KnowPrompt: Knowledge-aware Prompt-tuning for Few-shot Extractive Question Answering

Abstract

Owing to the rapid development of large-scale Pre-trained Language Models (PLMs), there has been remarkable success on span-based extractive Question Answering (QA) by leveraging the two-stage pre-training and fine-tuning paradigm. However, existing methods heavily rely on labeled training data, and may perform poorly when facing the few-shot learning scenario. In this paper, we propose *KnowPrompt*, a novel framework for extractive QA based on prompt-tuning over PLMs. Specifically, given a passage and a corresponding query, we first convert the query to a prompt template containing several masked language tokens. Next, the knowledge-aware prompt encoder is proposed to enhance the representations of the prompt tokens, which injects the knowledge from knowledge bases into prompt embeddings. We also transform the traditional span selection problem in extractive QA into the multiple-token generation problem, and propose two training schemes based on Masked Language Modeling (MLM) to generate the answer from the passage through beam search. Experiments on multiple benchmarks demonstrate that our method consistently outperforms strong baselines in two different few-shot learning settings by a large margin. Surprisingly, we achieve a 75.79% F1 value on SQuAD2.0 with only 16 training samples. [1]

1 Introduction

Span-based extractive Question Answering (QA) is one of the most challenging tasks of Machine Reading Comprehension (MRC). The task is based on an assumption that the answer text is present in the sub-string of the passage (Rajpurkar et al. 2016; Seo et al. 2016).

A majority of recent approaches employ Pre-trained Language Models (PLMs) to extract the answer from the passage. As shown in Figure 1(a), several models (Wang and Jiang 2019; Yang et al. 2019a; Dai et al. 2021) utilize two softmax heads or a pointer network (Vinyals, Fortunato, and Jaitly 2015) to predict the start and the end positions of the answer in the passage. Yet, these conventional fine-tuning frameworks heavily depend on the time-consuming and labor-intensive process of data annotation. These models may perform poorly with few training samples due to model over-fitting (Gao, Fisch, and Chen 2021). Additionally, there is a large gap between the pre-training objective of Masked Language Modeling (MLM) (i.e., the distribution over the entire vocabularies) and the fine-tuning objective of span selection (i.e., the distribution of positions), which hinders the transfer and adaptation of knowledge in PLMs to downstream MRC tasks (Brown, Mann, and Nick Ryder 2020). Ram et al. (2021) propose a straightforward solution named *Splinter* that leverages the span selection task during pre-training. However, it may cost a lot of computational resources to pre-train for different PLMs separately.

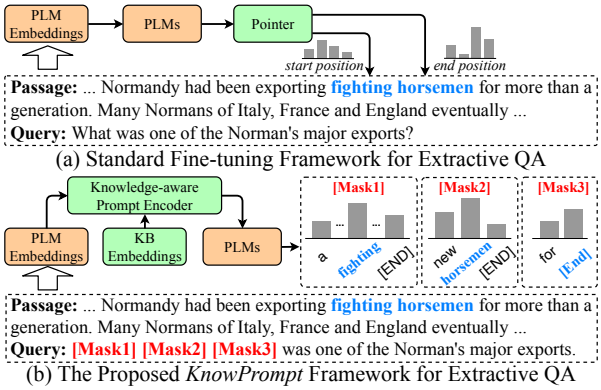

Figure 1: The comparison of standard fine-tuning and *KnowPrompt*. The blocks in orange and green denote the modules of PLMs and newly initialized modules, respectively. In our *KnowPrompt* framework, we transform the span selection task to the prompt-based answer generation problem.

Recently, the prompt-tuning paradigm has been proposed to bridge the gap between pre-training and fine-tuning. A variety of works (Schick and Schütze 2021a; Han et al. 2021; Li and Liang 2021a; Gao, Fisch, and Chen 2021; Liu et al. 2021) view the text classification task as a cloze-style problem. Consider the sentiment analysis task for an example. A task-specific prompt template (e.g., "It was `[MASK]`.") is added to the review text (e.g., "This dish is very attractive."). The probabilistic distribution of the masked token is calculated by the MLM head of the PLM and used for class label prediction (e.g., "delicious" for the positive label and "unappetizing" for the negative label). In this manner, we can

[1] All datasets are publicly available. Source codes are provided in the appendix and will be released to public upon acceptance.

use few samples to fast adapt the prior knowledge in PLMs to downstream tasks with few or no new parameters that are required to learn. It is worth mentioning that previous works mostly focus on simple text classification only. The extractive QA task, however, requires the complicated natural language inference and answer extraction process. Therefore, a natural question arises: *how to employ prompt-tuning over PLMs for extractive QA to achieve high performance in the few-shot learning setting?*

To solve this issue, in this work, we introduce *KnowPrompt*, a novel **Know**ledge-aware **Prompt**-tuning framework for few-shot extractive QA. Take Figure 1(b) for an example. Given a passage and a corresponding query as input, we transform the query into a prompt template containing multiple `[MASK]` tokens. Inspired by *P-tuning* (Liu et al. 2021), a lightweight knowledge-aware prompt encoder module is added to enrich the representations of selected tokens in the prompt by the integration of word and knowledge base embeddings. As for the output, we transform the span selection problem into the MLM generation objective. To ensure that the generated answer is included in the original passage, we propose two training schemes based on MLM with three losses. Particularly, the searching space of the `[MASK]` tokens in query prompts is limited to the tokens appearing in the passage. The final answer is then obtained by the beam search over all possible output token sequences.

In the experiments, we evaluate our proposed framework in two few-shot learning scenarios, namely instance-level and task-level. We employ SQuAD2.0 (Rajpurkar, Jia, and Liang 2018) and various benchmarks from the MRQA 2019 shared tasks (Fisch et al. 2019) to evaluate the effectiveness of *KnowPrompt*. The results show that our method consistently outperforms strong baselines by a large margin. Specifically, we achieve a 75.79% F1 value on SQuAD2.0 with only 16 training samples. To sum up, we make the following major contributions:

- We propose the *KnowPrompt* framework for few-shot extractive QA tasks based on prompt-tuning.
- In *KnowPrompt*, the traditional span selection problem is transformed into the MLM generation problem, which alleviates model over-fitting and bridges the gap between pre-training and fine-tuning. We further propose two training schemes with three losses based on MLM.
- Experiments show that *KnowPrompt* outperforms all the baselines in instance-level and task-level few-shot scenarios for extractive QA.

2 Related Work

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

2.1 Extractive Question Answering

Extractive QA 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 extractive QA has been released and attracted great interest (Rajpurkar et al. 2016; Rajpurkar, Jia, and Liang 2018; Lai et al. 2017; Trischler et al. 2017; Levy et al. 2017; Joshi et al. 2017; Fisch et al. 2019). Early works tend to aggregate the knowledge semantics through the attentive interactions between the passage and the query, including Match-LSTM (Wang and Jiang 2017), R-Net (Wang et al. 2017), Bi-DAF (Seo et al. 2016), QA-Net (Yu et al. 2018), etc.

Recently, benefited from the powerful modeling abilities of PLMs, we have witnessed the qualitative improvement of MRC. These PLMs include GPT (Radford et al. 2018), ELMo (Peters et al. 2018), BERT (Lan et al. 2020), XL-Net (Yang et al. 2019b), RoBERTa (Liu et al. 2019), ALBERT (Lan et al. 2020), SpanBERT (Joshi et al. 2020), etc., which are pre-trained on the large-scale corpus. Based on these PLMs, several fine-tuning based approaches (Zhang et al. 2020; Wang and Jiang 2019; Yang et al. 2019a; Qiu et al. 2019; Zhang, Yang, and Zhao 2021; Peng et al. 2021) aim to stack new modules to integrate knowledge, and use pointers (such as softmax heads of the PLMs or the pointer network (Vinyals, Fortunato, and Jaitly 2015)) to obtain the start and the end positions of the answer span from the passage.

However, this fine-tuning based technique may cause over-fitting in the few-shot learning setting. To solve the problem, Ram et al. (2021) is the first research for few-shot extractive QA by pre-training over the span selection task, but it costs much time to pre-train the span selection objective for different PLMs. On the contrary, we leverage prompt-tuning for few-shot extractive QA without any additional pre-training steps.

2.2 Prompt-tuning for PLMs

The huge GPT-3 model (Brown, Mann, and Nick Ryder 2020) enables few-shot learning for various NLP tasks without fine-tuning, which relies on handcraft prompts and achieves outstanding performance. However, it is hard to handcraft the best prompt for specific tasks. To facilitate automatic prompt construction, Gao, Fisch, and Chen (2021) generate prompts from the T5 model (Raffel et al. 2020). Jiang et al. (2020) mines prompts from the training corpus. AutoPrompt (Shin et al. 2020) employs token-based gradient searching to detect prompts from texts. However, these approaches focus on discrete prompts only. P-tuning (Liu et al. 2021) learns continuous prompt embeddings with differentiable parameters. Other works (Han et al. 2021; Lester, Al-Rfou, and Constant 2021; Zou et al. 2021; Li and Liang 2021b; Qin and Eisner 2021; Schick and Schütze 2021b) also focus on continuous prompts. Our framework further extends previous works in that it incorporates the rich knowledge to enhance the token representations, and allows prompt-tuning on few-shot extractive QA.

3 The *KnowPrompt* Framework

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

3.1 Task Overview

Given a passage $P = p_1, p_2, \cdots, p_n$ and the corresponding query $Q = q_1, q_2, \cdots, q_m$, the goal is to find a sub-string of

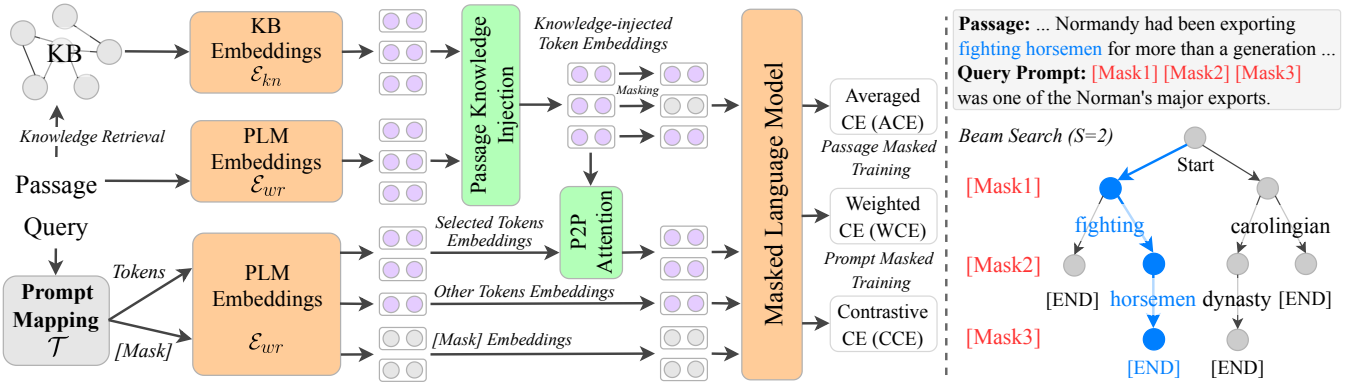

Figure 2: The *KnowPrompt* Framework. Given a passage and a query, we first obtain the query prompt by prompt mapping, and then feed them into PLM embeddings layer. The representations of selected tokens from query prompt are enhanced by knowledge injected passage token embeddings through proposed new modules (printed in green). At last, we introduce two novel training schemes with three losses, and generate the answer based on MLM with beam search. (Best viewed in color.)

the passage as the answer $Y = y_1, y_2, \cdots, y_l$, where n, m, l are the lengths of the passage, the query and the answer, respectively. p_i $(i = 1, \cdots, n)$, q_j $(j = 1, \cdots, m)$ and y_k $(k = 1, \cdots, l)$ refer to the tokens in P, Q and Y. Rather than span selection that predicting the start and the end positions, we view the extractive QA task as a non-autoregressive generation task based on PLMs (Su et al. 2021).

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. Let $\mathcal{T} : s \rightarrow s'$ be the prompt mapping where s and s' represent the original sentence and the prompt template, respectively. Recent approaches (Brown, Mann, and Nick Ryder 2020; Gao, Fisch, and Chen 2021; Jiang et al. 2020) generate templates based on handcrafting or neural networks for few-shot text classification. In contrastive to these works, we find that in extractive QA tasks, the query Q naturally provides hints for prompt construction. Based on the analysis on queries from SQuAD2.0 (Rajpurkar, Jia, and Liang 2018) and MRQA (Fisch et al. 2019), we design several heuristic rules to implement the template mappings:

- $\mathcal{T}$(`what/who/whose/whom/which be/done` $\cdots$?) $\rightarrow$ `[MASK]` $\cdots$ `be/done` $\cdots$
- $\mathcal{T}$(`where be/done` $\cdots$?) $\rightarrow \cdots$ `be/done at the place of [MASK]` $\cdots$
- $\mathcal{T}$(`when be/done` $\cdots$?) $\rightarrow \cdots$ `be/done at the time of [MASK]` $\cdots$
- $\mathcal{T}$(`how many/much/old/far/often be/done` $\cdots$?) $\rightarrow$ `[MASK]` $\cdots$ `be/done` $\cdots$
- $\mathcal{T}$(`why be/done` $\cdots$?) $\rightarrow$ `the reason why` $\cdots$ `be/done [MASK]` $\cdots$

For example, in Figure 2, the query "What was one of the Norman's major exports?" from SQuAD2.0 (Rajpurkar, Jia, and Liang 2018) can be transformed into a query prompt: "`[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 is a pre-defined hyper-parameter.

3.3 Knowledge-aware Prompt Encoder

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

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

A straightforward approach is to feed x_{input} directly into the PLM. However, we find this practice fails to produce the expected results due to the poor abilities of model inference. Inspired by Liu et al. (2021) where pseudo tokens are added to the input with continuous prompt embeddings, we propose the *Knowledge-aware Prompt Encoder* (KPE). It consists of two main components: *Passage Knowledge Injection* (PKI) and *Passage-to-Prompt Attention* (P2PA), where the first aims to integrate word embeddings and knowledge base embeddings for passage representation learning, and the second leverages the Gated Recurrent Units (GRUs) (Cho et al. 2014) with the attention mechanism to flow the knowledge-enhanced passage representations to the query prompts.

Passage Knowledge Injection (PKI) For knowledge injection, we first introduce two embedding mappings $\mathcal{E}_{wr}$ and $\mathcal{E}_{kn}$. $\mathcal{E}_{wr} : x \rightarrow \mathbf{w}$ is the word embedding mapping that maps the token x to the vector $\mathbf{w}$ retrieved from the embedding table of the PLM. $\mathcal{E}_{kn} : x \rightarrow \mathbf{e}$ is the knowledge base mapping that maps x to the knowledge embedding $\mathbf{e}$ pre-trained by the ConVE algorithm (Dettmers et al. 2018) [2] based on ConceptNet (Speer, Chin, and Havasi 2017) [3].

All the tokens in Q_{prompt} are encoded into word embeddings: $\mathbf{Q} = \mathcal{E}_{wr}(Q_{prompt}) \in \mathbb{R}^{m' \times h}$, where h is the em-

[2] Url: https://github.com/TimDettmers/ConvE.

[3] Url: https://github.com/commonsense/conceptnet5.

bedding size. Additionally, for each token $p_i \in P$, we retrieve the entities from the knowledge base that have the same lemma with p_i, and the averaged entity embeddings are stored as their knowledge base embeddings. Formally, we generate the knowledge base representation $\mathbf{p}_i^{kn}$ of the passage token p_i:

$$\mathbf{p}_i^{kn} = Mean(\mathcal{E}_{kn}(e_j)|lemma(p_i) = lemma(e_j)), \quad (2)$$

where $lemma$ is the lemmatization operator (Dai et al. 2021). We then use the gating mechanism to inject the knowledge retrieved from the knowledge base to plain word embeddings:

$$\mathbf{g}_i = gate_i \cdot \mathbf{p}_i^{wr} + (1 - gate_i) \cdot \mathbf{p}_i^{kn}, \quad (3)$$

where $gate_i \in [0, 1]$ is the gating coefficient. $\mathbf{p}_i^{wr} = \mathcal{E}_{wr}(p_i)$ is the passage word embeddings. $\mathbf{g}_i \in \mathbb{R}^h$ is the embeddings with knowledge injected. Let $\mathbf{G} = \mathbf{g}_1\mathbf{g}_2, \cdots, \mathbf{g}_n$. The input embeddings to the PLM can be written as:

$$\mathbf{x} = [\mathcal{E}_{wr}(\texttt{[CLS]}); \mathbf{Q}; \mathcal{E}_{wr}(\texttt{[SEP]}); \mathbf{G}; \mathcal{E}_{wr}(\texttt{[SEP]})]. \quad (4)$$

Passage-to-Prompt Attention (P2PA) The goal of P2PA is to enhance the representations $\mathbf{Q}$ of prompt tokens by the interaction between them and the passage representations. Specifically, we utilize self-attention to obtain the soft embeddings from the passage for each prompt token, and encourage the discreteness of the embeddings by GRUs (Cho et al. 2014). As discovered by Liu et al. (2020); Zhang et al. (2021), injecting too much background knowledge to PLM representations 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 randomly 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 $\mathbf{v}^{sp} \in \mathbb{R}^{r \times h}$:

$$\mathbf{v}^{sp} = \alpha\mathbf{G}, \quad (5)$$

$$\alpha = \sigma(\mathbf{q}^{sp}\mathbf{W}_\alpha\mathbf{G}), \quad (6)$$

where $\mathbf{W}_\alpha \in \mathbb{R}^{h \times h}$ is the trainable matrix. σ denotes the softmax function. We add residual connection to $\mathbf{v}^{sp}$ and $\mathbf{q}^{sp}$ by linear combination as $\mathbf{u}^{sp} = \mathbf{v}^{sp} + \mathbf{q}^{sp}$. Finally, as the embeddings of prompt tokens should be dependent on each other, we choose recurrent models to encourage the discreteness of the embeddings. Different from P-tuning (Liu et al. 2021), we choose GRUs rather than LSTMs due to fewer parameters. Formally, we have:

$$\overrightarrow{\mathbf{h}_j} = \overrightarrow{GRU}(\overrightarrow{\mathbf{h}_{j-1}}, \mathbf{u}_j^{sp}), \quad (7)$$

$$\overleftarrow{\mathbf{h}_j} = \overleftarrow{GRU}(\overleftarrow{\mathbf{h}_{j+1}}, \mathbf{u}_j^{sp}), \quad (8)$$

$$\mathbf{h}^{sp} = [\overrightarrow{\mathbf{h}_j}; \overleftarrow{\mathbf{h}_j}], \quad (9)$$

where $\mathbf{h}^{sp}$ is the knowledge-enhanced embedding of the selected prompt token. We replace the original word embeddings of selected prompt tokens $\mathbf{q}^{sp}$ with $\mathbf{h}^{sp}$. Therefore, the novel input denoted as $\mathbf{x}'$. We feed $\mathbf{x}'$ into MLM to obtain the logit values of all the `[MASK]` tokens.

3.4 Training Schemes of *KnowPrompt*

In prompt-based text classification, the verbalizer is defined to map the original labels to the words in the vocabulary set (Gao, Fisch, and Chen 2021). However, it is not suitable for extractive QA because of different searching spaces and the constrain that the answer must be the sub-string of the passage. Consider that MLM is a non-autoregressive model which ignores the dependence between each token. Here, we propose two training schemes to fine-tune the PLM.

Passage Masked Training Scheme (PaMT) We propose the PaMT to encourage the PLM to learn the contextual semantics of the answer. Specifically, we mask the answer tokens [4] in the passage and replace the token representations by `[MASK]` embeddings after calculating Equal 9. Hence, we can obtain the logit values from the MLM head to compute the Averaged Cross Entropy (ACE) loss:

$$\mathcal{L}_{ACE} = -\frac{1}{l}\sum_{i=1}^{l} \log \Pr(y_i|P, Q; \Theta). \quad (10)$$

where Θ is the collection of all trainable parameters.

Prompt Masked Training Scheme (PrMT) In contrastive to traditional text generation, the searching space of each `[MASK]` token in Q_{prompt} is constrained by previous predictions. 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. In order to satisfy this condition, we define two kinds of vocabulary sets: the *passage vocab* $\mathcal{V}^P$ and the *next vocab* $\mathcal{V}_t^P$, where $\mathcal{V}^P$ denotes the set of tokens that appear in the passage P, and $\mathcal{V}_t^P$ denotes the set of tokens that appear in the passage P and occur to the next of token t. Also take Figure 2 as an example, suppose P ="$\cdots$ Normandy had been exporting fighting horsemen for more than$\cdots$". "horsemen" is in $\mathcal{V}^P$ and $\mathcal{V}_{fighting}^P$. The sizes of the two vocabulary sets are: $N^P = |\mathcal{V}^p \cup \{\texttt{[END]}\}|$ and $N_t^P = |\mathcal{V}_t^P \cup \{\texttt{[END]}\}|$.

Intuitively, tokens with larger searching spaces are harder to predict. Therefore, when given the ground-truth answer, we assign different weights for each `[MASK]` token by:

$$\beta_i = \frac{\exp\{N_{y_{i-1}}^P/T\}}{\sum_j^l \exp\{N_{y_{j-1}}^P/T\}}, \quad (11)$$

where T is the smoothing factor. $i, j \in 1, 2, \cdots, l+1$, y_{l+1} is `[END]` and $N_{y_0}^P$ equals to N^P. Thus, the Weighted Cross Entropy (WCE) loss is defined as:

$$\mathcal{L}_{WCE} = -\sum_{i=1}^{l+1} \beta_i \log \Pr(y_i|P, Q; \Theta), \quad (12)$$

Additionally, in the preliminary experiments, we find that the model may generate incorrect answers which are also extracted from the passage and similar to the correct answers in semantics. Inspired by contrastive learning (Chen

[4] For the unanswerable sample, the loss in PaMT is zero.

et al. 2020; Qiu, Zhang, and Zhou), we can distinguish between the positive and negative predictions and alleviate the confusion problem. Here, we introduce a novel Contrastive Cross Entropy (CCE) loss. Specifically, we first use beam search to recall top-S predicted answers without the ground truth, and denote them as $\mathcal{Y}_b$[5]. Next, we randomly select S' sub-strings from the passage (which are not the ground truth), and denote them as $\mathcal{Y}_r$. For each negative answer $Y_i' = y_{i1}', y_{i2}', \cdots$, we compute its score by the weighted sum of all tokens probabilities, i.e.,

$$s_i' = \sum_j \beta_j \Pr(y_{ij}'|P, Q; \Theta), \tag{13}$$

where β_j can be computed by Equal 11. We can also obtain the score s of the ground truth in the same way. Hence, the CCE loss can be calculated as:

$$\mathcal{L}_{CCE} = -\frac{1}{|\mathcal{Y}_b \cup \mathcal{Y}_r|} \sum_{Y_i' \in \mathcal{Y}_b \cup \mathcal{Y}_r} \log[\frac{\exp\{s\}}{\exp\{s\} + \exp\{s_i'\}}], \tag{14}$$

Finally, the total loss function is written as follows:

$$\mathcal{L} = \mathcal{L}_{WCE} + \lambda \mathcal{L}_{ACE} + \mu \mathcal{L}_{CCE} + \gamma ||\Theta||, \tag{15}$$

where $\lambda \in [0, 1]$ and $\mu \in [0, 1]$ are pre-defined hyper-parameters balancing the three loss functions, γ is the regularization hyper-parameter that chosen by cross-validation.

3.5 Model Inference of *KnowPrompt*

In the model inference stage, we omit the passage masking process, and only generate the answer for the query prompt. Specifically, as shown in Figure 2, we utilize beam search to generate the top-S results from the passage through the *passage vocab* $\mathcal{V}^P$ and the *next vocab* $\mathcal{V}_t^P$. We calculate each candidate answer score by Equal 13, and choose the highest one as the final prediction.

4 Experiments

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

4.1 Baselines

As *KnowPrompt* is a general prompt-tuning framework, we choose BERT (Devlin et al. 2019) as our model backbone. To the best knowledge, *Splinter* (Ram et al. 2021) is state-of-the-art for few-shot extractive QA. Following their work, we consider the following methods as strong baselines:

- **BERT** (Devlin et al. 2019) utilizes two softmax heads to predict the start and the end positions of the answer span from the passage.
- **RoBERTa** (Liu et al. 2019) is the optimized version to improve the effectiveness of BERT.
- **SpanBERT** (Joshi et al. 2020) utilizes the span masking strategy and predicts the masked tokens based on boundaries representations.
- **Splinter** (Ram et al. 2021) is the first to regard span selection as a pre-training task for few-shot extractive QA.

[5]If the ground truth is present in $\mathcal{Y}_b$, we simply remove it from the collection.

Dataset	#Train	#Dev	#All
SQuAD2.0	118446	11873	130319
SQuAD	86588	10507	97095
NewsQA	74160	4212	78372
TriviaQA	61688	7785	69573
SearchQA	117384	16980	134364
HotpotQA	72928	5904	78832
NQ	104071	12836	116907

Table 1: The statistics of multiple benchmarks.

4.2 Benchmarks

Our framework is evaluated over two benchmarks, Table 1 shows the statistics of each dataset:

SQuAD 2.0 [6] (Rajpurkar, Jia, and Liang 2018): It is a widely used extractive QA benchmark, combining 43k unanswerable examples with original 87k answerable examples in SQuAD1.1. As the testing set is not publicly available, we use the public development set for evaluation.

MRQA 2019 [7] (Fisch et al. 2019): It is a shared task containing six extractive QA datasets formed in a unified format: SQuAD (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). Follow Ram et al. (2021), we use the subset of Split I, where the training set is used for training and the official development set is used for evaluation.

4.3 Experimental Settings

KnowPrompt is a general framework that can be used in a variety of few-shot learning settings:

Instance-level Few-shot Learning. Follow the same settings as in Ram et al. (2021), given an extractive QA dataset $\mathcal{D} = (\mathcal{D}^{train}, \mathcal{D}^{dev}, \mathcal{D}^{test})$, we randomly choose K samples from $\mathcal{D}^{train}$ and $\mathcal{D}^{dev}$ to construct a few-shot training set $\mathcal{D}_K^{train}$ and a development set $\mathcal{D}_K^{dev}$, respectively. In this case, we train our framework on $\mathcal{D}_K^{train}$ and choose the best model based on $\mathcal{D}_K^{dev}$. We evaluate the results on the whole testing set $\mathcal{D}^{test}$. All the tasks from SQuAD2.0 and MRQA are independently used in this setting.

Task-level Few-shot Learning. Given M similar few-shot extractive QA tasks, we randomly select $M-1$ training tasks and the remaining one as the target task [8]. For each training task, we randomly sample K training and development samples, respectively. We train and develop on $K \times (M-1)$ samples, respectively, and then evaluate it on the target testing task. Only six tasks from MRQA are used in this setting.

In our experiments, the underlying PLM and the default hyper-parameters are from the HuggingFace transformers [9]. We train our model by the Adam algorithm (Kingma and Ba

[6]Data url: https://rajpurkar.github.io/SQuAD-explorer/

[7]Data url: https://github.com/mrqa/MRQA-Shared-Task-2019

[8]Note that in this work, we simply regard one extractive QA dataset as a task. Knowledge transfer across significantly different tasks in the few-shot learning setting is highly challenging and is left for future work.

[9]Url: https://huggingface.co/transformers/index.html. We use the base version of PLMs in all the experiments.

Methods	SQuAD2.0	SQuAD*	NewsQA*	TriviaQA*	SearchQA*	HotpotQA*	NQ*	Avg.
Instance-level Few-shot Learning ($K = 16$)								
BERT (Devlin et al. 2019)	8.97	10.70	5.67	10.80	9.52	10.40	16.98	10.43
RoBERTa (Liu et al. 2019)	9.55	12.50	6.24	12.00	11.87	12.05	20.88	12.16
SpanBERT (Joshi et al. 2020)	9.90	12.50	6.00	12.80	13.00	12.60	19.70	12.36
Splinter (Ram et al. 2021)	53.05	54.60	20.80	18.90	26.30	24.00	27.40	32.15
KnowPrompt	**75.79**	**67.25**	**28.47**	**24.95**	**35.28**	**28.04**	**31.90**	**41.67**
Task-level Few-shot Learning ($K = 16$)								
BERT (Devlin et al. 2019)	-	31.10	11.57	10.55	15.38	11.88	21.18	16.94
RoBERTa (Liu et al. 2019)	-	33.98	12.29	12.50	21.05	16.09	26.20	20.35
SpanBERT (Joshi et al. 2020)	-	34.70	14.00	23.79	19.00	22.95	29.73	24.03
Splinter (Ram et al. 2021)	-	61.08	38.15	39.90	44.97	49.57	35.90	44.93
KnowPrompt	-	**68.22**	**40.88**	**42.65**	**44.10**	**54.89**	**43.82**	**49.09**

Table 2: The instance-level and task-level few-shot testing results of *KnowPrompt* and baselines in terms of F1 (%). * refers to the tasks of MRQA (Fisch et al. 2019).

2015). The learning rate for MLM is fixed as 1e-5, while the initial learning rate for other new modules in *KnowPrompt* is set in {1e-5, 3e-5, 5e-5, 1e-4} with a warm-up rate of 0.1 and an L2 weight decay of 0.01. The gating coefficient in PKI is 0.5. The smoothing factor T in PrMT is 100. The balance hyper-parameters are set as $\lambda = \mu = 0.5$. The number of `[MASK]` in query prompt is set to 10. The beam size of negative sampling in PrMT and inference stage are 5 and 3, respectively. In few-shot settings, the definition scope of sample number is $K \in \{16, 32, 64, \cdots, 1024\}$. We set the batch size and epoch number as 8 and 64, respectively. The F1 metric is used for evaluation, which measures the average overlap between the predicted and the ground-truth answer texts at the token level. During experiments, we choose five different random seeds $\{12, 21, 42, 87, 100\}$ (Gao, Fisch, and Chen 2021) and report the averaged performance.

4.4 Main Results

The results of *KnowPrompt* and all the baselines are in Table 2. The results indicate that *KnowPrompt* outperforms all baselines in both instance-level and task-level few-shot learning. i) In the instance-level few-shot learning, both the span-based selection pre-trained *Splinter* (Ram et al. 2021) and our *KnowPrompt* outperform baseline PLMs, which shows that it is vital to bridge the gap between pre-training and fine-tuning objectives. ii) Surprisingly, we achieve a 75.79% F1 value on SQuAD2.0 based on only 16 examples, which outperforms the state-of-the-art method *Splinter* (Ram et al. 2021) by 22.74%. We have also got a substantial improvement in a series of MRQA shared tasks, showing that prompt-tuning based on MLM generation is more suitable for extractive QA. iii) In the task-level few-shot learning, the performance gains of *KnowPrompt* over all six sets of tasks in MRQA (Fisch et al. 2019) are consistent, which demonstrates our framework possesses better generalization and transferability across similar extractive QA tasks.

4.5 Model Analysis

Effects on the Number of Training Samples. In this part, we further explore the model effects when the number of training samples varies. Figure 3 shows the performance with the different numbers of training samples over the SQuAD2.0 task. Here, each point refers the averaged score across 5 random sampled datasets. We observe that the proposed *KnowPrompt* achieves the highest scores regardless of the number of training samples. Specifically, when given 16 and more samples, we outperform all baselines by a large margin. When the models are trained on the whole training set, the performance is also improved by a 1.04% F1 score than the state-of-the-art method *Splinter*. It indicates that it is more effective to narrow the differences between fine-tuning with pre-training by transforming the task into the prompt-based paradigm rather than re-defining the pre-training tasks.

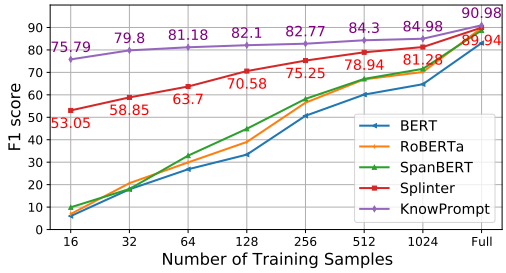

Figure 3: Performances (the F1 score) on different numbers of training samples over the SQuAD2.0 task.

Ablation Study. To further understand why *KnowPrompt* achieves high performance, we perform an ablation analysis to better validate the contributions of each component. For simplicity, we only present the ablation experimental results on SQuAD2.0 with 16, 1024 and all training samples.

We show all ablation experiments in Table 3, where w/o. PKI equals to set gating coefficient as 1. w/o. P2PA denotes only use the original passage embeddings. w/o. ACE and w/o. CCE equals to set $\lambda = 0$ and $\mu = 0$, respectively. w/o. WCE equals to set β_i in Equal 11 as $1/(l+1)$. We find that no matter which module is removed, the effect is decreasing. Particularly, when we remove P2PA, the performance is decreased by 17.69%, 12.81% and 6.36%, respectively. The declines of P2PA are larger than other cases, which indicates the significant impact of the passage-aware knowledge enhancement. We also find the CCE employed in this work also plays an important role in our framework, indicating

#Training Samples⟶	16	1024	All
KnowPrompt	**75.79**	**84.98**	**90.98**
w/o. PKI	73.36	82.53	90.70
w/o. P2PA	58.10	72.17	84.62
w/o. ACE	73.08	80.19	89.22
w/o WCE	73.96	82.11	89.58
w/o. CCE	66.45	74.25	86.40

Table 3: The ablation F1 scores of *KnowPrompt* for instance-level few-shot learning. w/o. denotes that we only remove one component from *KnowPrompt*.

Prompt Mapping	SQuAD2.0	NewsQA	HotpotQA
$\mathcal{T}_1$ (None)	89.08	72.40	79.12
$\mathcal{T}_2$ (Manual)	88.87	72.77	78.40
$\mathcal{T}$ **(Proposed)**	**90.98**	**73.90**	**81.37**

Table 4: Comparison with proposed prompt template mapping $\mathcal{T}$ with two alternative methods $\mathcal{T}_1$ and $\mathcal{T}_2$ (%).

that there are many confusing texts in the passage that need to be effectively distinguished by contrastive learning.

Effects of Different Prompt Templates. As for the findings by Gao, Fisch, and Chen (2021), different prompt template have different effects on the overall performance. To make a clear comparison, we replace our proposed query prompt mapping $\mathcal{T}$ with two alternative methods:

- $\mathcal{T}_1$ **(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]` $\cdots$").

We randomly select three tasks (e.g., SQuAD2.0 (Rajpurkar, Jia, and Liang 2018), NewsQA (Trischler et al. 2017) and HotpotQA (Yang et al. 2018)) and evaluate the templates with full training sets. As shown in Table 4, we find that two simple templates have the similar performance. Our proposed method outperforms them by more than 1.0% in terms of the F1 score.

To show the intuitive comparison, we randomly choose one case from HotpotQA (Yang et al. 2018). As shown in Figure 4, the text in blue denotes the correct prediction. It shows our template (which is directly converted from the original query text) performs better than others. We guess that `[MASK]` tokens embedded into the query can easily "find" similar context semantics (in red) in the passage. For example, based on proposed prompt template, the MLM succeeds in capturing corresponding semantics in the question context, and tends to recall all candidates related to people entities.

4.6 Limitation of Our Work: Difficulty of Answer Token Generation

A major difference between previous works and ours is that we model the extractive QA task as text generation. Hence, it is difficult but vital to generate tokens of the correct answer, due to the lack of the classification heads. 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. Table 5 reports the accuracy of predicting the first answer tokens on chosen tasks with full training samples. Here, we check whether the generated first token and the first token of the ground truth are within a fixed window size. In the experiments, we find the accuracy of our method is lower than BERT (Devlin et al. 2019) when $n = 1$. This is contradictory to the overall results where the F1 score of our method is higher than BERT. We increase the window size to 5 and thus obtain the best performance. We suggest that *KnowPrompt* can be further improved by applying controllable text generation, which will be left as future work.

Passage: 18th Independent Spirit Awards. The 2002 Independent Spirit Awards , honoring the best in independent film making for 2002, were announced on March 22, 2003. It was hosted by John Waters. John Waters (born April 22, 1946) is an American film director, screenwriter, author, actor, stand-up comedi-an, journalist, visual artist, and art collector, who rose to fame in the early 1970s for his transgressive cult films.
Query: Who hosted the 18th Independent Spirit Awards in 2002?

Query Prompt (Proposed): [MASK] [MASK] [MASK] [MASK] hosted the 18th Independent Spirit Awards in 2002.
Top1 Answer: John Waters
Top2 Answer: American film director
Top3 Answer: transgressive cult films

Query Prompt (None): Who hosted the 18th Independent Spirit Awards in 2002? [SEP] [MASK] [MASK] [MASK] [MASK] [SEP]
Top1 Answer: American film director
Top2 Answer: John Waters
Top3 Answer: in the early 1970s

Query Prompt (Manual): Who hosted the 18th Independent Spirit Awards in 2002? [SEP] The answer is [MASK] [MASK] [MASK] [MASK] [SEP]
Top1 Answer: independent film
Top2 Answer: by John Waters
Top3 Answer: cult films.

Figure 4: Sample comparison between different prompt templates. We remove the `[END]` terminator after generation.

Method	SQuAD2.0	NewsQA	HotpotQA
BERT (#1)	83.19	56.80	56.11
KnowPrompt (#1)	55.10	42.80	40.29
KnowPrompt (#3)	68.36	44.22	47.10
***KnowPrompt* (#5)**	**85.90**	**59.93**	**57.14**

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

5 Conclusion

To bridge the gap between the pre-training and fine-tuning objectives in extractive QA, *KnowPrompt* views extractive QA as an answer generation task. In *KnowPrompt*, the knowledge-aware prompt encoder injects external knowledge into the passage, and enhances the representations of query prompts. Two main training schemes with averaged, weighted and contrastive losses are proposed to train the model. Evaluation experiments on multiple benchmarks show that our approach outperforms the state-of-the-art methods. In the future, we will i) improve the performance of *KnowPrompt* and ii) explore the prompt-tuning paradigm for other types of MRC tasks, such as cloze-style MRC and multiple choices.

References

Brown, T. B.; Mann, B.; and Nick Ryder, e. 2020. Language Models are Few-Shot Learners. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *NeurIPS*.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. E. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, 1597–1607. PMLR.

Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*, 1724–1734. ACL.

Dai, D.; Zheng, H.; Sui, Z.; and Chang, B. 2021. Incorporating Connections Beyond Knowledge Embeddings: A Plug-and-Play Module to Enhance Commonsense Reasoning in Machine Reading Comprehension. *CoRR*, abs/2103.14443.

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. In *AAAI*, 1811–1818. AAAI Press.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*, 4171–4186. Association for Computational Linguistics.

Dunn, M.; Sagun, L.; Higgins, M.; Güney, V. U.; Cirik, V.; and Cho, K. 2017. SearchQA: A New Q&A Dataset Augmented with Context from a Search Engine. *CoRR*, abs/1704.05179.

Fisch, A.; Talmor, A.; Jia, R.; Seo, M.; Choi, E.; and Chen, D. 2019. MRQA 2019 Shared Task: Evaluating Generalization in Reading Comprehension. In *EMNLP*, 1–13. Association for Computational Linguistics.

Gao, T.; Fisch, A.; and Chen, D. 2021. Making Pre-trained Language Models Better Few-shot Learners. In *ACL/IJCNLP*, 3816–3830. Association for Computational Linguistics.

Han, X.; Zhao, W.; Ding, N.; Liu, Z.; and Sun, M. 2021. PTR: Prompt Tuning with Rules for Text Classification. *CoRR*, abs/2105.11259.

Jiang, Z.; Xu, F. F.; Araki, J.; and Neubig, G. 2020. How Can We Know What Language Models Know. *Trans. Assoc. Comput. Linguistics*, 8: 423–438.

Joshi, M.; Chen, D.; Liu, Y.; Weld, D. S.; Zettlemoyer, L.; and Levy, O. 2020. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *Trans. Assoc. Comput. Linguistics*, 8: 64–77.

Joshi, M.; Choi, E.; Weld, D. S.; and Zettlemoyer, L. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *ACL*, 1601–1611. Association for Computational Linguistics.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Kwiatkowski, T.; Palomaki, J.; Redfield, O.; Collins, M.; Parikh, A. P.; Alberti, C.; Epstein, D.; Polosukhin, I.; Devlin, J.; Lee, K.; Toutanova, K.; Jones, L.; Kelcey, M.; Chang, M.; Dai, A. M.; Uszkoreit, J.; Le, Q.; and Petrov, S. 2019. Natural Questions: a Benchmark for Question Answering Research. *Trans. Assoc. Comput. Linguistics*, 7: 452–466.

Lai, G.; Xie, Q.; Liu, H.; Yang, Y.; and Hovy, E. H. 2017. RACE: Large-scale ReAding Comprehension Dataset From Examinations. In *EMNLP*, 785–794. Association for Computational Linguistics.

Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; and Soricut, R. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR*. OpenReview.net.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. *CoRR*, abs/2104.08691.

Levy, O.; Seo, M.; Choi, E.; and Zettlemoyer, L. 2017. Zero-Shot Relation Extraction via Reading Comprehension. In *CoNLL*, 333–342. Association for Computational Linguistics.

Li, X. L.; and Liang, P. 2021a. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *ACL/IJCNLP*, 4582–4597. Association for Computational Linguistics.

Li, X. L.; and Liang, P. 2021b. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *ACL/IJCNLP*, 4582–4597. Association for Computational Linguistics.

Liu, W.; Zhou, P.; Zhao, Z.; Wang, Z.; Ju, Q.; Deng, H.; and Wang, P. 2020. K-BERT: Enabling Language Representation with Knowledge Graph. In *AAAI*, 2901–2908.

Liu, X.; Zheng, Y.; Du, Z.; Ding, M.; Qian, Y.; Yang, Z.; and Tang, J. 2021. GPT Understands, Too. *CoRR*, abs/2103.10385.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR*, abs/1907.11692.

Peng, W.; Hu, Y.; Yu, J.; Xing, L.; Xie, Y.; Zhu, Z.; and Sun, Y. 2021. MCR-Net: A Multi-Step Co-Interactive Relation Network for Unanswerable Questions on Machine Reading Comprehension. *CoRR*, abs/2103.04567.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep Contextualized Word Representations. In *NAACL-HLT*, 2227–2237. Association for Computational Linguistics.

Qin, G.; and Eisner, J. 2021. Learning How to Ask: Querying LMs with Mixtures of Soft Prompts. In *NAACL-HLT*, 5203–5212. Association for Computational Linguistics.

Qiu, D.; Zhang, Y.; Feng, X.; Liao, X.; Jiang, W.; Lyu, Y.; Liu, K.; and Zhao, J. 2019. Machine Reading Comprehension Using Structural Knowledge Graph-aware Network. In *EMNLP-IJCNLP*, 5895–5900. Association for Computational Linguistics.

Qiu, Y.; Zhang, J.; and Zhou, J. ???? Improving Gradient-based Adversarial Training for Text Classification by Contrastive Learning and Auto-Encoder. In *Findings of ACL: ACL/IJCNLP*, volume ACL 2021 of *Findings of ACL*, 1698–1707.

Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving language understanding by generative pre-training.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.*, 21: 140:1–140:67.

Rajpurkar, P.; Jia, R.; and Liang, P. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. *CoRR*, abs/1806.03822.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100, 000+ Questions for Machine Comprehension of Text. *CoRR*, abs/1606.05250.

Ram, O.; Kirstain, Y.; Berant, J.; Globerson, A.; and Levy, O. 2021. Few-Shot Question Answering by Pretraining Span Selection. In *ACL/IJCNLP*, 3066–3079. Association for Computational Linguistics.

Schick, T.; and Schütze, H. 2021a. Exploiting Cloze-Questions for Few-Shot Text Classification and Natural Language Inference. In *EACL*, 255–269. Association for Computational Linguistics.

Schick, T.; and Schütze, H. 2021b. It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners. In *NAACL-HLT*, 2339–2352. Association for Computational Linguistics.

Seo, M. J.; Kembhavi, A.; Farhadi, A.; and Hajishirzi, H. 2016. Bidirectional Attention Flow for Machine Comprehension. *CoRR*, abs/1611.01603.

Shin, T.; Razeghi, Y.; IV, R. L. L.; Wallace, E.; and Singh, S. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *EMNLP*, 4222–4235. Association for Computational Linguistics.

Speer, R.; Chin, J.; and Havasi, C. 2017. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. In *AAAI*, 4444–4451. AAAI Press.

Su, Y.; Cai, D.; Wang, Y.; Vandyke, D.; Baker, S.; Li, P.; and Collier, N. 2021. Non-Autoregressive Text Generation with Pre-trained Language Models. In Merlo, P.; Tiedemann, J.; and Tsarfaty, R., eds., *EACL 2021, Online, April 19 - 23, 2021*, 234–243. Association for Computational Linguistics.

Trischler, A.; Wang, T.; Yuan, X.; Harris, J.; Sordoni, A.; Bachman, P.; and Suleman, K. 2017. NewsQA: A Machine Comprehension Dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, 191–200.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer Networks. In *NIPS*, 2692–2700.

Wang, C.; and Jiang, H. 2019. Explicit Utilization of General Knowledge in Machine Reading Comprehension. In *ACL*, 2263–2272. Association for Computational Linguistics.

Wang, S.; and Jiang, J. 2017. Machine Comprehension Using Match-LSTM and Answer Pointer. In *ICLR*. OpenReview.net.

Wang, W.; Yang, N.; Wei, F.; Chang, B.; and Zhou, M. 2017. Gated Self-Matching Networks for Reading Comprehension and Question Answering. In *ACL*, 189–198.

Yang, A.; Wang, Q.; Liu, J.; Liu, K.; Lyu, Y.; Wu, H.; She, Q.; and Li, S. 2019a. Enhancing Pre-Trained Language Representations with Rich Knowledge for Machine Reading Comprehension. In *ACL*, 2346–2357. Association for Computational Linguistics.

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J. G.; Salakhutdinov, R.; and Le, Q. V. 2019b. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NeurIPS*, 5754–5764.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W. W.; Salakhutdinov, R.; and Manning, C. D. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *EMNLP*, 2369–2380. Association for Computational Linguistics.

Yu, A. W.; Dohan, D.; Luong, M.; Zhao, R.; Chen, K.; Norouzi, M.; and Le, Q. V. 2018. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. In *ICLR*. OpenReview.net.

Zhang, N.; Deng, S.; Cheng, X.; Chen, X.; Zhang, Y.; Zhang, W.; Chen, H.; and Center, H. I. 2021. Drop Redundant, Shrink Irrelevant: Selective Knowledge Injection for Language Pretraining. In *IJCAI*.

Zhang, Z.; Wu, Y.; Zhou, J.; Duan, S.; Zhao, H.; and Wang, R. 2020. SG-Net: Syntax-Guided Machine Reading Comprehension. In *AAAI*, 9636–9643. AAAI Press.

Zhang, Z.; Yang, J.; and Zhao, H. 2021. Retrospective Reader for Machine Reading Comprehension. In *AAAI*, 14506–14514. AAAI Press.

Zou, X.; Yin, D.; Zhong, Q.; Yang, H.; Yang, Z.; and Tang, J. 2021. Controllable Generation from Pre-trained Language Models via Inverse Prompting. In *KDD*, 2450–2460. ACM.