Knowledge Prompting in Pre-trained Language Model for Natural Language Understanding

Jianing Wang[1], **Wenkang Huang**[2], **Qiuhui Shi**[2], **Hongbin Wang**[2],
Minghui Qiu[3], **Xiang Li**[1]*, **Ming Gao**[1,4]
[1] School of Data Science and Engineering, East China Normal University, Shanghai, China
[2] Ant Group, Hangzhou, China [3] Alibaba Group, Hangzhou, China
[4] KLATASDS-MOE, School of Statistics, East China Normal University, Shanghai, China
lygwjn@gmail.com, wenkang.hwk@alibaba-inc.com
{qiuhui.sqh,hongbin.whb}@antgroup.com
minghui.qmh@alibaba-inc.com
{xiangli,mgao}@dase.ecnu.edu.cn

Abstract

Knowledge-enhanced Pre-trained Language Model (PLM) has recently received significant attention, which aims to incorporate factual knowledge into PLMs. However, most existing methods modify the internal structures of fixed types of PLMs by stacking complicated modules, and introduce redundant and irrelevant factual knowledge from knowledge bases (KBs). In this paper, to address these problems, we introduce a seminal knowledge prompting paradigm and further propose a knowledge-prompting-based PLM framework KP-PLM. This framework can be flexibly combined with existing mainstream PLMs. Specifically, we first construct a knowledge sub-graph from KBs for each context. Then we design multiple continuous prompts rules and transform the knowledge sub-graph into natural language prompts. To further leverage the factual knowledge from these prompts, we propose two novel knowledge-aware self-supervised tasks including prompt relevance inspection and masked prompt modeling. Extensive experiments on multiple natural language understanding (NLU) tasks show the superiority of KP-PLM over other state-of-the-art methods in both full-resource and low-resource settings [1].

1 Introduction

Pre-trained Language Models (PLMs) have become the dominant infrastructure for a majority of downstream natural language understanding (NLU) tasks, which generally adopt a two-stage training strategy, i.e., *pre-training* and *fine-tuning*. Recent notable PLMs include BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2020), XLNet (Yang et al., 2019), GPT-2 (Radford et al., 2019) and DeBERTa (He et al., 2021b). Although many existing models learn useful inherent *linguistic knowledge* from large-scale corpora, it is hard for them to understand the explicit *factual knowledge* (IV et al., 2019; Sun et al., 2020).

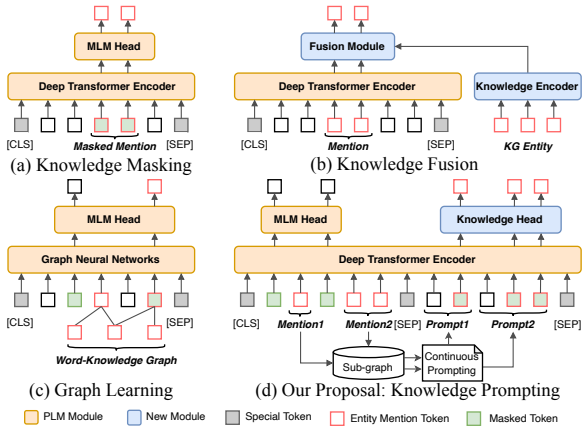

Figure 1: The comparison between knowledge prompting with other paradigms. (Best viewed in color.)

To further leverage factual knowledge, a branch of knowledge-enhanced methods (Zhang et al., 2019; Sun et al., 2020; He et al., 2021a; Wang et al., 2021b,a; Zhang et al., 2021; Arora et al., 2022) have been proposed for PLMs to capture rich semantic knowledge from knowledge bases (KBs). Figure 1 summarizes the paradigms of existing approaches, which can be mainly divided into three categories. First, *knowledge-masking-based methods* perform alignment, mask mentions and learn explicit knowledge of entities based on masked language modeling (MLM). Second, *knowledge-fusion-based methods* separately learn embeddings of entities in sentences and that of

* Correspondence to Xiang Li.
[1] All the codes and datasets have been released to https://github.com/wjn1996/KP-PLM

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 3164–3177
December 7-11, 2022

nodes in KBs, which are further aggregated. Third, *graph-learning-based methods* construct a graph based on contextualized sentences and KBs, and adopt a graph-based encoder to learn entity embeddings, such as graph neural networks (Kipf and Welling, 2017). Despite the success, there could still be two problems in these methods. On the one hand, some approaches modify internal structures of existing PLMs by stacking complicated modules, which adversely affects the computational cost of the model. Further, these methods are generally based on fixed types of PLMs, which leads to the inflexibility and restricts their wide applicability (He et al., 2021a; Zhang et al., 2021). On the other hand, some methods introduce redundant and irrelevant knowledge from KBs, which are the knowledge noises and could degrade the model performance (Peters et al., 2019): 1) Redundant Knowledge. Previous methods (Sun et al., 2020; Liu et al., 2020; Sun et al., 2019; Wang et al., 2021b) inject the corresponding triples or pre-trained knowledge embeddings into each entity in the context. However, the entity may appear multiple times in one sentence, these methods could introduce duplicate information, which can be viewed as redundant knowledge. 2) Irrelevant Knowledge. Some entities or the corresponding sub-graphs are irrelevant with the whole sentence semantics, which are useless and have less contributions to the performance improvement.

Recently, the paradigm of prompt-based fine-tuning (i.e., prompt-tuning) has been proposed to explore the inherent knowledge of PLMs by introducing a task-specific template with the `[MASK]` token (Schick and Schütze, 2021)[2]. Inspired by prompt-tuning, in this paper, we introduce a novel *knowledge prompting* paradigm for knowledge-enhanced PLMs and propose an effective **K**nowledge-**P**rompting-based **PLM** framework, namely, KP-PLM. As shown in Figure 1(d), we aim to construct the related knowledge subgraph for each sentence. To alleviate introducing too much knowledge noises, we restrict all the relation paths start from the topic entity (the entry of each sentence) and end at the tail entity which is mentioned in this sentence. To bridge the structure gap between the context and KB, we can transform each relation path into a natural language prompt, and then concatenate these prompts with the original sentence to form a unified input sequence, which can be flexibly fed into any PLMs without changing their internal structures (Brown et al., 2020; Schick and Schütze, 2021). Further, to leverage the factual knowledge in the prompts, we propose two novel knowledge-aware self-supervised learning tasks: *Prompt Relevance Inspection* (PRI) and *Masked Prompt Modeling* (MPM). Specifically, PRI encourages the PLM to learn the semantic relevance of multiple knowledge prompts, while MPM aims to predict the masked entity in a prompt. We conduct extensive experiments to verify the effectiveness of KP-PLM on multiple NLU tasks. Our results show that KP-PLM can be flexibly integrated with mainstream PLMs and the integrated model can outperform strong baselines in both full-resource and low-resource settings. We next summarize our main contributions as follows:

- We propose a novel knowledge prompting paradigm to incorporate factual knowledge into PLMs.
- We present the KP-PLM framework that can be flexibly combined with mainstream PLMs.
- We design two knowledge-aware self-supervised tasks to learn factual knowledge from prompts.
- We conduct extensive experiments to show the effectiveness of KP-PLM in both full-resource and low-resource scenarios.

2 Related Work

Knowledge-enhanced PLMs. To further inject factual knowledge into PLMs, recent knowledge-enhanced PLMs have been proposed to incorporate factual knowledge in the pre-training stage (Sun et al., 2019; Xiong et al., 2020; Liu et al., 2020; Sun et al., 2020; Zhang et al., 2019; He et al., 2021a; Zhang et al., 2021; Su et al., 2021; Arora et al., 2022; Ye et al., 2022; Yu et al., 2022a,b; Chen et al., 2022; Liu et al., 2021c). For example, ERNIE-Baidu (Sun et al., 2019) introduces two novel phrase-level masking and entity-level masking strategies to learn explicit semantic knowledge. K-BERT (Liu et al., 2020) and CoLAKE (Sun et al., 2020) utilize contextualized sentences and KBs to construct a graph, based on which graph-based

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

learning methods are employed to capture semantic information. In addition, ERNIE-THU (Zhang et al., 2019), KEPLER (Wang et al., 2021b) and KLMo (He et al., 2021a) integrate the pre-trained knowledge base embeddings into PLMs through attentive fusion modules to improve the contextual representations. Different from these methods, we propose a novel knowledge prompting paradigm to enhance PLMs, which learns factual knowledge from prompts.

Prompting for PLMs. In the fine-tuning stage, traditional fine-tuning paradigms introduce new parameters for task-specific predictions, which could lead to the over-fitting problem in low-resource scenarios (Brown et al., 2020; Schick and Schütze, 2021). To address the issue, prompt-tuning has recently been proposed (Brown et al., 2020; Schick and Schütze, 2021; Gao et al., 2021a; Liu et al., 2021b). For example, GPT-3 (Brown et al., 2020) proposes to enable in-context learning with handcraft prompts in zero-shot scenarios. In addition, Schick and Schütze (2021) and Gu et al. (2021) explore all the tasks in a unified cloze-style format, which boosts the performance of PLMs in few-shot learning tasks. There are also recent works that improve prompt-tuning based on heuristic rules (Han et al., 2021), automatic prompt construction (Gao et al., 2021a; Shin et al., 2020) and continuous prompt learning (Liu et al., 2021b,a; Hu et al., 2021; Gu et al., 2021). Inspired by prompt-tuning, we inject factual knowledge into PLMs by designing multiple continuous prompts.

3 The KP-PLM Framework

In this section, we first describe the basic notations, and then formally present the techniques of the KP-PLM in detail.

3.1 Notations

A knowledge graph is denoted as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$. Here, $\mathcal{E}$ is a set of entities, $\mathcal{R}$ is a set of relations, and $\mathcal{T}$ is a set of triples to express semantic knowledge. Specifically, $\mathcal{T} = \{(e_h, r, e_t)|e_h, e_t \in \mathcal{E}, r \in \mathcal{R}\}$, where e_h, r and e_t denote head entity, relation and tail entity, respectively. Given a knowledge graph $\mathcal{G}$, let $(e_0, \{(r_i, e_i)\}_{i=1}^{k})$ be a k-hop relation path starting from entity e_0 to e_k in $\mathcal{G}$. We further denote $\mathcal{V}$ as the vocabulary set used in PLMs and $S = \{w_1, w_2, \cdots\}$ as a sentence, where $w_i \in \mathcal{V}$ is the i-th token of the sentence.

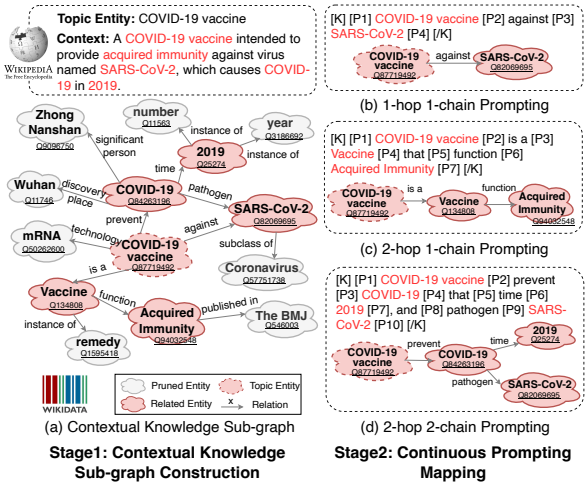

Figure 2: The illustration on knowledge prompting. We first construct a 2-hop sub-graph for each context based on target entity, and then generate continuous prompts based on three kinds of mapping formats. (Best viewed in color.)

3.2 Knowledge Prompting

Knowledge prompting aims to construct the knowledge sub-graph for each sentence, and then transform factual knowledge into natural language prompts. Figure 2 illustrates the process in two steps.

Contextual Knowledge Sub-graph Construction. Most existing methods integrate knowledge from KBs indiscriminately with all the mentions in the original context, which could bring in redundant and irrelevant information to PLMs (Zhang et al., 2021). Generally, a pre-training context is derived from the encyclopedia (e.g. Wikipedia), it is usually tagged with an entry that describes the topic of the context and can be viewed as the *topic entity*. Intuitively, the sub-graph centering at the topic entity in KBs could contain semantic knowledge that is highly correlated with the context. Therefore, we propose to construct a KB sub-graph for each context based on the corresponding topic entity.

We first scan all the entries in Wikipedia Dumps and derive their corresponding texts, which form a large-scale corpus. After that, for each sentence S, we utilize the TAGME toolkit (Ferragina and Scaiella, 2010) to generate M_S, a set of entity mentions in the sentence. Then, for each sentence S and its associated topic entity e_S, we generate a 2-hop sub-graph $\mathcal{G}_S$ for S, which centers at e_S and includes all the k-hop relation paths starting from e_S in $\mathcal{G}$. Here, $k \leq 2$. However, this will lead to identical sub-graphs for various sentences with the

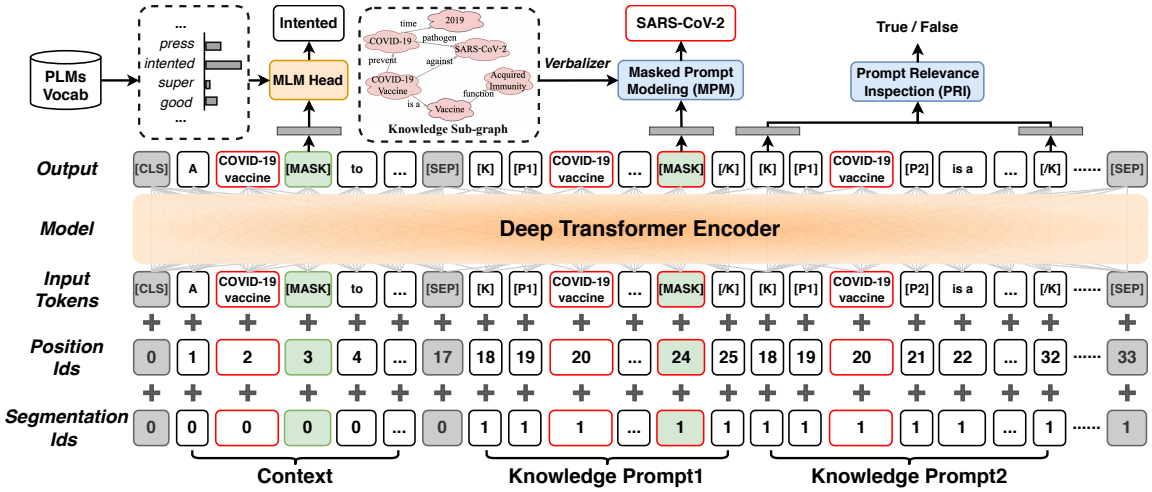

Figure 3: The model architecture of KP-PLM framework. (Best viewed in color.)

same topic entity. To distinguish these sentences and filter irrelevant knowledge from the sub-graph, we propose to further prune $\mathcal{G}_S$. The procedure can be summarized as follows. We first traverse all the 2-hop relation paths in $\mathcal{G}_S$. For each 2-hop relation path $(e_S, r_1, e_1, r_2, e_2)$, if the tail entity e_2 is not mentioned in the sentence, we remove e_2 and all the relations linked to it from $\mathcal{G}_S$. After that, we further traverse all the 1-hop relation paths in the pruned sub-graph. For each 1-hop relation path (e_S, r_1, e_1), if e_1 is not mentioned in the sentence and there are not any 2-hop relation paths that contain e_1, we remove e_1 and all the linked relations from the sub-graph. For example, as shown in Figure 2 (a), while the entity "Vaccine" is not a mention in the sentence, we retain it due to its inclusion in the 2-hop relation path *"(COVID-19 Vaccine, (is a, Vaccine), (function, Acquired Immunity))"*. Finally, we take the generated pruned sub-graph as the contextual knowledge sub-graph $\hat{\mathcal{G}}_S$ for S.

Continuous prompting mapping. After $\hat{\mathcal{G}}_S$ is constructed, we next extract semantic knowledge from it. While there are some methods (Yao et al., 2019; Sun et al., 2020) that directly convert relation triples into discrete texts, it has been pointed out in (Liu et al., 2021b) that the optimization in these methods could suffer from local minima. Inspired by (Liu et al., 2021b), we employ continuous prompts to inject factual knowledge into PLMs, which have been shown to be effective in capturing semantic knowledge (Gu et al., 2021).

Specifically, we design three types of prompt mapping rules based on the first-order and second-order structural information in $\hat{\mathcal{G}}_S$. As shown in Figure 2, the 1-hop 1-chain sub-structure is a single triple, the 2-hop 1-chain sub-structure denotes a 2-hop relation path and the 2-hop 2-chain sub-structure represents two 2-hop relation paths that share the same prefix triple. These three types of sub-structures are the building blocks of other complex and higher-order ones, so they are adopted to design prompts. Our framework can also be easily incorporated with other sub-structures. For each type of sub-structure, we design continuous templates with knowledge trigger tokens `[K]`, `[/K]` and pseudo tokens `[Pi]`. Examples of prompt templates are given in Figure 2 (b)-(d).

Finally, we concatenate all the derived prompts with the original context to form an input sequence $X = \texttt{Concat}(S, \{P_i\}_{i=1}^m)$, where P_i denotes the i-th prompt and m is the number of the prompts.

3.3 Model Architecture

After the input sequence X is generated, we first utilize the PLM tokenizer to transform X into a token sequence $\{x_j\}_{j=1}^n$, where x_j is the j-th token and n is the sequence length. For notation simplicity, we overload $X = \{x_j\}_{j=1}^n$. Then we feed X into a PLM, whose architecture is shown in Figure 3. We next describe the two major components.

Input Embedding Layer. Similar as previous methods (Liu et al., 2020; Sun et al., 2020), there are three types of embeddings including token embeddings, position embeddings and segmentation embeddings. For token embeddings, the trigger and pseudo tokens are randomly initialized while others are initialized by looking up the PLM embedding table. To alleviate the influence of the prompt order on the model performance, we use

the same segmentation id for all the prompts and set the same position id to their start tokens. For example, in Figure 3, the segmentation ids of all the prompts are set to 1 and the position ids of their start token `[K]` are uniformly set to 18.

Deep Transformer Encoder. Following (Devlin et al., 2019; Liu et al., 2019), we use a multi-layer transformer encoder (Vaswani et al., 2017) to capture the semantics from both context and knowledge prompts. To further mitigate the model's sensitivity to the prompt order, we design a mask matrix $\mathbf{M}$ for self-attention in PLMs, whose (i, j)-th entry characterizes the relation between two tokens $x_i, x_j \in X$, and is formally defined as:

$$\mathbf{M}_{ij} = \begin{cases} -\inf & x_i \in P_u \wedge x_j \in P_v \wedge u \neq v; \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The self-attention in the transformer can then be calculated as:

$$\text{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \texttt{Softmax}\big(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}} + \mathbf{M}\big)\mathbf{V}, \tag{2}$$

where $d > 0$ is the scale factor and $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times h}$ are the query, key and value matrices, respectively. Here, $\mathbf{M}_{ij} = -\inf$ enforces that the attention weight between x_i and x_j from different prompts will be set to 0.

3.4 Self-supervised Pre-training Tasks

We next provide the detailed description on two self-supervised pre-training tasks.

Prompt Relevance Inspection (PRI). Since prompts can inject factual knowledge into PLMs, they are expected to be semantically relevant to the context sequences. Therefore, we design a novel prompt relevance inspection task, which enhances the model's capability in learning the relevance of a prompt to a sentence. For each sentence S in the training corpus, the knowledge prompting process (see Section 3.2) can generate a set of relevant prompts $\mathcal{P}_S = \{P_i\}_{i=1}^m$ to S. We then construct a "positive" prompt set `Pos` by randomly selecting a prompt from $\mathcal{P}_S$ for each sentence S in the corpus. Further, we need to construct a "negative" prompt set `Neg`. For each sentence S, we randomly select a prompt P from $\mathcal{P}_S$ and replace a randomly selected entity in P by an arbitrary entity in KBs. The updated prompt is then labeled as a negative prompt and added to `Neg`. We repeat the above process if more negative samples are needed.

After `Pos` and `Neg` are generated, we can construct a training set $\mathcal{D}_1 = \{(P, y_P)\}$, where $y_P = 1$ if $P \in$ `Pos`; 0, otherwise. For each sample (P, y_P) from $\mathcal{D}_1$, inspired by SpanBERT (Joshi et al., 2020), we represent P by using its two boundary tokens (`[K]` and `[/K]`). Let $\mathbf{x}_{\texttt{[K]}}, \mathbf{x}_{\texttt{[/K]}} \in \mathbb{R}^h$ be input embeddings, respectively. Formally, we have:

$$\mathbf{h}_P = \mathcal{LN}(\sigma((\mathcal{F}_\theta(\mathbf{x}_{\texttt{[K]}}) + \mathcal{F}_\theta(\mathbf{x}_{\texttt{[/K]}}))\mathbf{W}_1)), \tag{3}$$

where $\mathbf{W}_1 \in \mathbb{R}^{h \times h}$ is the trainable parameter matrix, $\sigma(\cdot)$ and $\mathcal{LN}(\cdot)$ are the Sigmoid and LayerNorm functions, respectively. Further, $\mathcal{F}_\theta(\cdot)$ is the output representation by the PLM and θ denotes the model parameters. Based on $\mathbf{h}_P$, we define a binary classifier whose objective function is:

$$\mathcal{L}_{\text{PRI}} = \mathbb{E}_{(P, y_P) \sim \mathcal{D}_1}[\log \Pr_\Phi(y_P|P)], \tag{4}$$

$$\Pr_\Phi(y_P|P) = \texttt{Softmax}(\mathcal{H}_1(\mathbf{h}_P)), \tag{5}$$

where $\mathcal{H}_1$ is the classification head with parameter Φ and $\Pr(\cdot)$ denotes the probability distribution.

Masked Prompt Modeling (MPM). We further propose a masked prompt modeling task, which aims to predict the masked entity in a prompt. Different from the vanilla masked language modeling (Devlin et al., 2019) that has an enormous search space over the whole PLM vocabulary $\mathcal{V}$, our task shrinks the search space for a prompt $P \in \mathcal{P}_S$ to the entity set of the contextual knowledge sub-graph $\hat{\mathcal{G}}_S$.

Given a training corpus, for each context S, we randomly mask an entity e_P (except the topic entity[3]) with the `[MASK]` token in an arbitrarily selected prompt $P \in \mathcal{P}_S$. Then we can generate a dataset $\mathcal{D}_2 = \{(P, e_P)\}$ that consists of all the (prompt, masked entity) pairs. To enforce the model to better learn factual knowledge expressed by the generated prompts, we employ contrastive learning (Jean et al., 2015) techniques and formulate an objective function as:

$$\mathcal{L}_{\text{MPM}} = \mathbb{E}_{(P, e_P) \sim \mathcal{D}_2} \left[\frac{\exp(g(e_P|P))}{\exp(g(e_P|P)) + N\mathbb{E}_{e'_P \sim q(e)}\left[\exp(g(e'_P|P))\right]} \right], \tag{6}$$

where $e'_P \neq e_P$ is a negative entity, N is the total number of sampled negative entities[4], and $q(\cdot)$ is a

[3] The topic entity is fixed at the start position in all the relation paths. We do not mask it because it is easy for PLMs to predict based on other knowledge prompts.

[4] If $|\mathcal{E}_S| < N + 1$, the remaining $N - |\mathcal{E}_S| + 1$ negative entities can be sampled from the whole $\mathcal{E}$.

sampling function implemented by the PageRank algorithm (Gao et al., 2021b), which is used to calculate the sampling probability scores for all the entities in $\mathcal{E}_S$. Further, $g(\cdot)$ is a scoring function that measures the similarity between the masked token representation $\mathbf{h}_{\texttt{[MASK]}}$ and the entity e_P. In particular, $g(\cdot)$ is expected to effectively learn the representation of the masked entity e_P by pulling embeddings of all its mentions in contexts together and pushing apart that of other negative entities. Therefore, we first introduce a verbalizer mapping function $\hat{f}(e_P)$, which maps e_P to the union of all its mention lists in the contexts of the corpus. Then following (Page et al., 1998), we compute:

$$g(e_P|P) = \mathbb{E}_{v\sim\hat{f}(e_P)}\left[(\mathbf{h}_{\texttt{[MASK]}})^{\mathsf{T}}\mathbf{x}_v\right] - \log q(e_P) \tag{7}$$

$$\mathbf{h}_{\texttt{[MASK]}} = \mathcal{LN}(\sigma(\mathcal{F}_\theta(\mathbf{x}_{\texttt{[MASK]}})\mathbf{W}_2)). \tag{8}$$

Here, $\mathbf{x}_{\texttt{[MASK]}} \in \mathbb{R}^h$ is the input embedding vector of the masked token and $\mathbf{W}_2 \in \mathbb{R}^{h\times h}$ is a trainable weight matrix. Further, $\mathbf{x}_v$ denotes the embedding of mention v from $\hat{f}(e_P)$, which is calculated by averaging embeddings of all tokens included in v.

Finally, the total loss can be computed as:

$$\mathcal{L} = \mathcal{L}_{\text{MLM}} + \lambda\mathcal{L}_{\text{PRI}} + \mu\mathcal{L}_{\text{MPM}}, \tag{9}$$

where $\mathcal{L}_{\text{MLM}}$ denotes the vanilla MLM objective in PLMs, $\lambda, \mu \in [0, 1]$ are the balancing coefficients.

4 Experiments

In this section, we comprehensively evaluate the effectiveness of KP-PLM. We also conduct the hyperparameter analysis in Appendix B.3.

4.1 Implementation Details

Following previous works (Sun et al., 2020; Zhang et al., 2021), the pre-training data is collected from Wikipedia Dumps (2020/03/01)[5] and consists of 25,933,196 sentences. Further, the KB used is WikiData5M (Wang et al., 2021b), which includes $3,085,345$ entities and 822 relation types.

For the baselines, we select six knowledge-enhanced PLMs: 1) **ERNIE-THU** (Zhang et al., 2019) integrates knowledge embeddings with aligned mentions. 2) **KnowBERT** (Peters et al., 2019) utilizes the attention mechanism to realize knowledge fusion. 3) **KEPLER** (Wang et al., 2021b) introduces a novel knowledge embedding loss for capturing knowledge. 4) **CoLAKE** (Sun et al., 2020) constructs a unified graph from context and knowledge base. 5) **K-Adapter** (Wang et al., 2021a) learns representations for different kinds of knowledge via neural adapters. 6) **DKPLM** (Zhang et al., 2021) aims at injecting long-tailed entities.

In the pre-training stage, we choose RoBERTa-base (Liu et al., 2019) from HuggingFace[6] as the default PLM. Our framework can also be easily combined with other PLMs, such as BERT (Pörner et al., 2019) and DeBERTa (He et al., 2021b). In the fine-tuning stage (if have), we fine-tune each task with only task-specific data. Further, we introduce a model variant KP-PLM$_{KNOW}$, which employs knowledge prompts in the fine-tuning stage by directly concatenating them with each example. More implementation details about the pre-training and fine-tuning stages are provided in Appendices A and B, respectively.

Models	P	R	F1
UFET	77.4	60.6	68.0
BERT	76.4	71.0	73.6
RoBERTa	77.4	73.6	75.4
ERNIE$_{BERT}$	78.4	72.9	75.6
ERNIE$_{RoBERTa}$	80.3	70.2	74.9
KnowBERT$_{BERT}$	77.9	71.2	74.4
KnowBERT$_{RoBERTa}$	78.7	72.7	75.6
KEPLER$_{WiKi}$	77.8	74.6	76.2
CoLAKE	77.0	75.7	76.4
DKPLM	79.2	75.9	77.5
KP-PLM	**80.8**	75.1	77.8
KP-PLM$_{KNOW}$	80.5	**76.1**	**78.2**

Table 1: The results (%) on Open Entity.

Methods	TACRED			FewRel		
	P	R	F1	P	R	F1
CNN	70.3	54.2	61.2	-	-	-
PA-LSTM	65.7	64.5	65.1	-	-	-
C-GCN	69.90	63.3	66.4	-	-	-
BERT	67.2	64.8	66.0	84.9	85.1	85.0
RoBERTa	70.0	69.6	70.2	85.4	85.4	85.3
ERNIE$_{BERT}$	70.0	66.1	68.1	88.5	88.4	88.3
KnowBERT	71.6	71.5	71.5	-	-	-
CoLAKE	72.8	73.5	73.1	**90.6**	**90.6**	**90.6**
DKPLM	72.6	73.5	73.0	-	-	-
KP-PLM	72.6	73.7	73.3	87.6	87.4	87.5
KP-PLM$_{KNOW}$	**73.3**	**73.9**	**73.5**	**88.9**	**88.9**	**88.8**

Table 2: The results (%) on TACRED and FewRel.

[5] https://dumps.wikimedia.org/enwiki/.

[6] https://huggingface.co/transformers.

Datasets	ELMo	BERT	RoBERTa	CoLAKE	K-Adapter*	KEPLER	DKPLM	KP-PLM
Google-RE	2.2	11.4	5.3	9.5	7.0	7.3	10.8	**11.0**
UHN-Google-RE	2.3	5.7	2.2	4.9	3.7	4.1	5.4	**5.6**
T-REx	0.2	32.5	24.7	28.8	29.1	24.6	32.0	**32.3**
UHN-T-REx	0.2	23.3	17.0	20.4	**23.0**	17.1	22.9	22.5

Table 3: The performance P@1 (%) of knowledge probing. Besides, K-Adapter* is based on RoBERTa-large and uses a subset of T-REx as its training data, which may contribute to its superiority over other methods.

		Single-Sentence Natural Language Understanding Tasks								
Paradigms	**Methods**	**SST-2** (acc)	**SST-5** (acc)	**MR** (acc)	**CR** (acc)	**MPQA** (acc)	**Subj** (acc)	**TREC** (acc)	**CoLA** (matt.)	**Avg.**
PT-Zero	RoBERTa	82.57	29.46	**65.10**	**82.15**	49.90	**69.20**	20.80	-4.89	49.29
	KP-PLM	**84.15**	**30.67**	64.15	81.60	**53.80**	68.70	**24.80**	**-2.99**	**50.61**
PT-Few	RoBERTa	86.35±1.3	36.79±2.0	**83.35**±0.9	**88.85**±1.4	66.40±1.9	89.25±2.6	76.80±5.0	6.61±6.9	66.80
	KP-PLM	**90.71**±1.0	**44.21**±2.9	82.00±1.5	85.35±0.4	**67.30**±1.2	**91.45**±0.4	**81.00**±3.3	**24.28**±11.3	**70.79**
FT-Full	RoBERTa	94.90	56.90	**89.60**	88.80	86.30	**96.50**	**97.10**	63.90	84.25
	KP-PLM	**95.30**	**57.63**	89.20	**89.10**	**87.40**	96.20	**97.10**	**64.87**	**84.60**
		Sentence-Pair Natural Language Understanding Tasks								
Paradigms	**Methods**	**MNLI** (acc)	**MNLI-mm** (acc)	**SNLI** (acc)	**QNLI** (acc)	**RTE** (acc)	**MRPC** (f1)	**QQP** (f1)	**STS-B** (pear.)	**Avg.**
PT-Zero	RoBERTa	38.92	39.09	**36.33**	**47.08**	55.96	54.87	45.72	-4.10	39.23
	KP-PLM	**40.10**	**43.53**	36.23	46.87	**58.29**	**58.97**	**46.30**	**-1.30**	**41.12**
PT-Few	RoBERTa	50.87±1.8	53.01±2.2	64.96±1.9	**60.08**±2.4	63.54±4.0	78.57±3.7	45.72±3.4	52.26±7.0	58.63
	KP-PLM	**55.50**±1.1	**57.56**±2.1	**66.33**±1.9	58.75±2.2	**66.25**±3.8	**79.78**±3.1	**62.60**±4.0	**68.79**±4.2	**64.45**
FT-Full	RoBERTa	87.33	87.01	**92.10**	92.58	77.32	90.23	92.16	91.12	88.73
	KP-PLM	**87.99**	**87.32**	92.01	**93.04**	**79.48**	**91.81**	**92.42**	**91.60**	**89.46**

Table 4: The comparison between KP-PLM and RoBERTa-base (Liu et al., 2019) over multiple natural language understanding (NLU) tasks in terms of acc/f1/matt./pear. (%) and standard deviation with three paradigms, such as zero-shot prompt-tuning (PT-Zero), few-shot prompt-tuning (PT-Few) and full-data fine-tuning (FT-Full).

4.2 Knowledge-aware Tasks

In the fine-tuning stage, we use three tasks: entity typing, relation extraction and knowledge probing, to evaluate the model performance.

Entity Typing. Given a sentence and a corresponding entity mention, the task is to predict the type of the mention. We choose precision (P), recall (R) and F1 score as the evaluation metrics. For all these metrics, the larger the value, the better the model performance. For fairness, we follow the same training settings as in (Zhang et al., 2021) and use the Open Entity (Choi et al., 2018) dataset. To show the effectiveness and competitiveness of our pre-trained model, we also choose UFET (Choi et al., 2018), BERT (Pörner et al., 2019) and RoBERTa (Liu et al., 2019) as traditional baselines for this task.

The results are summarized in Table 1. From the table, we see that knowledge-enhanced PLMs generally perform better than traditional methods. In particular, our methods can achieve the best results w.r.t. all the metrics. For example, the F1 score of KP-PLM$_{KNOW}$ is 78.2%, which improves that of the runner-up by 0.7%. Further, we also find that both KP-PLM and KP-PLM$_{KNOW}$ outperform RoBERTa. This indicates that injecting knowledge prompts into both pre-training and fine-tuning stages are useful for improving the model performance on this task.

Relation Extraction. It aims to classify the relation between two given entities based on the corresponding texts. We follow (Sun et al., 2020) to choose two widely used tasks: TACRED (Zhang et al., 2017) and FewRel (Han et al., 2018). Similar as in entity typing, we take precision (P), recall (R) and F1 as the evaluation metrics. We also compare our models with five traditional models including CNN, PA-LSTM (Zhang et al., 2017), C-GCN (Zhang and Qi, 2018), BERT (Pörner et al., 2019) and RoBERTa (Liu et al., 2019).

As shown in Table 2, most knowledge-enhanced PLMs outperform traditional methods by a large

Knowledge Paradigms	PLM	*Full-data Fine-tuning* Open Entity (f1)	TACRED (f1)	FewRel (f1)	SST-2 (acc)	CoLA (matt.)	QQP (f1)
None	BERT	73.6	66.0	85.0	93.5	52.1	71.2
	RoBERTa	75.4	70.2	85.3	94.9	63.9	92.2
	DeBERTa	76.5	72.1	87.0	95.1	64.9	89.3
Knowledge Masking	BERT	74.4	70.5	85.8	93.9	51.6	71.4
	RoBERTa	75.8	71.3	86.0	95.0	64.3	92.0
	DeBERTa	77.0	72.6	86.8	94.7	65.0	90.3
Knowledge Fusion†	BERT	75.6	68.1	88.3	93.5	52.3	71.2
	RoBERTa	74.9	68.4	88.4	93.9	63.6	91.5
	DeBERTa	76.8	70.6	88.8	94.2	65.7	89.9
Graph Learning‡	BERT	75.1	**72.9**	**88.4**	94.2	53.9	72.0
	RoBERTa	76.4	73.1	**90.6**	94.6	63.4	**93.3**
	DeBERTa	77.1	72.8	**89.5**	94.0	66.1	**92.2**
Knowledge Prompting	BERT	**76.0**	72.2	87.1	**94.6**	**57.3**	**75.8**
	RoBERTa	**77.8**	**73.3**	87.5	**95.3**	**64.9**	92.4
	DeBERTa	**77.7**	**73.5**	88.0	**95.6**	**66.3**	**92.2**

Table 5: The comparison between KP-PLM and other knowledge-enhanced paradigms on different base PLMs. For each base PLM, we highlight the largest score in bold.

margin. This shows the necessity of external knowledge incorporation for PLMs. For the TACRED task, our models perform the best. For the FewRel task, while CoLAKE obtains the best results, KP-PLM$_{KNOW}$ can improve the F1 score over ERNIE and RoBERTa by 0.5% and 3.5%, respectively. In addition, the outperformance of KP-PLM$_{KNOW}$ over KP-PLM shows the importance of injecting knowledge prompts in the fine-tuning stage.

Knowledge Probing. Knowledge probing aims to evaluate whether the PLM possesses the intrinsic factual knowledge in zero-shot settings. We compare all the methods w.r.t. P@1. We select LAMA (Petroni et al., 2019) and the advanced version LAMA-UHN (Pörner et al., 2019) tasks with four datasets, including Google-RE, UHN-Google-RE, T-REx, and UHN-T-REx.

From Table 3, our model KP-PLM beats other knowledge-enhanced PLMs on Google-RE, UHN-Google-RE and T-REx datasets. For UHN-T-REx, KP-PLM performs comparably with the winner and improve RoBERTa by 5.5% in terms of P@1. In addition, we see that BERT achieves the best performance over multiple tasks. This is because it uses a small vocabulary set, which has also been pointed out in (Sun et al., 2020).

4.3 Performance on General NLU Tasks

We further investigate whether knowledge prompting can consistently improve the PLM performance in both full-resource and low-resource learning. We follow (Gao et al., 2021a) to select 15 widely used NLU tasks. We consider three training settings, including zero-shot prompt-tuning (PT-Zero), few-shot prompt-tuning (PT-Few), and full data fine-tuning (FT-Full). Details on these datasets and training procedures are provided in Appendix B.2.

From Table 4, we make the following observations: 1) KP-PLM achieves the best overall results over all the training settings. The much larger margins of KP-PLM over others in PT-Zero and PT-Few show that continual pre-training by knowledge prompting indeed improves the performance when using the prompt-based fine-tuning technique. 2) In the low-resource scenarios, the performances of both RoBERTa and KP-PLM on single-sentence tasks are better than sentence-pair tasks, which indicates that the sentence-pair task is more difficult in the few-shot settings. 3) We also find KP-PLM sometimes performs worse than RoBERTa on MR and CR datasets. We conjecture that this is because these tasks are sensitive to external knowledge.

4.4 Knowledge Prompting Study

We end this section with a further comparison between knowledge prompting and other knowledge-enhanced paradigms including knowledge masking, knowledge fusion and graph learning. For each paradigm, we employ three PLMs: BERT-base, RoBERTa-base and DeBERTa-base. For knowledge masking, we mask all the entity mentions in each training sentence, and train the model by the vanilla MLM objective. For knowledge fusion and graph learning, we directly run the source codes

Models	Open Entity	TACRED	FewRel
KP-PLM	**77.8**	**73.3**	**87.5**
w/o. PRI	77.5	72.8	87.1
w/o. MPM	77.4	72.5	86.8
w/o. PRI & MPM	77.2	72.4	86.6
w/o. cont.	77.7	73.1	87.3
w/o. mask.	77.5	72.9	87.1
w/o. all	76.3	70.8	85.9

Table 6: The ablation study results (F1 score %). w/o. PRI removes the PRI task, w/o. MPM removes the MPM task, w/o. PRI & MPM removes both the PRI and MPM tasks, w/o. cont. removes all continuous pseudo tokens, w/o. mask. replaces the mask matrix in Eq. 1 with the default in RoBERTa, and w/o. all denotes to remove all the proposed techniques.

of ERNIE and CoLAKE, respectively. The results are summarized in Table 5. From the table, we see that, compared with other knowledge-enhanced paradigms, for each base PLM, knowledge prompting can lead to largest performance gains in most downstream tasks. This further verifies the effectiveness of knowledge prompting in boosting the performance of existing PLMs.

4.5 Ablation Study

We conduct an ablation study to investigate the characteristics of main components in KP-PLM, including prompt relevance inspection (PRI), masked prompt modeling (MPM), continuous pseudo tokens in the prompt and the mask matrix for self-attention in PLMs (Eq. 1). Table 6 reports the F1 scores on the Open Entity, TACRED, and FewRel tasks. From the table, we observe that the removal of each component leads to the performance drop of KP-PLM, which shows the importance of all these components in KP-PLM.

5 Conclusion

In this paper, we presented a seminal knowledge prompting paradigm, based on which a novel knowledge-prompting-based PLM framework KP-PLM was proposed. We constructed contextual knowledge sub-graphs for contexts and employed continuous prompting mapping to generate knowledge prompts. After that, we designed two self-supervised pre-training tasks to learn semantic knowledge from prompts. Finally, we conducted extensive experiments to evaluate the model performance. Experimental results validate the effectiveness of knowledge prompting in boosting the performance of PLMs.

Limitations

We have listed some limitations: 1) In the experiments, we follow most previous works to only focus on general natural language understanding (NLU) tasks. We do not evaluate the performance over some question answering tasks (e.g. SQuAD, HotpotQA, etc.), we let it as the future research. 2) Our work focuses on the PLM without any transformer decoders. We think it is possible to extend our method in natural language generation (NLG) tasks.

Ethical Considerations

Our contribution in this work is fully methodological, namely a knowledge-prompting-based pre-trained language model (KP-PLM) to boost the performance of PLMs with factual knowledge. Hence, there is no explicit negative social influences in this work. However, transformer-based models may have some negative impacts, such as gender and social bias. Our work would unavoidably suffer from these issues. We suggest that users should carefully address potential risks when the KP-PLM models are deployed online.

Acknowledgement

Xiang Li would like to acknowledge the support from Shanghai Pujiang Talent Program (Project No. 21PJ1402900). This work has also been supported by the National Natural Science Foundation of China under Grant No. U1911203, Alibaba Group through the Alibaba Innovation Research Program, and 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

Simran Arora, Sen Wu, Enci Liu, and Christopher Ré. 2022. Metadata shaping: A simple approach for knowledge-enhanced language models. In *ACL*, pages 1733–1745.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*, pages 632–642.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda

Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS*.

Qianglong Chen, Feng-Lin Li, Guohai Xu, Ming Yan, Ji Zhang, and Yin Zhang. 2022. Dictbert: Dictionary description knowledge enhanced language model pre-training via contrastive learning. In *IJCAI*, pages 4086–4092.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 87–96.

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

Paolo Ferragina and Ugo Scaiella. 2010. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In *CIKM*, pages 1625–1628.

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

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021b. Simcse: Simple contrastive learning of sentence embeddings. In *EMNLP*, pages 6894–6910.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2021. PPT: pre-trained prompt tuning for few-shot learning. *CoRR*, abs/2109.04332.

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

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. *arXiv preprint arXiv:1810.10147*.

Lei He, Suncong Zheng, Tao Yang, and Feng Zhang. 2021a. Klmo: Knowledge graph enhanced pretrained language model with fine-grained relationships. In *EMNLP*, pages 4536–4542.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021b. Deberta: decoding-enhanced bert with disentangled attention. In *ICLR*.

Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Juanzi Li, and Maosong Sun. 2021. Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. *CoRR*, abs/2108.02035.

Robert L. Logan IV, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. 2019. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *ACL*, pages 5962–5971.

Sébastien Jean, KyungHyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *ACL*, pages 1–10.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Trans. Assoc. Comput. Linguistics*, 8:64–77.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *ICLR*.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-BERT: enabling language representation with knowledge graph. In *AAAI*, pages 2901–2908.

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.

Ye Liu, Yao Wan, Lifang He, Hao Peng, and Philip S. Yu. 2021c. KG-BART: knowledge graph-augmented BART for generative commonsense reasoning. In *AAAI*, pages 6418–6425.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL*, pages 1003–1011.

Page, Lawrence, Brin, Sergey, and Terry. 1998. Page, l., et al.: The pagerank citation ranking: Bringing order to the web. *stanford digital libraries working paper*.

Matthew E. Peters, Mark Neumann, Robert L. Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *EMNLP*, pages 43–54.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language models as knowledge bases? In *EMNLP*, pages 2463–2473. Association for Computational Linguistics.

Nina Pörner, Ulli Waltinger, and Hinrich Schütze. 2019. BERT is not a knowledge base (yet): Factual knowledge vs. name-based reasoning in unsupervised QA. *CoRR*, abs/1911.03681.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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*, pages 4222–4235.

Yusheng Su, Xu Han, Zhengyan Zhang, Yankai Lin, Peng Li, Zhiyuan Liu, Jie Zhou, and Maosong Sun. 2021. Cokebert: Contextual knowledge selection and embedding towards enhanced pre-trained language models. *AI Open*, 2:127–134.

Tianxiang Sun, Yunfan Shao, Xipeng Qiu, Qipeng Guo, Yaru Hu, Xuanjing Huang, and Zheng Zhang. 2020. Colake: Contextualized language and knowledge embedding. In *COLING*, pages 3660–3670.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. ERNIE: enhanced representation through knowledge integration. *CoRR*, abs/1904.09223.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2021a. K-adapter: Infusing knowledge into pre-trained models with adapters. In *ACL*, pages 1405–1418.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021b. 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*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, pages 5754–5764.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for knowledge graph completion. *CoRR*, abs/1909.03193.

Hongbin Ye, Ningyu Zhang, Shumin Deng, Xiang Chen, Hui Chen, Feiyu Xiong, Xi Chen, and Huajun Chen. 2022. Ontology-enhanced prompt-tuning for few-shot learning. In *WWW2022*, pages 778–787.

Donghan Yu, Chenguang Zhu, Yiming Yang, and Michael Zeng. 2022a. JAKET: joint pre-training of knowledge graph and language understanding. In *AAAI*, pages 11630–11638.

Wenhao Yu, Chenguang Zhu, Yuwei Fang, Donghan Yu, Shuohang Wang, Yichong Xu, Michael Zeng, and Meng Jiang. 2022b. Dict-bert: Enhancing language model pre-training with dictionary. In *ACL*, pages 1907–1918.

Taolin Zhang, Chengyu Wang, Nan Hu, Minghui Qiu, Chengguang Tang, Xiaofeng He, and Jun Huang. 2021. DKPLM: decomposable knowledge-enhanced pre-trained language model for natural language understanding. *CoRR*, abs/2112.01047.

Yuhao Zhang and Peng Qi. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *EMNLP*, pages 2205–2215.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *EMNLP*, pages 35–45.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: enhanced language representation with informative entities. In *ACL*, pages 1441–1451.

A Details of the Pre-training

A.1 Training Corpus and Knowledge Base

We follow (Liu et al., 2020; Sun et al., 2020; Zhang et al., 2021) to collect training corpora from Wikipedia (2020/03/01)[7], and use WikiExtractor[8] to process the training data. The knowledge base (KB) $\mathcal{G}$ we choose is WikiData5M (Wang et al., 2021b), which is an urge-large structure data source based on Wikipedia. The entity linking toolkit and

[7] https://dumps.wikimedia.org/enwiki/.

[8] https://github.com/attardi/wikiextractor.

the verbalizer we used are both the TAGME[9] (Ferragina and Scaiella, 2010), which can be viewed as a well-designed mapping function between KB entities set $\mathcal{E}$ and the PLM vocabulary set $\mathcal{V}$. In total, we have 3,085,345 entities and 822 relation types in $\mathcal{G}$, and 25,933,196 training sentences. For each sentence, we construct the contextual knowledge sub-graph and generate continuous knowledge prompts offline by the process described in Section 3.2. In average, the number of the entities and knowledge prompts for each sentence are 15 and 8, respectively.

As mentioned above, our framework consists of three main training objectives. For the Masked Language Modeling (MLM) task, we follow (Devlin et al., 2019; Liu et al., 2019) to randomly select 15% tokens only in the context. For the selected tokens, 80% of them are replaced with `[MASK]` token, and 10% of them are replaced with randomly sampled tokens from the whole vocabulary set. For the Prompt Relevance Inspection (PRI) task, we only select one prompt from each input example to make the decision. For the Masked Prompt Modeling (MPM) task, we also only select one prompt (not equal to the selected prompt in PRI) and mask one entity.

A.2 Pre-training Implement Details

In the pre-training stage, we choose RoBERTa-base (Liu et al., 2019) from the HuggingFace[10] as the default underlying PLM. We train our model by AdamW algorithm with $\beta_1 = 0.9, \beta_2 = 0.98$. The learning rate is set as 1e-5 with a warm up rate 0.1. The best balance coefficients we found are $\lambda = \mu = 0.5$. The number of negative entities is $N = 10$. Especially, if the number of the entities in the current sub-graph $|\mathcal{E}_S| < N + 1$, the remaining $N - |\mathcal{E}_S| + 1$ negative entities can be sampled from the whole entities set $\mathcal{E}$ from KB $\mathcal{G}$. We also leverage dropout and regularization strategies to avoid over-fitting. The model is trained on 8 V100-32G GPUs for 2.5 days with a total batch size of 768.

B Details of the Task-specific Fine-tuning

We choose multiple natural language processing (NLP) tasks for the model evaluations, including knowledge-aware tasks and general natural language understanding (NLU) tasks. In this section, we provide the details of the dataset information and fine-tuning procedures for each task.

[9] https://sobigdata.d4science.org/group/tagme.

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

Datasets	#Train	#Develop	#Test	#Class
Open Entity	2,000	2,000	2,000	6
TACRED	68,124	22,631	15,509	42
FewRel	8,000	8,000	16,000	80

Table 7: The statistics of Open Entity, TACRED and FewRel.

B.1 Knowledge-aware Tasks

We select three knowledge-aware tasks, such as entity typing, relation extraction, and knowledge probing. The dataset statistics can be found in Table 7.

Entity typing. Entity typing requires to classify the designated entity mention based on the given sentence. We select the Open Entity dataset for evaluation. It consists of 6 entity types, and 2,000 sentences for training, developing, and testing data, respectively.

The variant method KP-PLM$_{KNOW}$ means adding injected knowledge prompts for each data. In this manner, the topic entity is the designated entity mention, we follow the proposed *knowledge prompting* procedure to obtain the knowledge prompts for each sentence, and directly concatenate them with the original sentence. In the fine-tuning stage, we do not use knowledge-aware pre-training objectives.

During fine-tuning, we follow (Sun et al., 2020) to add two special tokens `[ENT]` and `[/ENT]` before and after the entity mention. We directly concatenate their representations at the top of the PLM and use them for classification. The max sequence length we set is 128, the learning rate is set as 2e-5 with a warm up rate of 0.1, and the batch size is set to 16. Generally, we can obtain the best performance after 5 epochs.

Relation Extraction. Relation extraction is one of the significant tasks for information retrieval. It aims to classify the relation class between two given entities based on the corresponding aligned sentence. We select two widely used datasets, i.e. TACRED (Zhang et al., 2017) and FewRel (Han et al., 2018). TACRED has 42 relation types and 106, 264 sentences in total. FewRel is constructed by distant supervised (Mintz et al., 2009) and denoised by human annotations. It consists of 320,000 sentences and 80 relations.

For the variant KP-PLM$_{KNOW}$, following the

Category	Dataset	#Class	#Train	#Test	Type	Labels (classification tasks)
single-sentence	SST-2	2	6,920	872	sentiment	positive, negative
	SST-5	5	8,544	2,210	sentiment	v. pos., positive, neutral, negative, v. neg.
	MR	2	8,662	2,000	sentiment	positive, negative
	CR	2	1,775	2,000	sentiment	positive, negative
	MPQA	2	8,606	2,000	opinion polarity	positive, negative
	Subj	2	8,000	2,000	subjectivity	subjective, objective
	TREC	6	5,452	500	question cls.	abbr., entity, description, human, loc., num.
	CoLA	2	8,551	1,042	acceptability	grammatical, not_grammatical
sentence-pair	MNLI	3	392,702	9,815	NLI	entailment, neutral, contradiction
	SNLI	3	549,367	9,842	NLI	entailment, neutral, contradiction
	QNLI	2	104,743	5,463	NLI	entailment, not_entailment
	RTE	2	2,490	277	NLI	entailment, not_entailment
	MRPC	2	3,668	408	paraphrase	equivalent, not_equivalent
	QQP	2	363,846	40,431	paraphrase	equivalent, not_equivalent
	STS-B	$\mathbb{R}$	5,749	1,500	sent. similarity	-

Table 8: The statistics of multiple NLU datasets. STS-B is a real-valued regression task over the interval $[0, 5]$). Since the original test data is invisible, we use the develop set as our test set.

procedure of *knowledge prompting*, we construct a knowledge sub-graph for each sentence. Since each sentence consists of two entities, both of them can be viewed as the topic entity. Hence, we obtain two sub-graphs for each sentence and then merge two sub-graphs by removing the duplicate relation paths. After that, we construct multiple prompts and concatenate them with the original sentence.

During fine-tuning, we follow (Sun et al., 2020) to add four special tokens `[HD]`, `[/HD]`, `[TL]` and `[/TL]` to identity the head entity and tail entity in the given sentence. We obtain the representations of `[HD]` and `[TL]` from the last layer of the PLM, and concatenate them to perform classification. For TACRED (Zhang et al., 2017) dataset, the max sequence length is 256, the learning rate is 3e-5 with a warm up rate 0.1, the batch size we set is 32. We run experiment for 5 epochs. For FewRel (Han et al., 2018), the max sequence length is 256, the learning rate is 2e-5 with a warm up rate 0.1, the batch size we set is 32. We run experiment for 8 epochs. Notice, we do not use the original N-way K-shot settings for FewRel (Han et al., 2018).

Knowledge Probing. Knowledge probing is a challenging task to evaluate whether the PLM grasps factual knowledge. The task requires no training stages which belongs to the zero-shot setting. We select LAMA (Peters et al., 2019) and LAMA-UHN (Pörner et al., 2019) tasks with four datasets, i.e., Google-RE, UHN-Google-RE, T-REx, and UHN-T-REx. The dataset statistics can be found in the original paper in (Pörner et al., 2019).

During the evaluation, we follow (Zhang et al., 2019) to directly use the given designed template. Specifically, given a head entity, a relation, and a template with `[MASK]` token, we feed the input sequence into the PLM and obtain the predicted result at the masked position. For example in Google-RE, a manually defined template of relation "place_of_birth" is "`[S]` was born in `[MASK]`.", we can denote `[S]` as the head entity "Obama", and let the PLM generate the answer "US." at the position of `[MASK]`.

B.2 General Natural Language Understanding Tasks

We select NLU tasks for the evaluation in both full-resource and low-resource scenarios. We follow (Gao et al., 2021a) to select 15 tasks include 8 tasks from GLUE benchmark (Wang et al., 2019), SNLI (Bowman et al., 2015), and 6 other sentence-level classification tasks. The statistics of each dataset are shown in Table 8.

As mentioned above, we provide three training settings, including zero-shot prompt-tuning (PT-Zero), few-shot prompt-tuning (PT-Few) and full-data fine-tuning (FT-Full). We summarize the detail settings in the following.

Prompt-tuning. For the settings of PT-Zero and PT-Few, we follow Gao et al. (2021a) to evaluate our framework based on the prompt-tuning method. Specifically, for each task, we directly use the automatically generated discrete template and verbalizer, where the discrete template denotes the

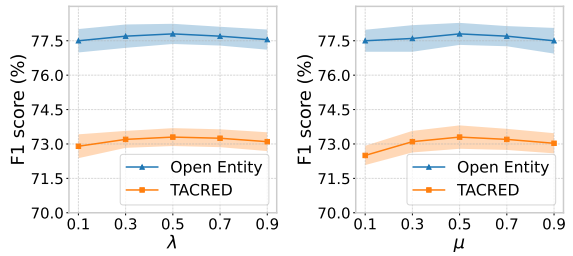

Figure 4: Hyper-parameter efficiency of λ and μ over Open Entity and TACRED.

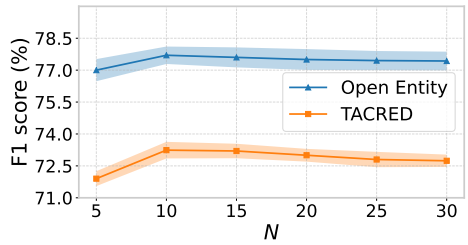

Figure 5: Hyper-parameter efficiency of N over Open Entity and TACRED.

token sequence with a single `[MASK]` token, and the verbalizer is the label mapping function that can map generated result token to the label. The template and verbalizer for each task we used can be found in Table E.1 in the original paper of LM-BFF (Gao et al., 2021a). Take sentiment analysis as an example, for the sentence S =“The film is very attractive.”, we have:

$$X = \texttt{[CLS]}\ S\ \text{It was}\ \texttt{[MASK].[SEP]}$$

$$\mathcal{M} = \{\text{“positive”: “great”; “negative”: “boring”}\}$$

where X is the input sequence, $\mathcal{M}(y)$ denotes the verbalizer that map the label y to a label word. For example, $\mathcal{M}(\text{“positive”}) = \text{“great”}$. Under these settings, we can transform arbitrary NLU tasks into a cloze-style problem. In our experiments, the baseline is RoBERTa-base (Liu et al., 2019). In addition, to validate the contributions of our proposal, we do not use other techniques mentioned in LM-BFF, such as demonstration learning and prompt ensemble.

For the zero-shot learning, we directly use the vanilla MLM to generate the result at the `[MASK]` position. Formally, given an input sequence X, the label set $\mathcal{Y}$, and the verbalizer $\mathcal{M}$, the prediction can be calculated as:

$$\hat{y} = \text{argmax}_{y \in \mathcal{Y}}(\Pr_{\mathcal{F}}(\texttt{[MASK]} = \mathcal{M}(y)|X))$$

where $\mathcal{F}$ denotes the PLM, $\Pr(\cdot)$ denotes the probability distribution generated by the MLM head of $\mathcal{F}$. For the regression task (i.e. SST-B), we directly use the probability value to represent the real-valued result.

k	#Entities	Open Entity	TACRED	FewRel	Avg.
1	6	77.1	72.9	86.9	79.0
2	15	**77.8**	**73.3**	87.5	**79.5**
3	32	77.6	71.7	**88.7**	79.3
4	62	75.8	69.9	85.4	77.0

Table 9: The effectiveness (F1 value %) of the k-hop sub-graph for *knowledge prompting*.

For the few-shot learning, we follow Gao et al. (2021a) to construct few-shot training/developing set, which consists only 16 examples for each label. For example, we have 32 examples for training and developing set for SST-2 dataset. During the training, the batch size is 4, the learning rate is set as 1e-5 with a warm up rate 0.1. We leverage the cross-entropy loss to train for 64 epochs, and evaluate the model over the whole testing set.

Fine-tuning. For the full-data fine-tuning (FT-Full), we follow the standard supervised training settings (which can also be found in LM-BFF.

B.3 Hyper-parameter Analysis

We also conduct a hyper-parameter sensitivity study on Open Entity and TACRED tasks. Specifically, we study three key hyper-parameters: the balancing coefficients λ and μ for PRI and MPM tasks in Eq. 9, and the number of negative entities N in the MPM task. We vary one hyper-parameter with others fixed. From Figure 4, as λ (μ) increases, the performance of KP-PLM first increases and then drops, and it can achieve the best result when $\lambda = 0.5$ ($\mu = 0.5$). For N in Figure 5, we observe that the model performance improves significantly at the beginning, and then decreases when $N > 10$. It shows that too many negative entities may introduce noise to degrade the model performance.

Finally, we analyze the effectiveness of the k-hop sub-graph. Specifically, we construct 1-hop, 2-hop, 3-hop and 4-hop sub-graphs for knowledge prompting. We conduct experiments on Open Entity, TACRED and FewRel task. Results in Table 9 suggest that we can achieve the best performance when $k = 2$. We also find that the overall model performance decreases when $k > 2$. This may be explained by the introduction of many redundant and irrelevant entities.