally present our task and the techniques of the *KECP* framework in detail. The overview of *KECP* is shown in Figure 2.

3.1 Task Overview

Given a passage $P = p_1, \cdots, p_n$ and the corresponding query $Q = q_1, \cdots, q_m$, the goal is to find a sub-string of the passage as the answer $Y = p_k, \cdots, p_l$, where n, m are the lengths of the passage, the query, respectively. p_i $(i = 1, \cdots, n)$ and q_j $(j = 1, \cdots, m)$ refer to the tokens in P and Q, respectively. k, l denotes the start and end position of the passage, $1 \leq k \leq l \leq n$. Rather than predict the start and the end positions of the answer span, we view the EQA task as a non-autoregressive MLM generation problem. In the following, we will provide the detailed techniques of the *KECP* framework.

3.2 Query Prompt Construction

Since we transform the conventional span selection problem into the MLM generation problem, we need to construct prompt templates for each passage-query pair. In contrast to previous approaches (Brown et al., 2020; Gao et al., 2021) which generate templates by handcrafting or neural networks, we find that the query Q in EQA tasks naturally provides hints for prompt construction. Specifically, we design a template mapping $\mathcal{T}$ based on several heuristic rules (please refer to Appendix A for more details). For example, the query "What was one of the Norman's major exports?" can be transformed into a template: "`[MASK] [MASK] [MASK]` was one of the Norman's major exports". If a sentence does not match any of these rules, multiple `[MASK]` tokens will be directly added to the end of the query. The number of `[MASK]` tokens in prompts is regarded as a pre-defined hyper-parameter denotes as l_{mask}.

Let $Q_{prompt} = q'_1, q'_2, \cdots, q'_{m'}$ denote a query prompt where q'_i is a dispersed prompt token, m' is the length of query prompt. We concatenate the query prompt Q_{prompt} and the passage text P with some special tokens as input x_{input}:

$$x_{input} = \texttt{[CLS]}\, Q_{prompt}\, \texttt{[SEP]}\, P\, \texttt{[SEP]}, \quad (1)$$

where `[CLS]` and `[SEP]` are two special tokens that represent the start and separate token in PLMs.

3.3 Knowledge-aware Prompt Encoder (KPE)

As mentioned above, to remedy the dilemma that vanilla MLM has poor abilities of model inference, empirical evidence suggests that we can introduce the KB to assist boosting PLMs. For example, when we ask the question "What was one of the Norman's major exports?", we expect the model to capture more semantics information of the selected tokens "Norman's major exports", which is the imperative component for model inference.

To achieve this goal, inspired by Liu et al. (2021b) where pseudo tokens are added to the input with continuous prompt embeddings, we propose the *Knowledge-aware Prompt Encoder* (KPE) to aggregate the multiple-resource knowledge to the input embeddings of the query prompt. It consists of two main steps: *Passage Knowledge Injection* (PKI) and *Passage-to-Prompt Injection* (PPI), where the first aims to generate knowledge-enhanced representations from passage context and KB, while the second is to flow these representations to the selected tokens of query prompts.

3.3.1 Passage Knowledge Injection (PKI)

For knowledge injection, we first introduce two embedding mappings $\mathcal{E}_{wr}(\cdot)$ and $\mathcal{E}_{kn}(\cdot)$, where $\mathcal{E}_{wr}(\cdot)$ aims to map the input token to the word embeddings from the PLM embedding table, $\mathcal{E}_{kn}(\cdot)$ denotes to map the input token to the KB embeddings pre-trained by the ConVE (Dettmers et al., 2018) algorithm based on WikiData5M (Wang et al., 2021) [3].

In the beginning, all the tokens in x_{input} are encoded into word embeddings $\mathbf{x}$. Hence, we can obtain the embeddings of query prompt and passage, denote as $\mathbf{Q} = \mathcal{E}_{wr}(Q_{prompt}) \in \mathbb{R}^{m' \times h}$ and $\mathbf{P} = \mathcal{E}_{wr}(P) \in \mathbb{R}^{n \times h}$. Additionally, for each token $p_i \in P$, we retrieve the entities from the KB that have the same lemma with p_i, and the averaged entity embeddings are stored as their KB embeddings. Formally, we generate the KB embeddings $\mathbf{p}_i^{kn}$ of the passage token p_i:

$$\mathbf{p}_i^{kn} = Mean(\mathbf{e}_j | lem(p_i) = lem(e_j)), \quad (2)$$

where lem is the lemmatization operator (Dai et al., 2021), $\mathbf{e}_j = \mathcal{E}_{kn}(e_j)$. We then directly combine word embeddings and KB embeddings by $\mathbf{g}_i = \mathbf{p}_i + \mathbf{p}_i^{kn}$, where $\mathbf{p}_i$ is the word embeddings of i-th token in the passage text. $\mathbf{g}_i \in \mathbb{R}^h$ is the embeddings with knowledge injected. Finally, we obtain knowledge-enhanced representations denoted as $\mathbf{G} = \mathbf{g}_1\mathbf{g}_2, \cdots, \mathbf{g}_n$, where $\mathbf{G} \in \mathbb{R}^{n \times h}$.

[3] URL: https://deepgraphlearning.github.io/project/wikidata5m.

3.3.2 Passage-to-Prompt Injection (PPI)

The goal of PPI is to enhance the representations $\mathbf{Q}$ of selected prompt tokens by the interaction between the query and the passage representations. As discovered by (Zhang et al., 2021), injecting too much background knowledge may harm the performance of downstream tasks, hence we only inject knowledge to the representations of part of the prompt tokens. To be more specific, given $r(< m')$ selected prompt tokens $q_j^{sp} \in Q_{prompt}$, we create the corresponding embeddings $\mathbf{q}^{sp} \in \mathbb{R}^{r \times h}$ by looking up the embeddings from $\mathbf{Q}$. For each prompt token, we leverage self-attention to obtain the soft embeddings $\mathbf{v}^{sp} \in \mathbb{R}^{r \times h}$:

$$\mathbf{v}^{sp} = \text{SoftMax}(\mathbf{q}^{sp}\mathbf{W}_\alpha\mathbf{G}^\mathsf{T}/\sqrt{d})\mathbf{G}, \quad (3)$$

where $\mathbf{W}_\alpha \in \mathbb{R}^{h \times h}$ is the trainable matrix. d denotes the scale value. We add residual connection to $\mathbf{v}^{sp}$ and $\mathbf{q}^{sp}$ by linear combination as $\mathbf{u}^{sp} = \mathbf{v}^{sp} + \mathbf{q}^{sp}$, where $\mathbf{u}^{sp}$ denotes the enhanced representations of selected prompt tokens.

Finally, we only replace the original word embeddings $\mathbf{x}$ of selected prompt tokens $\mathbf{q}^{sp}$ with $\mathbf{u}^{sp}$ in the PLM's embeddings layer. To this end, we use very few parameters to implement the rich knowledge injection, which alleviate over-fitting during few-shot learning.

3.4 Span-level Contrastive Learning (SCL)

As mentioned above, many negative span texts in the passage have similar and confusing semantics with the correct answer. This may cause the PLM to generate wrong results. For example, given the passage "Google News releases that Apple founder Steve Jobs will speak about the new iPhone 4 product at a press conference in 2014." and the query "Which company makes iPhone 4?". The model is inevitably confused by some similar entities. For examples, "Google" is also a company name but is insight of the entity "Apple" in the sentence, and "Steve Jobs" is not a company name although it is as expected from the answer.

Inspired by contrastive learning (Chen et al., 2020), we can distinguish between the positive and negative predictions and alleviate this confusion problem. Specifically, we firstly obtain a series of span texts by the slide window, suppose as

$Y_i' = p_{k_i'} \cdots p_{l_i'}$, where k_i' and l_i' denote the start and the end positions of the i-th span. Then, we filter out some negative spans that have similar semantics with the correct answer Y. In detail, we follow SpanBERT (Joshi et al., 2020) to represent each span by the span boundary. The embeddings that we choose are the knowledge-enhanced representations $\mathbf{G}$ in Section 3.3, which consists of rich context and knowledge semantics. For each positive-negative pair (Y, Y_i'), we compute the similarity score and the candidate intervals with top-S similarity scores are selected as the negative answers, which can be viewed as the semantically confusion w.r.t. the correct answer. For the i-th negative answer Y_i', we have:

$$\mathcal{Z}_i' = \sum_j \Pr(Y_{ij}' | P, Q; \Theta), \tag{4}$$

where $\Pr$ denotes the prediction function of the MLM head. Y_{ij}' denotes the j-th token in the corresponding span. We can also calculate the score $\mathcal{Z}$ of the ground truth in the same manner. Hence, for each training sample, the objective of the span-level contrastive learning can be formulated as:

$$\mathcal{L}_{SCL} = -\frac{1}{S+1} \log[\frac{\exp\{\mathcal{Z}\}}{\exp\{\mathcal{Z}\} + \sum_{i=1}^{S} \exp\{\mathcal{Z}_i'\}}], \tag{5}$$

Finally, the total loss function is written as follows:

$$\mathcal{L} = \mathcal{L}_{MLM} + \lambda \mathcal{L}_{SCL} + \gamma ||\Theta||, \tag{6}$$

where $\mathcal{L}_{MLM}$ denotes the training objective of token-level MLM. Θ denotes the model parameters. $\lambda, \gamma \in [0, 1]$ are the balancing hyper-parameter and the regularization hyper-parameter, respectively.

3.5 Model-free Prefix-tree Decoder

Different from conventional text generation, we should guarantee that the generated answer must be the **sub-string** in the passage text. In other words, the searching space of each position is constrained by the prefix token. For example, in Figure 2, if the prediction of the first `[MASK]` token in Q_{prompt} is "fighting", the searching space of the second token shrinks down to "{horsemen, `[END]`}", where `[END]` is the special token as the answer terminator. We implement a simple model-free prefix-tree (i.e. trie-tree) decoder without any parameters, which is a highly-efficient data structure that preserves the dependency of each passage token. At each `[MASK]` position, we use beam search algorithm to select top-S results. The predicted text of the masked positions with highest score calculated by Eq. (4) is selected as the final answer.

4 Experiments

In this section, we conduct extensive experiments to evaluate the performance of our framework.

4.1 Baselines

To evaluate our proposed method, we consider the following methods as strong baselines: 1) **RoBERTa** (Liu et al., 2019) is the optimized version of BERT, which introduces dynamic masking strategy. 2) **SpanBERT** (Joshi et al., 2020) utilizes the span masking strategy and predicts the masked tokens based on boundary representations. 3) **WKLM** (Xiong et al., 2020) belongs to knowledge-enhanced PLM, which continue to pre-trains on BERT with a novel entity replacement task. 4) **Splinter** (Ram et al., 2021) is the first work to regard span selection as a pre-training task for EQA. 5) **P-tuning-V2** (Liu et al., 2021a) is the prompt-based baseline for text generation tasks.

4.2 Benchmarks

Our framework is evaluated over two benchmarks, including SQuAD2.0 (Rajpurkar et al., 2018) and MRQA 2019 shared task (Fisch et al., 2019). The statistics of each dataset are shown in Appendix.

SQuAD 2.0 (Rajpurkar et al., 2018): It is a widely-used EQA benchmark, combining 43k unanswerable examples with original 87k answerable examples in SQuAD1.1 (Rajpurkar et al., 2016). As the testing set is not publicly available, we use the public development set for the evaluation.

MRQA 2019 shared task (Fisch et al., 2019): It is a shared task containing 6 EQA datasets formed in a unified format, such as SQuAD1.1 (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2017), TriviaQA (Joshi et al., 2017), SearchQA (Dunn et al., 2017), HotpotQA (Yang et al., 2018) and NQ (Kwiatkowski et al., 2019). Following (Ram et al., 2021), we use the subset of Split I, where the training set is used for training and the development set is for evaluation.

4.3 Implementation Details

Follow the same settings as in (Ram et al., 2021), for each EQA dataset, we randomly choose K samples from the original training set to construct the few-shot training set and development set, respectively. As the test set is not available, we evaluate the model on the whole development set.

Paradigm	Methods	Use KB	EQA Datasets						
			SQuAD2.0	SQuAD1.1	NewsQA	TriviaQA	SearchQA	HotpotQA	NQ.
FT	RoBERTa	No	9.55%+1.9	12.50%+2.7	6.24%+0.8	12.00%+1.5	11.87%+1.1	12.05%+1.4	19.68%+1.9
	SpanBERT	No	9.90%+1.0	12.50%+1.2	6.00%+2.0	12.80%+1.3	13.00%+1.7	12.60%+1.5	19.15%+2.0
	WKLM *	Yes	17.22%+2.0	16.30%+1.0	8.80%+1.5	14.16%+1.8	15.30%+1.4	13.30%+1.4	19.85%+1.7
	Splinter	No	53.05%+5.2	54.60%+5.9	20.80%+2.8	18.90%+1.6	26.30%+2.5	24.00%+0.9	27.40%+1.2
PT	RoBERTa †	No	39.50%+1.1	27.10%+2.0	12.20%+3.9	16.82%+2.0	19.10%+1.8	22.26%+1.9	20.18%+2.2
	P-tuning V2	No	60.48%+4.2	59.10%+4.4	22.33%+2.9	22.42%+0.7	28.08%+4.1	26.33%+2.3	27.52%+2.4
	KECP w/o. KPE	No	63.07%+3.6	64.22%+4.3	23.80%+2.0	21.35%+0.8	29.41%+3.1	27.80%+2.6	27.95%+2.4
	KECP	Yes	**75.45%**+3.8	**67.05%**+4.7	**28.38%**+1.9	**24.80%**+2.4	**35.33%**+2.4	**33.90%**+2.0	**31.85%**+2.2

Table 1: The averaged F1 performance of each benchmarks with standard deviation in few-shot scenario ($K = 16$). FT and PT denote Fine-tuning and Prompt-tuning paradigms, respectively. RoBERTa† in PT uses the vanilla MLM head to predict the answer text. WKLM * denotes our re-produced version based on RoBERTa-base.

In our experiments, the underlying PLM is RoBERTa-base (Liu et al., 2019) and the default hyper-parameters are initialized from the HuggingFace [4]. We train our model by the Adam algorithm. The learning rate for MLM is fixed as 1e-5, while the initial learning rate for other new modules (self-attention in PPI) in *KECP* is set in {1e-5, 3e-5, 5e-5, 1e-4} with a warm-up rate of 0.1, the L2 weight decay value is $\gamma = 0.01$. The balance hyper-parameter is set as $\lambda = 0.5$. The number of `[MASK]` tokens in query prompts is $l_{mask} = 10$. The number of negative spans is $S = 5$. In few-shot settings, the definition scope of the sample number is $K \in \{16, 32, 64, \cdots, 512\}$. We set the batch size and the epoch number as 8 and 64, respectively. During experiments, we choose five different random seeds $\{12, 21, 42, 87, 100\}$ (Gao et al., 2021) and report the averaged performance. Because the generated answer text can be easy converted to a span with start and end position, we follow (Ram et al., 2021) to use the same F1 metric protocol, which measures the average overlap between the predicted and the ground-truth answer texts at the token level.

4.4 Main Results

As shown in Table 1, the results indicate that *KECP* outperforms all baselines with only 16 training examples. Surprisingly, we achieve 75.45% and 67.05% F1 values over SQuAD2.0 (Rajpurkar et al., 2018) and SQuAD1.1 (Rajpurkar et al., 2016) with only 16 training examples, which outperforms the state-of-the-art method Splinter (Ram et al., 2021) by 22.40% and 12.45%, respectively. We also observe that the result of RoBERTa† with vanilla MLM head is lower than any other of PT methods. It explains the necessity of the improvement of reasoning ability and the constraints on answer generation. To make fairly comparison, we also report the results of *KECP*$_{\text{w/o. KPE}}$, which is the basic model without injected KB. It makes a substantial improvement in all tasks, showing that prompt-tuning based on MLM generation is more suitable than span selection pre-training. In addition, we find that all results of traditional PLMs (e.g. RoBERTa (Liu et al., 2019) and SpanBERT (Joshi et al., 2020)) over seven tasks are lower than WKLM (Xiong et al., 2020), which injects domain-related knowledge into the PLM. Simultaneously, our model outperforms P-tuning V2 (Liu et al., 2021a) and *KECP*$_{\text{w/o. KPE}}$ by a large margin. These phenomenon indicate that EQA tasks can be further improved by injecting domain-related knowledge.

[4] https://huggingface.co/transformers/index.html.

#Training Samples⟶	16	1024	All
KECP	**75.45%**	**84.90%**	**90.85%**
w/o. KPE (w/o. PKI & PPI)	63.07%	73.17%	84.90%
w/o. PPI	73.36%	82.53%	90.70%
w/o. SCL	66.27%	74.40%	86.10%

Table 2: The ablation F1 scores over SQuAD2.0 of *KECP* for few-shot learning setting. w/o. denotes that we only remove one component from *KECP*.

Prompt Mapping	SQuAD2.0	NewsQA	HotpotQA
$\mathcal{T}_1$ (None)	89.19%	72.15%	79.26%
$\mathcal{T}_2$ (Manual)	88.62%	72.70%	78.35%
$\mathcal{T}$ **(Proposed)**	**90.85%**	**73.28%**	**81.19%**

Table 3: Comparison with proposed prompt template mapping $\mathcal{T}$ with two alternative methods $\mathcal{T}_1$ and $\mathcal{T}_2$.

4.5 Detailed Analysis and Discussions

Ablation Study. To further understand why *KECP* achieves high performance, we perform an ablation analysis to better validate the contributions of each component. For simplicity, we only present the

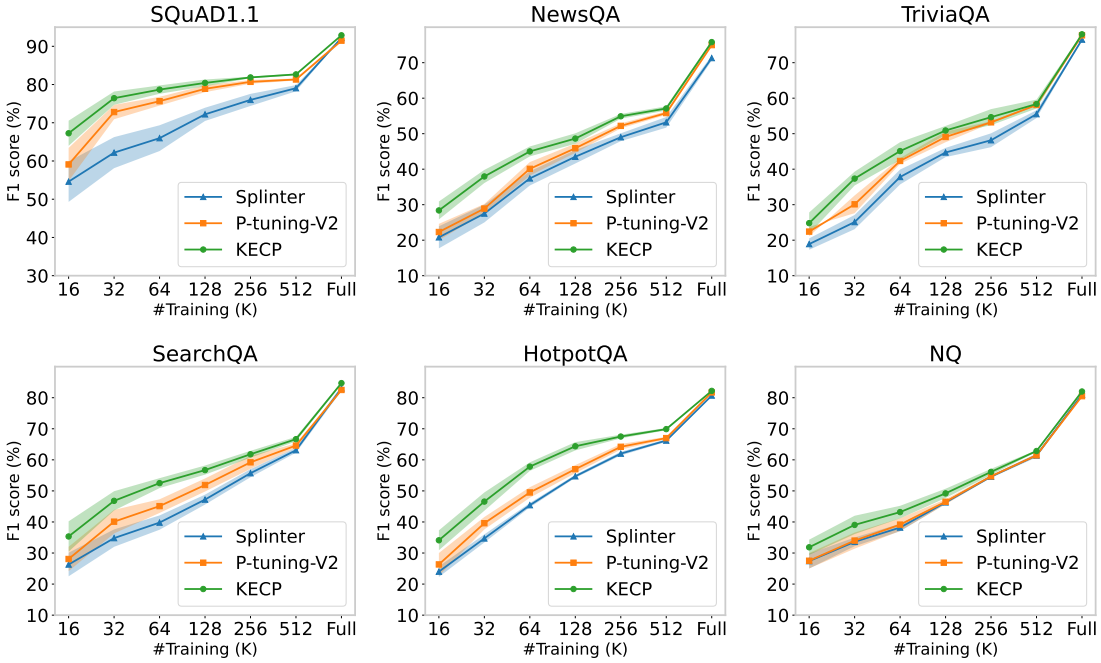

Figure 3: Results of sample efficiency analysis. We compare *KECP* with strong baselines with different numbers of training samples K over MRQA 2019 shared tasks. "Full" denotes to the models trained over full training data.

ablation experimental results on SQuAD2.0 with 16, 1024 and all training samples.

We show all ablation experiments in Table 2, where w/o. KPE equals to the model without any domain-related knowledge (denotes to remove both PKI & PPI). w/o. PPI denotes to only inject knowledge into selected prompt tokens without trainable self-attention. w/o. SCL means training without span-level contrastive learning (i.e. $\lambda = 0$). We find that no matter which module is removed, the effect is decreasing. Particularly, when we remove both PKI and PPI, the performance is decreased by 12.38%, 11.73% and 5.95%, respectively. The declines are larger than other cases, which indicates the significant impact of the passage-aware knowledge enhancement. We also find the SCL employed in this work also plays an important role in our framework, indicating that there are many confusing texts in the passage that need to be effectively distinguished by contrastive learning.

Sample Efficiency. We further explore the model effects with different numbers K of training samples. Figure 3 shows the performance with the different numbers of training samples over the MRQA 2019 shared task (Fisch et al., 2019). Each point refers the averaged score across 5 randomly sampled datasets. We observe that our *KECP* consistently achieves higher scores regardless of the number of training samples. In particular, our method has more obvious advantages in low-resource scenarios than in full data settings. In addition, the results also indicate that prompt-tuning can be another novel paradigm for EQA.

Parameters	Values	Few	Full	Time
l_{mask} =? $\lambda = 0.5$ $S = 5$	4	39.20%	77.17%	0.9s
	7	41.35%	82.90%	1.3s
	10	**42.30%**	**83.27%**	**1.5s**
	13	41.98%	82.84%	1.9s
λ =? $l_{mask} = 10$ $S = 5$	0	37.62%	76.91%	1.2s
	0.25	41.80%	82.99%	1.5s
	0.5	**42.30%**	**83.27%**	**1.5s**
	0.75	42.09%	83.13%	1.5s
	1.0	40.10%	81.70%	1.6s
S =? $\lambda = 0.5$ $l_{mask} = 10$	3	39.25%	80.02%	1.3s
	5	**42.30%**	**83.27%**	**1.5s**
	7	42.30%	82.98%	1.9s
	9	42.41%	83.32%	2.3s

Table 4: The efficiency of hyper-parameters. All results are the average results of all datasets in both few-shot (Few) and full training data (Full) scenarios.

Effects of Different Prompt Templates. In this part, we design two other template mappings:

- $\mathcal{T}_1$ **(None)**: directly adding a series of `[MASK]` tokens without any template tokens.
- $\mathcal{T}_2$ **(Manual)**: designing a fixed template with multiple `[MASK]` tokens (e.g., "The answer is `[MASK]`$\cdots$").

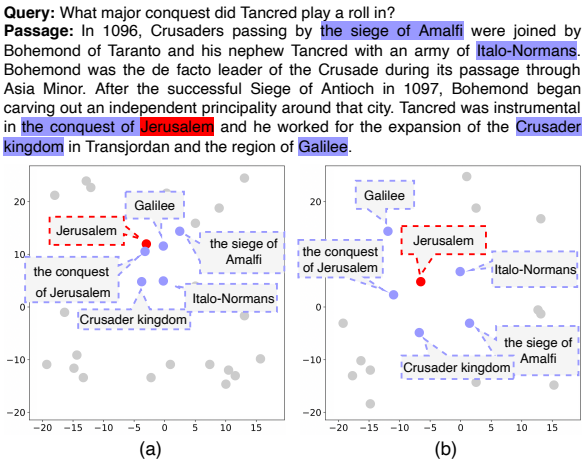

Figure 4: Visualizations of answer span texts. (a) is the result of the PLM without contrastive learning. (b) is the result of the PLM with contrastive learning.

To evaluate the efficiency of our proposed template mapping method compared with these baselines, we randomly select three tasks (i.e., SQuAD2.0, NewsQA and HotpotQA) and train models with full data. As shown in Table 3, we find that two simple templates have the similar performance. Our proposed method outperforms them by more than 1.0% in terms of F1 score. [5]

Hyper-parameter Analysis. In this part, we investigate on some hyper-parameters in our framework, including the number of masked tokens l_{mask}, the balance coefficient λ and the negative spans sampling number S. We also record the inference time over a batch with 8 testing examples. As shown in Table 4, when we tune l_{mask}, λ and S are fixed as 0.5 and 5, respectively. Results show that length of masked tokens plays an important role in prompt-tuning. We fix $l_{mask} = 10, S = 5$ and tune λ, and achieve the best performance when $\lambda = 0.5$. We fix $\lambda = 0.5, l_{mask} = 10$ and tune the parameter S. We find the overall performance increases when increasing the sampled negatives. However, we recommend to set S around 5 due to the faster inference speed.

Effectiveness of Span-level Contrastive Learning. Furthermore, to evaluate how the model improved by span-level contrastive learning (SCL), we randomly select one example from the development set of SQuAD2.0 (Rajpurkar et al., 2018), and visualize it by t-sne (Van der Maaten and Hinton, 2008) to gain more insight into the model performance. As shown in Figure 4, the correct answer is "Jerusalem" (in red). We also obtain 5 negative spans (in blue) which may be confused with the correct answer. When the PLM is trained without SCL, in Figure 4(a), we observe that all negative answers are agglomerated together with the correct answer "Jerusalem". It makes the PLM hard to search for the suitable results. In contrast, Figure 4(b) represents the model trained with SCL. The result demonstrates that all negative spans can be better divided with the correct answer "Jerusalem". This shows that SCL in our *KECP* framework is reliable and can improve the performance for EQA.

[5] We also provide intuitive cases in the experiments. More details can be found in the appendix.

Method	SQuAD2.0	NewsQA	HotpotQA
RoBERTa (#1)	83.47%	69.80%	78.70%
KECP (#1)	58.06%	51.30%	59.64%
KECP (#3)	74.57%	64.78%	72.11%
***KECP* (#5)**	**86.44%**	**72.90%**	**81.43%**

Table 5: The accuracy of predicting the first `[MASK]` in the query prompt with full training samples for each task. #n_w denotes the window size.

The Accuracy of Answer Generation. A major difference between previous works and ours is that we model the EQA task as text generation. Intuitively, if the model correctly generates the first answer token, it is easy to generate the remaining answer tokens because of the very small search space. Therefore, we analyze how difficult it is for the model to generate the first token correctly. Specifically, we check whether the generated first token and the first token of the ground truth are within a fixed window size n_w. As shown in Table 5, we find the accuracy of our method is lower than RoBERTa-base (Liu et al., 2019) when $n_w = 1$. Yet, we achieve the best performance when increasing the window size n_w to 5. We think that our *KECP* can generate some rehabilitation text for the answer. For example in Figure 4, the PLM may generate "the conquest of Jerusalem" rather than the correct answer with single token "Jerusalem". This phenomenon reflects the reason why we achieve lower accuracy when $n_w = 1$. But, we think that the generated results are still in the vicinity of the correct answer.

5 Conclusion

To bridge the gap between the pre-training and fine-tuning objectives, *KECP* views EQA as an answer generation task. In *KECP*, the knowledge-aware

prompt encoder injects external domain-related knowledge into the passage, and then enhances the representations of selected prompt tokens in the query. The span-level contrastive learning objective is proposed to improve the performance of EQA. Experiments on multiple benchmarks in both instance-level and task-level few-shot scenarios show that our framework consistently outperforms the state-of-the-art methods.

Limitations

Our work addresses the problem of few-shot span-based EQA only (a type of MRC tasks in NLP) based on contrastive prompting. We believe that prompt-tuning can be applied to other types of MRC tasks, such as cloze-style MRC and multiple-choice MRC. We leave it as future work. Another limitation is that the correct generation of the first answer token is still not satisfactory, as discussed in the experiments. We will also improve the performance of *KECP* by applying controllable text generation techniques in the future.

Ethical Considerations

Our contribution in this work is fully methodological, namely a knowledge-enhanced contrastive prompting (*KECP*) to boost the performance of few-shot extractive question answering. Hence, there are no direct negative social impacts of our work.

Acknowledgments

This work has been supported by the National Natural Science Foundation of China under Grant No. U1911203, Alibaba Group through the Alibaba Innovation Research Program, the National Natural Science Foundation of China under Grant No. 61877018, the Research Project of Shanghai Science and Technology Commission (20dz2260300) and the Fundamental Research Funds for the Central Universities.

References

Haytham Assem, Rajdeep Sarkar, and Sourav Dutta. 2021. Qasar: Self-supervised learning framework for extractive question answering. In *Big Data*, pages 1797–1808.

Tom B. Brown, Benjamin Mann, and etc. Nick Ryder. 2020. Language models are few-shot learners. In *NeurIPS*.

Rakesh Chada and Pradeep Natarajan. 2021. Fewshotqa: A simple framework for few-shot learning of question answering tasks using pre-trained text-to-text models. In *EMNLP*, pages 6081–6090.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A simple framework for contrastive learning of visual representations. In *ICML*, volume 119, pages 1597–1607.

Damai Dai, Hua Zheng, Zhifang Sui, and Baobao Chang. 2021. Incorporating connections beyond knowledge embeddings: A plug-and-play module to enhance commonsense reasoning in machine reading comprehension. *CoRR*, abs/2103.14443.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Güney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *CoRR*, abs/1704.05179.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *EMNLP*, pages 1–13.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *ACL*, pages 3816–3830.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2021. PTR: prompt tuning with rules for text classification. *CoRR*, abs/2105.11259.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *TACL*, 64–77.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*, pages 1601–1611.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, and et al. 2019. Natural questions: a benchmark for question answering research. *TACL*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. RACE: large-scale reading comprehension dataset from examinations. In *EMNLP*, pages 785–794.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *CoNLL*, pages 333–342.

Xiang Lisa Li and Percy Liang. 2021a. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL/IJCNLP*, pages 4582–4597. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021b. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, pages 4582–4597.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021a. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *CoRR*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. GPT understands, too. *CoRR*, abs/2103.10385.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, and et al. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying lms with mixtures of soft prompts. In *NAACL-HLT*, pages 5203–5212.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*.

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

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *EACL*, pages 255–269.

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 *EMNLP*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *WRLNLP*, pages 191–200.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *NIPS*, pages 2692–2700.

Chao Wang and Hui Jiang. 2019. Explicit utilization of general knowledge in machine reading comprehension. In *ACL*, pages 2263–2272. Association for Computational Linguistics.

Chengyu Wang, Minghui Qiu, Taolin Zhang, Tingting Liu, Lei Li, Jianing Wang, Ming Wang, Jun Huang, and Wei Lin. 2022. Easynlp: A comprehensive and easy-to-use toolkit for natural language processing. *CoRR*, abs/2205.00258.

Shuohang Wang and Jing Jiang. 2017. Machine comprehension using match-lstm and answer pointer. In *ICLR*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *ACL*, pages 189–198.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *TACL*, 9:176–194.

Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. 2020. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. In *ICLR*.

An Yang, Quan Wang, Jing Liu, Kai Liu, Yajuan Lyu, Hua Wu, Qiaoqiao She, and Sujian Li. 2019. Enhancing pre-trained language representations with rich knowledge for machine reading comprehension. In *ACL*, pages 2346–2357.

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. In *EMNLP*, pages 2369–2380.

Ningyu Zhang, Shumin Deng, Xu Cheng, Xi Chen, Yichi Zhang, Wei Zhang, Huajun Chen, and Hangzhou Innovation Center. 2021. Drop redundant, shrink irrelevant: Selective knowledge injection for language pretraining. In *IJCAI*.

A The Mapping Rules of Query Prompt

Based on the analysis on the syntactic forms of queries from SQuAD (Rajpurkar et al., 2016, 2018) and the MRQA 2019 shared task (Fisch et al., 2019), we find that the queries in EQA can be directly transformed into the prompt templates with multiple `[MASK]` tokens. Let $\mathcal{T} : s \rightarrow s'$ be the prompt mapping where s and s' represent the original sentence and the prompt template, respectively. We list four rules for query prompt construction with corresponding example:

- **Rule 1**. $\mathcal{T}$(<s> be/done $\cdots$?) $\rightarrow \cdots$ [MASK] $\cdots$ be/done $\cdots$, where <s> can be chosen among {"what", "who", "whose", "whom", "which", "how"}.

Rule	Original Query	Query Prompt
Rule 1	A Japanese manga series based on a 16 year old high school student Ichitaka Seto, is written and illustrated by someone born in what year?	A Japanese manga series based on a 16 year old high school student Ichitaka Seto, is written and illustrated by someone born in `[MASK] [MASK]`.
Rule 2	Where is the company that Sachin Warrier worked for as a software engineer?	The company that Sachin Warrier worked for as a software engineer is at the place of `[MASK] [MASK]`.
Rule 3	When the Canberra was introduced to service with the Royal Air Force (RAF), the type's first operator, in May 1951, it became the service's first jet-powered bomber aircraft.	The Canberra was introduced to service with the Royal Air Force (RAF) at the time of `[MASK] [MASK]`, the type's first operator, in May 1951, it became the service's first jet-powered bomber aircraft.
Rule 4	Why did Rudolf Hess stop serving Hitler in 1941?	The reason why did Rudolf Hess stop serving Hitler in 1941 is that `[MASK] [MASK]`.
Other	How much longer after he was born did Werder Bremen get founded in the northwest German federal state Free Hanseatic City of Bremen?	How much longer after he was born did Werder Bremen get founded in the northwest German federal state Free Hanseatic City of Bremen? `[MASK] [MASK]`.

Table 6: Example of each query prompt mapping rule.

Dataset	#Train	#Dev	#All
SQuAD2.0	118,446	11,873	130,319
SQuAD1.1	86,588	10,507	97,095
NewsQA	74,160	4,212	78,372
TriviaQA	61,688	7,785	69,573
SearchQA	117,384	16,980	134,364
HotpotQA	72,928	5,904	78,832
NQ	104,071	12,836	116,907

Table 7: The statistics of multiple EQA benchmarks.

- **Rule 2**. $\mathcal{T}$(`where be/done`$\cdots$?) $\rightarrow \cdots$ `be/done at the place of [MASK]` $\cdots$.
- **Rule 3**. $\mathcal{T}$(`when be/done` $\cdots$?) $\rightarrow \cdots$ `be/done at the time of [MASK]` $\cdots$.
- **Rule 4**. $\mathcal{T}$(`why be/done` $\cdots$?) $\rightarrow$ `the reason why` $\cdots$ `be/done [MASK]` $\cdots$.

For the query that does not match these rules will be directly appended with multiple masked language tokens. Table 6 shows the examples of each mapping rule.

B Data Sources

In this section, we give more details on data sources used in the experiments.

B.1 The Benchmarks of EQA

We choose two widely used EQA benchmarks for the evaluation, including SQuAD2.0 (Rajpurkar et al., 2018) and the MRQA 2019 shared task (Fisch et al., 2019). Specifically, the MRQA 2019 shared task was proposed to evaluate the domain transferable of neural models, where the authors selected 18 distinct question answering datasets, then adapted and unified them into the same format. They divided all datasets into 3 splits, where Split I is used for model training and development, Split II is used for development only and Split III is used for evaluation. Because our work focuses on few-shot learning settings, we simply choose 6 dataset from Split I in our experiments, including SQuAD1.1 (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2017), TriviaQA (Joshi et al., 2017), SearchQA (Dunn et al., 2017), HotpotQA (Yang et al., 2018) and NQ (Kwiatkowski et al., 2019). We also choose SQuAD2.0 (Rajpurkar et al., 2018) to conduct evaluations.

In few-shot learning settings, for each dataset, we randomly select K examples with five different random seeds for training and development, respectively. For the full data settings, we follow the same settings of Splinter (Ram et al., 2021) to use all training data.

B.2 External Knowledge Base

For the domain-related knowledge base, we use WikiData5M (Wang et al., 2021), which is a large-scale knowledge graph aligned with text descriptions from the corresponding Wikipedia pages. It consists of 4,594,485 entities, 20,510,107 triples and 822 relation types. We use the ConVE (Dettmers et al., 2018) algorithm to pretrain the entity and relation embeddings. We set its dimension as 512, the negative sampling size as 64, the batch size as 128 and the learning rate as 0.001. Finally, we only store the embeddings of all

#Training Samples⟶	16	1024	All
KECP	**75.45%**	**84.90%**	**90.85%**
w/o. SCL	66.27%	74.40%	86.10%
w/o. filter & sort	71.35%	79.05%	87.80%
w/o. dist	74.90%	84.60%	90.55%

Table 8: The ablation F1 scores over SQuAD2.0 to evaluate the importance of each technique in the confusion span contrastive task. w/o. denotes that we only remove one component from *KECP*.

the entities. For the passage knowledge injection, we use entity linking tools (e.g, TAGME tool in python [6]) to align the entity mentions in passages. The embeddings of tokens are calculated by the lemmatization operator (Dai et al., 2021).

C Details of Negative Span Sampling

To construct negative spans for span-level contrastive learning (SCL), we follow a simple pipeline to implement confusion span sampling. At first, we use a sliding window to obtain a series of span texts. Next, we filter out span texts which are incomplete sequences or dissatisfy with the lexical and grammatical rules. Finally, we calculate the semantic similarity between each candidate span text and the true answer. Formally, suppose $Y = y_1, y_2, \cdots, y_l$ is the ground truth. Given one candidate span $X = x_1, x_2, \cdots, y_{l'}$, where l, l' are the lengths of the ground truth and the candidate span text, respectively, we have:

$$\begin{aligned}\text{Sim}(X, Y) =& \texttt{dist}(X, Y) \\ & \cdot \cos(\frac{1}{l}\sum_{i=1}^{l}\mathbf{y}_i, \frac{1}{l'}\sum_{i=1}^{l'}\mathbf{y}'_i)\end{aligned} \tag{7}$$

where $\mathbf{y}_i$, $\mathbf{y}'_i$ denote the knowledge-injected representations of i-th token, respectively. $\cos(X, Y)$ aims to compute the cosine similarity between X and Y. We also introduce the $\texttt{dist}(X, Y)$ function to represent the normalized position distance between X and Y by the intuition that the text closer to the correct answer is prone to confusion. Specifically, for each candidate X, we obtain the distance between the first token of X and Y, and calculate the normalized weight for each candidate. For example in Figure 1, the distance between the candidate "avid Crusad" and the answer "fighting horsemen" is 16, and the normalized weight is 0.15.

We provide a brief ablation study for this module. Specifically, w/o. SCL means that we remove all techniques of this module (setting $\lambda = 0$ in Equation (6)). w/o. filter & sort denotes randomly sampling S spans without the pipeline. w/o. $\texttt{dist}$ represents setting $\texttt{dist}(X, Y) = 1$ in Equation (7). As shown in Table 8, the results demonstrate that our model can be improved by the combination of all techniques.

[6] https://pypi.org/project/tagme/.