AI Open 3 (2022) 182–192

Contents lists available at ScienceDirect

AI Open

journal homepage: www.keaipublishing.com/en/journals/ai-open

PTR: Prompt Tuning with Rules for Text Classification

Xu Han [a,1], Weilin Zhao [a,1], Ning Ding [a], Zhiyuan Liu [a,b,c,d,*], Maosong Sun [a,b,c,d,*]

[a] Dept. of Comp. Sci. & Tech., Institute for AI, Tsinghua University, Beijing National Research Center for Information Science and Technology, China
[b] Institute Guo Qiang, Tsinghua University, China
[c] International Innovation Center of Tsinghua University, China
[d] Beijing Academy of Artificial Intelligence, BAAI, China

ARTICLE INFO

Keywords:
Pre-trained language models
Prompt tuning

ABSTRACT

Recently, prompt tuning has been widely applied to stimulate the rich knowledge in pre-trained language models (PLMs) to serve NLP tasks. Although prompt tuning has achieved promising results on some few-class classification tasks, such as sentiment classification and natural language inference, manually designing prompts is cumbersome. Meanwhile, generating prompts automatically is also difficult and time-consuming. Therefore, obtaining effective prompts for complex many-class classification tasks still remains a challenge. In this paper, we propose to encode the prior knowledge of a classification task into rules, then design sub-prompts according to the rules, and finally combine the sub-prompts to handle the task. We name this **P**rompt **T**uning method with **R**ules "**PTR**". Compared with existing prompt-based methods, PTR achieves a good trade-off between effectiveness and efficiency in building prompts. We conduct experiments on three many-class classification tasks, including relation classification, entity typing, and intent classification. The results show that PTR outperforms both vanilla and prompt tuning baselines, indicating the effectiveness of utilizing rules for prompt tuning. The source code of PTR is available at https://github.com/thunlp/PTR.

1. Introduction

Recently, pre-trained language models (PLMs) (Devlin et al., 2019) have emerged as an effective NLP instrument, since PLMs can capture linguistic (Jawahar et al., 2019), semantic (Yenicelik et al., 2020), syntactic (Hewitt and Manning, 2019), and world knowledge (Petroni et al., 2019) from large-scale corpora. By fine-tuning PLMs on task-specific data, the rich knowledge of PLMs can well serve various downstream NLP tasks (Qiu et al., 2020). Despite the success of fine-tuning PLMs, recent studies (Liu et al., 2021a) indicate one critical challenge — the gap of objective forms between pre-training and fine-tuning. As shown in Fig. 1(a), PLMs are usually pre-trained with a cloze-style task. However, in the fine-tuning phase like Fig. 1(b), we are required to tune all PLMs' parameters and task-specific heads (e.g., linear layers for classification) with task-specific objectives. From a transfer learning perspective (Pan and Yang, 2009), this objective gap may hinder the adaptation of PLMs' knowledge to downstream tasks.

To bridge the above-mentioned gap of objective forms, prompt tuning has been introduced (Brown et al., 2020). A typical prompt consists of a template and a label word set. As shown in Fig. 1(c), a classification task can be transformed into a cloze-style objective form, by combining the input and the template to predict the masked position [M] and then mapping the predicted words to the corresponding class labels. The label word set serves as the candidate set for predicting [M]. Intuitively, using the template "$< S_1 >$. It was [M]." and the label set {"great", "terrible"} for sentiment classification, we can determine whether $< S_1 >$ is positive or negative based on PLMs predicting "great" or "terrible" at [M].

Besides sentiment classification, prompt tuning has also achieved promising results on some other few-class classification tasks such as natural language inference (Schick and Schütze, 2021; Liu et al., 2021b). However, for those tasks with many classes, it becomes challenging to manually find appropriate templates and suitable label words to distinguish different classes. For example, relation classification, a typical many-class classification task, requires models to predict semantic relations between two marked entities in the text. Given the relation "person:parent" and the relation "organization:parent", it is hard to pick templates and label words to distinguish them.

* Corresponding authors.
E-mail addresses: hanxu2022@tsinghua.edu.cn (X. Han), zwl19@mails.tsinghua.edu.cn (W. Zhao), dingn18@mails.tsinghua.edu.cn (N. Ding), liuzy@tsinghua.edu.cn (Z. Liu), sms@tsinghua.edu.cn (M. Sun).
[1] Equal contribution.
[2] In some preliminary works, prompt tuning means using prompts to tune PLMs. After the work (Lester et al., 2021) prompt tuning gradually refers to freezing PLMs' parameters and only tuning soft prompts. In this paper, "prompt tuning" refers to all methods using prompts to tune PLMs.

https://doi.org/10.1016/j.aiopen.2022.11.003
Received 14 November 2022; Accepted 22 November 2022
Available online 23 November 2022

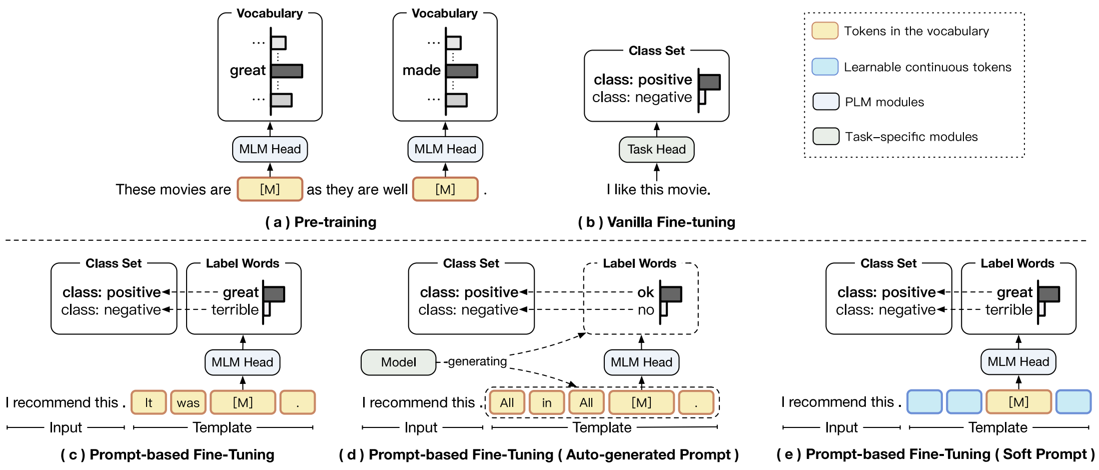
Vocabulary
great
made
MLM Head
These movies are [M] as they are well [M] .
(a) Pre-training
Class Set
class: positive
class: negative
Task Head
I like this movie.
(b) Vanilla Fine-tuning
Tokens in the vocabulary
Learnable continuous tokens
PLM modules
Task–specific modules
Label Words
terrible
I recommend this . It was [M] .
Input
Template
(c) Prompt-based Fine-Tuning
ok
no
Model
generating
All in All
(d) Prompt-based Fine-Tuning (Auto-generated Prompt)
(e) Prompt-based Fine-Tuning (Soft Prompt)

Fig. 1. An illustration of pre-training, fine-tuning and prompt tuning.

One straightforward solution is to automatically generate prompts like Fig. 1(d), yet auto-generated prompts (Shin et al., 2020; Gao et al., 2020) require extra computation costs for generation and verification. Another solution is using soft prompts like Fig. 1(e) instead of hard discrete prompt tokens (Lester et al., 2021; Gu et al., 2021),[2], but soft prompts require model parameters to be large enough to be effective.

In this paper, we propose prompt tuning with rules (PTR) for many-class classification tasks. Given a classification task, as shown in Fig. 2, we first encode the prior task knowledge into rules and decompose the task into sub-tasks, then design essential sub-prompts, and finally follow the rules to compose sub-prompts to handle the task. Compared with existing prompt tuning methods, PTR has two advantages:

(1) **Prior Knowledge Encoding**. Prior task knowledge can effectively help solve specific tasks. For example, in relation classification, considering that the prediction results are related to both sentence semantics and entity types, and we can build prompts based on two sub-prompts, one for detecting entity types and the other for detecting relational semantics between entities. For other tasks like entity typing and intent classification, their class hierarchies are also good priors for designing prompts. By encoding prior task knowledge, we can obtain effective models even without sufficient training data.

(2) **Efficient and Effective Prompt Design**. Compared with manually designing individual templates and label words for each class, it is easier to design a few simple sub-prompts and combine them according to rules to handle specific complex tasks. Compared with auto-generated and soft prompts, using rules to generate prompts is more efficient and effective. Considering intuitiveness, we use human-picked sub-prompts in this paper to introduce PTR, yet PTR is an open framework and sub-prompts can also be auto-generated and soft ones.

To verify the effectiveness of PTR, we conduct experiments on three typical many-class classification tasks, including relation classification, entity typing, and intent classification. The experimental results show that PTR can outperform vanilla fine-tuning and prompt tuning baselines, indicating that PTR is a promising approach to take advantage of both prior task knowledge and PLMs for those complicated classification tasks.

2. Preliminaries

Before introducing PTR, we first give some essential preliminaries, especially some details about prompt tuning for classification. Formally, a classification task can be denoted as $\mathcal{T} = \{\mathcal{X}, \mathcal{Y}\}$, where $\mathcal{X}$ is the instance set and $\mathcal{Y}$ is the class set. For $x \in \mathcal{X}$, it consists of tokens $x = \{w_1, w_2, \dots, w_{|x|}\}$, and is annotated with a label $y_x \in \mathcal{Y}$.

2.1. *Vanilla fine-tuning for PLMs*

Given a PLM $\mathcal{M}$, vanilla fine-tuning first converts the instance $x = \{w_1, w_2, \dots, w_{|x|}\}$ to the input sequence $\{\texttt{[CLS]}, w_1, w_2, \dots, w_{|x|}, \texttt{[SEP]}\}$, and then uses $\mathcal{M}$ to encode all tokens of the input sequence into corresponding vectors $\{\mathbf{h}_{\texttt{[CLS]}}, \mathbf{h}_{w_1}, \dots, \mathbf{h}_{w_{|x|}}, \mathbf{h}_{\texttt{[SEP]}}\}$. For a classification task, a task-specific head is used to get the probability distribution over $\mathcal{Y}$ with the softmax function $p(\cdot|x) = \texttt{Softmax}(\mathbf{W}\mathbf{h}_{\texttt{[CLS]}} + \mathbf{b})$, where $\mathbf{h}_{\texttt{[CLS]}}$ is the vector of [CLS], $\mathbf{b}$ is a learnable bias vector, and $\mathbf{W}$ is a learnable matrix randomly initialized before fine-tuning. The parameters of $\mathcal{M}$, $\mathbf{b}$, and $\mathbf{W}$ are tuned to maximize the objective $\frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} \log p(y_x|x)$.

2.2. *Prompt tuning for PLMs*

To tune PLMs with cloze-style objectives, prompt tuning has been proposed. Formally, a prompt consists of a template $T(\cdot)$ and a label word set $\mathcal{V}$. For each instance x, the template is first used to map x to the prompt input $x_{\text{prompt}} = T(x)$. The template defines where each token of x is placed and whether to add any additional tokens. Besides retaining the original tokens in x, at least one [M] is placed into x_{prompt} for $\mathcal{M}$ to fill label words. For instance, for a binary sentiment classification task, we set a template $T(\cdot) =$ "$\cdot$ It was[M]." and map x to $x_{\text{prompt}} =$ "x It was[M].".

By input x_{prompt} into $\mathcal{M}$, we compute the hidden vector $\mathbf{h}_{\texttt{[M]}}$ of [M]. Given $v \in \mathcal{V}$, we produce the probability that the token v can fill the masked position $p(\texttt{[M]} = v|x_{\text{prompt}}) = \frac{\exp(\mathbf{v}\cdot\mathbf{h}_{\texttt{[M]}})}{\sum_{\tilde{v}\in\mathcal{V}}\exp(\tilde{\mathbf{v}}\cdot\mathbf{h}_{\texttt{[M]}})}$, where $\mathbf{v}$ and $\tilde{\mathbf{v}}$ are the embeddings of the token v and $\tilde{v}$ respectively in $\mathcal{M}$.

There also exists an injective mapping function $\phi : \mathcal{Y} \to \mathcal{V}$ that bridges the set of classes and the set of label words. In some papers, the function ϕ is named "verbalizer". With the verbalizer ϕ, we can formalize the probability distribution over $\mathcal{Y}$ with the probability distribution over $\mathcal{V}$ at the masked position, i.e. $p(y|x) = p(\texttt{[M]} = \phi(y)|x_{\text{prompt}})$. Here, we also take a binary sentiment classification task as an example. We map the positive sentiment to "great" and the negative sentiment to "terrible". According to $\mathcal{M}$ fills the masked position of x_{prompt} with "great" or "terrible", we can know the instance x is positive or negative.

With the template $T(\cdot)$, the label word set $\mathcal{V}$, and the verbalizer ϕ, the objective is to maximize $\frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} \log p(\texttt{[M]} = \phi(y_x)|T(x))$. Fig. 1 can clearly show the connections and differences among pre-training, fine-tuning and prompt tuning.

3. Prompt tuning with rules (PTR)

Considering the difficulty of designing effective prompts for many-class classification tasks, we propose PTR to decompose one challenging task into several simple sub-tasks, design sub-prompts respectively, and

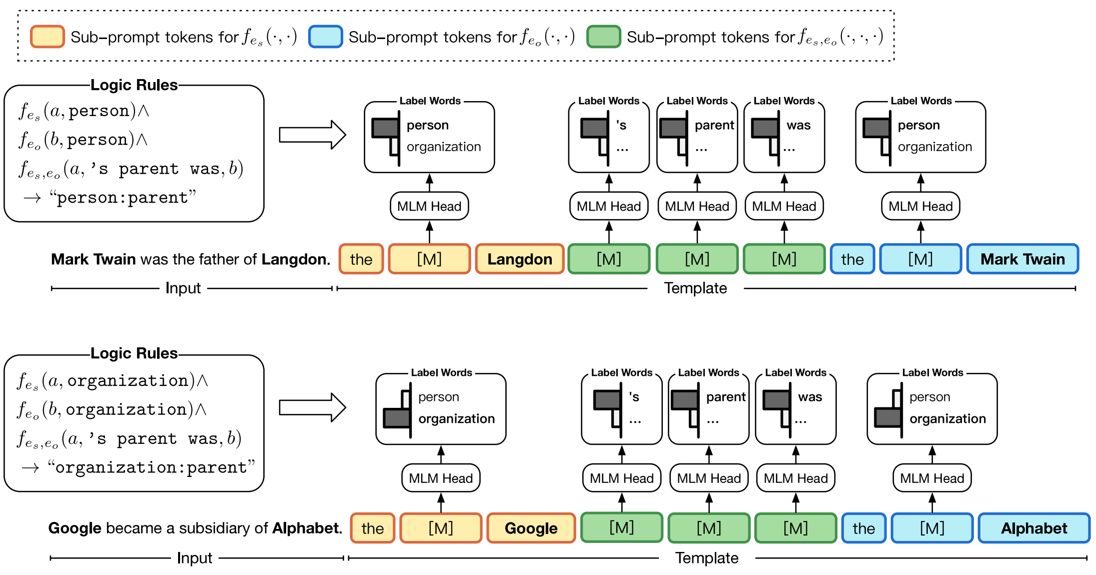
Sub-prompt tokens for $f_{e_s}(\cdot,\cdot)$
Sub-prompt tokens for $f_{e_o}(\cdot,\cdot)$
Sub-prompt tokens for $f_{e_s,e_o}(\cdot,\cdot,\cdot)$
Logic Rules
$f_{e_s}(a,\texttt{person})\wedge$ $f_{e_o}(b,\texttt{person})\wedge$ $f_{e_s,e_o}(a,\texttt{'s parent was},b)$ $\rightarrow$ "person:parent"
Label Words
person organization
's ...
parent ...
was ...
MLM Head
Mark Twain was the father of Langdon.
the [M] Langdon [M] [M] [M] the [M] Mark Twain
Input
Template
Logic Rules
$f_{e_s}(a,\texttt{organization})\wedge$ $f_{e_o}(b,\texttt{organization})\wedge$ $f_{e_s,e_o}(a,\texttt{'s parent was},b)$ $\rightarrow$ "organization:parent"
Google became a subsidiary of Alphabet.
the [M] Google [M] [M] [M] the [M] Alphabet

Fig. 2. Illustration of PTR. Here we take the task relation classification for example. By composing the sub-prompts of the conditional functions $f_{e_s}(\cdot,\cdot)$, $f_{e_o}(\cdot,\cdot)$ and $f_{e_s,e_o}(\cdot,\cdot,\cdot)$, we could easily get an effective prompt to distinguish "person:parent" and "organization:parent".

incorporate sub-prompts with rules to better solve the original complex task. In this section, we will introduce the details of PTR and how to apply PTR for classification tasks.

3.1. Overall framework of PTR

PTR is driven by basic human inferences. For example, in relation classification, if we wonder whether two marked entities in a sentence have the relation "person:parent", with prior knowledge, we would check whether the sentence and two marked entities meet certain conditions: (1) *The two marked entities are human beings*; (2) *the sentence indicates the parental semantics between the two marked entities*.

Inspired by this, for any text classification task $\mathcal{T}=\{\mathcal{X},\mathcal{Y}\}$, we could decompose the task into a set of conditional function, named $\mathcal{F}$. Each function $f\in\mathcal{F}$ determines whether the function input meets certain conditions. For instance, we can design $f(a,\texttt{person})$ to determine whether a is a person and $f(a,\texttt{'s parent was},b)$ to determine whether b is the parent of a. Intuitively, these conditional functions are essentially the predicates of first-order rules.

For each conditional function $f\in\mathcal{F}$, PTR set a template $T_f(\cdot)$ and a label word set $\mathcal{V}_f$ to build sub-prompt. According to the semantics of $\mathcal{Y}$, we can use rules to transform the task into the calculation of a series of conditional functions. As shown in Fig. 2, for relation classification, determining whether the entities a and b have the relation "person:parent" can be formalized as

$$f_{e_s}(a,\texttt{person})\wedge f_{e_s,e_o}(a,\texttt{'s parent was},b)\wedge f_{e_o}(b,\texttt{person})$$
$$\rightarrow \text{"person:parent"},$$

where $f_{e_s}(\cdot,\cdot)$ is the conditional function to determine the subject entity types, $f_{e_o}(\cdot,\cdot)$ is the conditional function to determine the object entity types, and $f_{e_s,e_o}(\cdot,\cdot,\cdot)$ is the conditional function to determine the semantic connection between entities. Based on the above-mentioned rules and conditional functions, we can compose the sub-prompts of rule-related conditional functions to handle the task $\mathcal{T}$. Next, we will introduce how to decompose the task into conditional functions, how to design sub-prompts for conditional functions as well as how to aggregate sub-prompts for tasks.

3.2. Task decomposition

The fundamental rule for decomposing tasks into sub-tasks is to use structural information. The first is to design rules based on structural labels. The labels of classification tasks may have internal structural information, and this information is usually presented as keywords that appear repeatedly in label tags. For example, in intent classification, "card_activating" and "card_linking" share the same word "card". Classifying these two labels can be decomposed into whether it is related to "card" as well as whether "activating" or "linking" is intended. The second is to design rules based on intrinsic correlation. Some label tags do not have any common keywords but have the same intrinsic meaning, such as "uber" and "car_rental".

According to the above rules, we can transform a many-class classification task into a hierarchical classification form and then design a condition function for each hierarchical layer. Following these rules, we can easily build several sub-prompts to handle complex classification tasks. More examples are depicted in Table 3, Table 6, and Table 10. These examples show how to decompose relation classification, entity typing, and intent classification. From these examples, we can find that the efforts needed to construct manual sub-prompts are not that hard.

How to decompose tasks and design rules is a very open problem. For example, besides using structural information, decomposing tasks with some more complex heuristic rules is also promising. Since this paper focuses on introducing the framework of rule-based prompt tuning, we left attempting more methods to decompose tasks and design rules for our future work.

3.3. Sub-prompts for conditional functions

For each conditional function $f\in\mathcal{F}$, we manually design a sub-prompt for it. Similar to the conventional prompt setting, a sub-prompt also consists of a template and a set of label words.

The basic conditional function is a unary function. For binary sentiment classification, unary functions can determine whether a sentence is positive or negative; For entity typing, unary functions can indicate entity types; For topic labeling, unary functions can be used to predict article topics.

We still take relation classification for example. Given a sentence $x=\{\dots e_s\dots e_o\dots\}$, where e_s is the subject entity and e_o is the object entity, we set $f_{e_s}(\cdot,\{\text{person}|\text{organization}|\dots\})$ to determine the type of e_s. The sub-prompt template and label word set of f_{e_s} can be formalized as

$$T_{f_{e_s}}(x)=\text{"}x\text{ the}[\texttt{M}]e_s\text{"},\qquad(1)$$
$$\mathcal{V}_{f_{e_s}}=\{\text{"person"},\text{"organization"},\dots\}.$$

For the object entity e_o, the sub-prompt template and label word set of f_{e_o} are similar to Eq. (1).

Another important type of conditional functions is a binary function. For natural language inference, binary functions can determine the relationship between two sentences as entailment, neutral, or contradiction; For relation classification, such functions can be used to determine the complex connections between two entities. If we set $f_{e_s,e_o}(\cdot, \{\text{'s parent was}|\text{was born in}|\ldots\}, \cdot)$, the sub-prompt template and label word set of f_{e_s,e_o} can be formalized as

$$\begin{aligned} T_{f_{e_s,e_o}}(x) &= \text{“ } x \ \ e_s \, \texttt{[M]}\,\texttt{[M]}\,\texttt{[M]}\, e_o\text{”},\\ \mathcal{V}_{f_{e_s,e_o}} &= \{\text{“'s parent was”}, \text{“was born in”}, \ldots\}. \end{aligned} \tag{2}$$

Normally, unary and binary functions are sufficient to compose rules to handle classification tasks. For classification tasks with complex semantics, these design methods of unary and binary functions can be extended to multi-variable functions to build more powerful sub-prompts.

3.4. Composing sub-prompts for tasks

Various conditional functions may interfere with each other. For example, the probability $P(f_{e_s,e_o}(a, \texttt{'s parent was}, b) = \text{true})$ would be low conditioned on the high probability of both $P(f_{e_s}(a, \texttt{person}) = \text{true})$ and $P(f_{e_o}(b, \texttt{person}) = \text{false})$.

Therefore, instead of training sub-prompts separately, we need to compose sub-prompts of several conditional functions into a complete task-specific prompt. In this paper, we apply a simple strategy: we use rules with conjunctive normal form and then directly concatenate the sub-prompts of all rule-related functions. For instance, in Fig. 2, by aggregating the sub-prompts in Eqs. (1) and (2), the complete prompt template is as follows,

$$\begin{aligned} &T(x) = [T_{f_{e_s}}(x); T_{f_{e_s,e_o}}(x); T_{f_{e_o}}(x)] = \\ &\text{“ } x \ \ \text{the } \texttt{[M]}_1 \ e_s \texttt{[M]}_2 \ \texttt{[M]}_3 \ \texttt{[M]}_4 \ \text{the } \texttt{[M]}_5 \ e_o \text{ ”}, \end{aligned} \tag{3}$$

where $[\cdot;\cdot;\cdot]$ is the aggregation function of sub-prompts. And the aggregated label word sets are

$$\begin{aligned} \mathcal{V}_{\texttt{[M]}_1} &= \{\text{“person”}, \text{“organization”}, \ldots\},\\ \mathcal{V}_{\texttt{[M]}_2} &= \{\text{“'s”}, \text{“was”}, \ldots\},\\ \mathcal{V}_{\texttt{[M]}_3} &= \{\text{“parent”}, \text{“born”}, \ldots\},\\ \mathcal{V}_{\texttt{[M]}_4} &= \{\text{“was”}, \text{“in”}, \ldots\},\\ \mathcal{V}_{\texttt{[M]}_5} &= \{\text{“person”}, \text{“organization”}, \ldots\}. \end{aligned} \tag{4}$$

As the aggregated template contains multiple [M] tokens, we must consider all masked positions to make predictions at the inference time, i.e.,

$$p(y|x) = \frac{\prod_{j=1}^{n} p(\texttt{[M]}_j = \phi_j(y)|T(x))}{\sum_{\tilde{y}\in\mathcal{Y}} \prod_{j=1}^{n} p(\texttt{[M]}_j = \phi_j(\tilde{y})|T(x))}, \tag{5}$$

where n is the number of masked positions in $T(x)$, and $\phi_j(y)$ is to map the class y to the label word set $\mathcal{V}_{\texttt{[M]}_j}$ for the jth masked position $\texttt{[M]}_j$. Eq. (5) can be used to tune PLMs and classify classes.

During training, we focus on optimizing the independent probability of each conditional function, and joint probability for the whole task. The final training loss of PTR is

$$\begin{aligned} \mathcal{L} &= \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} \left[\mathcal{L}_{\text{indep}}(x) + \mathcal{L}_{\text{joint}}(x)\right],\\ \mathcal{L}_{\text{indep}}(x) &= -\frac{1}{n}\sum_{j=1}^{n} \log p\big(\texttt{[M]}_j = \phi_j(y_x)|T(x)\big),\\ \mathcal{L}_{\text{joint}}(x) &= -\log p\big(y_x|x\big), \end{aligned} \tag{6}$$

where $\mathcal{L}_{\text{joint}}$ is built based on Eq. (5).

Table 1
Statistics of relation classification datasets.

Dataset	#train	#dev	#test	#rel
SemEval	6,507	1,493	2,717	19
TACRED	68,124	22,631	15,509	42
TACREV	68,124	22,631	15,509	42

3.5. Sub-prompts with multiple label words

In some cases, using an injective mapping function ϕ to map each class to a single label word is not sufficient. We take entity typing for example, given an entity type class “location/city”, mapping the class to the label word “city” or several words that are highly related to “city” like “urban” or “town” are both possible options. In fact, there are also some entity types that refer to a series of related entities such as “location/lake_sea_ocean”. For such types, it is difficult to assign them to a single label word. To this end, instead of choosing only one label word, a simple and effective way is to map each class to multiple related label words.

Formally, given an input sentence x, we change the label words for its class y_x at the jth masked position $\texttt{[M]}_j$ from $\phi_j(y_x) = w_j$ to $\phi_j(y_x) = \{w_{j,1}, w_{j,2}, \ldots, w_{j,m}\}$. With the template $T(\cdot)$, the conditional probability $p\big(\texttt{[M]}_j = \phi_j(y_x)|T(x)\big)$ in Eq. (6) becomes

$$\frac{1}{m}\sum_{k=1}^{m} \lambda_k \ p\big(\texttt{[M]}_j = w_{j,k}|T(x)\big) \tag{7}$$

where λ_k is the weighted average coefficient indicating the importance of each label word. In our experiments, λ_k are default to be all 1.

4. Experiments

We conduct experiments on three typical many-class classification tasks including relation classification, entity typing, and intent classification.

4.1. The results on relation classification

In this section, we will present the experimental results on relation classification.

Datasets

As shown in Table 1, we carry out our experiments on the following three datasets:

TACRED (Zhang et al., 2017): one of the largest and most widely used datasets for relation classification. It is obtained via crowd-sourcing and contains 42 relation types (including “no_relation”).

TACREV (Alt et al., 2020): one dataset built based on the original TACRED. They correct some errors in the original development set and test set of TACRED, while the training set is left intact.

ReTACRED (Stoica et al., 2021): another version of TACRED. They address some shortcomings of the original TACRED, and refactor its training set, development set, and test set.

Baselines and implementation details

In this part, we compare PTR with several typical models for relation classification.

Learning Models from Scratch. For relation classification, the typical approach is learning neural models from scratch. Here we choose PA-LSTM (Zhang et al., 2017) and C-GCN (Zhang et al., 2018) as our baselines, as these two models are the most effective ones based on recurrent neural networks and graph neural networks.

Fine-Tuning PLMs. Some efforts are devoted to fine-tuning PLMs for relation classification, including: (1) For vanilla fine-tuning, we directly select RoBERTa_LARGE as our baseline without adding any special markers to the input. (2) Considering entity information is important

Table 2
The template example with "Mark Twain was born in Florida" as input. "(punct)" indicates markers are from PLMs' vocabulary.

Model	Template Example
Using randomly initialized markers	
ENT Marker	[E1] Mark Twain [/E1] was born in [E2] Florida [/E2]
TYP Marker	[E1:Person] Mark Twain [/E1:Person] was born in [E2:State] Florida [/E2:State]
Using the vocabulary of PLMs	
ENT Marker (punct)	@ Mark Twain @ was born in # Florida #
TYP Marker (punct)	@ * person * Mark Twain @ was born in # ^ state ^ Florida #
AdaPrompt	Mark Twain was born in Florida. Florida was [M] of Mark Twain
PTR	Mark Twain was born in Florida. The [M] Mark Twain [M] [M] [M] the [M] Florida

Table 3
For some typical relations of TACRED, TACREV, and ReTACRED, their relation-specific label words for the PTR template: "< S_1 >.The $[M]_1$ < E_1 > $[M]_2$ $[M]_3$ $[M]_4$ the $[M]_5$ < E_2 >.", where < S_1 > is the input sentence, < E_1 > and < E_2 > are entity mentions in the sentence.

Class Label	$[M]_1$	$[M]_2$	$[M]_3$	$[M]_4$	$[M]_5$
per:country_of_birth	person	was	born	in	country
per:city_of_birth	person	was	born	in	city
per:employee_of	person	's	employee	was	organization
per:parents	person	's	parent	was	person
org:founded_by	organization	was	founded	by	person
org:country_of_headquarters	organization	was	located	in	country
org:city_of_headquarters	organization	was	located	in	city
org:members	organization	's	member	was	organization
org:parents	organization	's	parent	was	organization
no_relation	entity	was	irrelevant	to	entity

to understand relational semantics, knowledge-enhanced PLMs have been further explored to enhance PLMs with knowledge bases. For knowledge-enhanced PLMs, we select SpanBERT (Joshi et al., 2020), KnowBERT (Peters et al., 2019), LUKE (Yamada et al., 2020), and MTB (Baldini Soares et al., 2019) as our baselines. These baselines use knowledge to enhance learning objectives, input features, model architectures, and pre-training strategies respectively.

Prompt tuning. We also compare PTR with several methods using prompts, including: (1) Zhou and Chen (2021) introduce several marker-based methods, which can add entity markers to the input as additional information. Among these marker-based models, ENT Marker is similar to prompting by introducing extra serial information to indicate entity positions, TYP Marker additionally introduces entity type information to build templates for relation classification. (2) AdaPrompt Chen et al. (2021a) introduces an adaptive selection method for label words, which can scatter each relation label into a variable number of label words to handle the complex many-class problem. In Table 2, we show some template examples of these prompt-base fine-tuning methods. For PTR, we also show some details of relation-specific prompts in Table 3.

Our model PTR is implemented based on the toolkit Transformers (Wolf et al., 2020) and OpenNRE (Han et al., 2019). Most hyper-parameters are set following previous works (Yamada et al., 2020; Zhou and Chen, 2021): applying RoBERTa_large as the model backbone and optimizing the model among the learning rate $\{1e-5, 3e-5, 5e-5\}$ and a linear warmup for the first 10% steps. The weight decay is set to $1e-2$. For all datasets, we tune models for 5 epochs with the batch size 64. The best model checkpoint is selected based on the performance on the development set. Following the settings of baselines, we randomly sample the data five times using different random seeds, and get the average results. For all the above-mentioned baselines, we report the results of their papers as well as some results reproduced by Zhou and Chen (2021).

Comparison between PTR and fine-tuning

Table 4 shows the overall performance. From the table, we can see that:

(1) Conventional methods design complicated models, including various recurrent neural layers, attention neural layers, and graph neural layers. Although learning these complicated models from scratch have achieved effective results, the vanilla PLM RoBERTa_large can easily outperform them, indicating that the rich knowledge of PLMs captured from large-scale unlabeled data is important for downstream tasks.

(2) Within PLMs, those knowledge-enhanced PLMs defeat the vanilla PLM RoBERTa_large. On the one hand, this shows that introducing extra task-specific knowledge to enhance models is effective. On the other hand, this indicates that simply fine-tuning PLMs cannot completely cover the knowledge required for downstream tasks.

(3) Within knowledge-enhanced PLMs, LUKE and KnowBERT are better than other models. This is because these two models respectively use knowledge as data augmentation and architecture enhancement in the fine-tuning phase. In contrast, SpanBERT and MTB inject task-specific knowledge into parameters in the pre-training phase and then fine-tune their parameters without extra knowledge. All the above results indicate that even if the task-specific knowledge is already contained in PLMs, it is difficult for fine-tuning to stimulate the knowledge for downstream tasks.

(4) PTR achieves significant improvements over all baselines. Even compared with those knowledge-enhanced models, PTR still outperforms these models on TACRED. In general, all these experimental results show that the pre-training phase enables PLMs to capture sufficient task-specific knowledge, how to stimulate the knowledge for downstream tasks is crucial. Compared with typical fine-tuning methods, PTR can better stimulate task-specific knowledge.

Comparison between PTR and prompt tuning

From Tables 2 and 4, we can find that using the vocabulary of PLMs for prompts can lead to better results. By designing particular sub-prompts, PTR can achieve comparable results to the TYP Marker methods. Furthermore, PTR establishes the effectiveness in experiments without additional human annotations. Note that PTR derives the sub-prompt label of an entity solely based on existing relation type annotations, and predicts entity type at the inference time. While TYP Marker(Zhou and Chen, 2021) uses extra human annotations and tells the model the ground-truth entity types both at the training and inference.

To further compare the models using prompts, we introduce the few-shot learning scenario. Following the few-shot setting in Gao et al. (2020), we sample K training instances and K validation instances per class from the original training set and development set, and evaluate models on the original test set. We set K from $\{8, 16, 32\}$ respectively and use a fixed set of 5 random seeds to sample instances.

From Table 5, we find that although PTR performs poorer than TYP Marker (punct) using the full training dataset, since TYP Marker (punct) uses entity typing human-annotations for training and testing, PTR still achieves comparable or even better results in the few-shot scenario, especially in ReTACRED where the data has fewer annotation errors. Compared with TYP Marker (punct) and AdaPrompt, we attribute our success to the use of both human knowledge and model knowledge.

4.2. The results on entity typing

In this section, we will present the experimental results on entity typing.

Table 4
F_1 scores (%) on TACRED, TACREV, ReTACRED. Fine-Tuning means the vanilla fine-tuning of RoBERTa_large without adding any entity markers. In the "Extra Data" column, "w/o" indicates that no additional data is used for pre-training and fine-tuning, yet "w/" means that extra data or knowledge bases are used for data augmentation. In the "Extra Annotation" column, we show whether models use extra human annotations on the datasets to provide more information. Since the models with † use additional human annotations to help training and inference, we do not compare their results with other models for fairness. The best results are bold.

Model	Extra Annotation	Extra Data	TACRED	TACREV	ReTACRED
Learning models from scratch					
PA-LSTM (Zhang et al., 2017)	w/o	w/o	65.1	73.3	79.4
C-GCN (Zhang et al., 2018)	w/o	w/o	66.3	74.6	80.3
Fine-tuning methods					
Fine-Tuning (Liu et al., 2019b)	w/o	w/o	68.7	76.0	84.9
KnowBERT (Peters et al., 2019)	w/o	w/	71.5	79.3	–
MTB (Baldini Soares et al., 2019)	w/o	w/	70.1	–	–
SpanBERT (Joshi et al., 2020)	w/o	w/	70.8	78.0	85.3
LUKE (Yamada et al., 2020)	w/o	w/	**72.7**	80.6	90.3
Prompt tuning methods					
AdaPrompt (Chen et al., 2021a)	w/o	w/o	–	80.8	–
ENT Marker (cls) (Zhou and Chen, 2021)	w/o	w/o	69.4	79.8	88.4
ENT Marker (Zhou and Chen, 2021)	w/o	w/o	70.7	80.2	90.5
ENT Marker (punct) (Zhou and Chen, 2021)	w/o	w/o	71.4	81.2	90.5
TYP Marker (Zhou and Chen, 2021) †	w/	w/o	71.0	80.8	90.5
TYP Marker (punct) (Zhou and Chen, 2021) †	w/	w/o	74.6	83.2	91.1
PTR	**w/o**	**w/o**	72.4	**81.4**	**90.9**

Table 5
F_1 scores (%) with different training instance numbers on TACRED, TACREV, and ReTACRED.

Model	TACRED			TACREV			ReTACRED		
	8	16	32	8	16	32	8	16	32
ENT Marker (punct) (Zhou and Chen, 2021)	27.0	31.3	31.9	27.4	31.2	32.0	44.1	53.7	59.9
TYP Marker (punct) (Zhou and Chen, 2021)	**28.9**	**32.0**	**32.4**	27.6	31.2	32.0	44.8	54.1	60.0
AdaPrompt (Chen et al., 2021a)	–	–	–	25.2	27.3	30.8	–	–	–
PTR	28.1	30.7	32.1	**28.7**	**31.4**	**32.4**	**51.5**	**56.2**	**62.1**

Table 6
For some typical types of FewNERD, OntoNotes, and BBN, their type-specific label words for the PTR and PLET templates. The PTR template is "< S_1>. In this sentence, the [M]$_1$ entity < E_1> is a [M]$_2$", and the PLET template is "< S_1>. In this sentence, < E_1> is a [M]$_1$". < S_1> is the input sentence and < E_1> is the entity mention for typing.

Dataset	Class Label	PTR		PLET
		[M]$_1$	[M]$_2$	[M]$_1$
FewNERD	location-bodiesofwater	location	water	water
	person-artist/author	person	artist/author	artist/author
	organization-sportsteam	organization	sport/team	sport/team
OntoNotes	location/geograpy/island	location	geograpy/island	location/geograpy/island
	person/religious_leader	person	religious/leader	person/religious/leader
	other/event/election	event	election	event/election
BBN	location/lake_sea_ocean	location	lake/sea/ocean	location/lake/sea/ocean
	facility/highway_street	facility	highway/street	facility/highway/street
	event/war	event	war	event/war

Table 7
Statistics of entity typing datasets.

Dataset	#train	#dev	#test	#type
FewNERD	340,382	48,758	96,901	66
Ontonotes	253,239	2,200	8,962	86
BBN	86,077	12,824	12,824	46

Datasets

As shown in Table 7, we carry out our experiments on the following three datasets:

FewNERD (Ding et al., 2021d): a large-scale manually annotated dataset, with well-structured entity types of two layers: 8 coarse-grained entity types and 66 fine-grained entity types. Each fine-grained entity type belongs to one of the 8 coarse-grained entity types.

OntoNotes (Weischedel et al., 2013): a dataset selected from the OntoNotes 5.0 dataset in experiments. We follow the setting of PLET (Ding et al., 2021a) to adopt the 86-classes version of OntoNotes, and the data split is identical to Shimaoka et al. (2017).

BBN (Weischedel and Brunstein, 2005): a dataset selected from Penn Treebank corpus of Wall Street Journal texts. We also follow the setting of PLET (Ding et al., 2021a) to adopt the 46-classes version of BBN, and the data split is identical to Ren et al. (2017).

For all the above three datasets, we use the accuracy (Acc), loose micro F_1 (MiF) and loose macro F_1 (MaF) as the metric for evaluation. The loose F_1-score calculation concerns type labels by different granularities.

Baselines and implementation details

In the experiments of entity typing, we use vanilla fine-tuning and PLET (Ding et al., 2021a) as our baseline. PLET is also a prompt-based

Table 8
The accuracy, loose micro-F1, loose macro-F1 scores (%) on the three datasets. The results of Fine-Tuning and PLET are from the original paper of PLET (Ding et al., 2021a). The best results are bold.

Shot	Metric	FewNERD			OntoNotes			BBN		
		Fine-Tuning	PLET	PTR	Fine-Tuning	PLET	PTR	Fine-Tuning	PLET	PTR
1	Acc	8.9	43.9	**47.0**	3.7	**39.0**	38.5	0.8	40.7	**53.6**
	MiF	19.9	60.6	**62.5**	19.0	**59.9**	57.6	5.8	49.3	**59.9**
	MaF	19.9	60.6	**62.5**	19.4	**61.4**	59.0	4.4	48.5	**60.2**
2	Acc	20.8	47.8	**52.0**	7.3	39.2	**40.2**	6.7	41.3	**59.7**
	MiF	32.7	62.1	**66.4**	24.9	**61.1**	60.5	13.7	54.0	**67.4**
	MaF	32.7	62.1	**66.4**	25.6	**62.7**	61.6	13.2	52.0	**67.4**
4	Acc	33.1	57.0	**58.2**	11.2	38.4	**39.9**	19.3	52.2	**59.9**
	MiF	44.1	68.6	**70.2**	27.7	59.8	**60.5**	27.0	61.1	**68.1**
	MaF	44.1	68.6	**70.2**	28.3	60.9	**61.7**	24.7	58.9	**67.3**
8	Acc	46.4	55.8	**60.7**	18.4	39.4	**42.7**	27.0	44.3	**51.2**
	MiF	57.8	68.7	**72.0**	38.2	58.0	**63.3**	40.2	56.2	**61.4**
	MaF	57.8	68.7	**72.0**	37.8	58.3	**64.3**	39.5	55.2	**59.1**
16	Acc	61.0	61.6	**62.9**	32.3	42.3	**46.4**	39.7	**55.0**	51.1
	MiF	71.6	72.4	**74.0**	51.4	60.8	**65.2**	49.0	**62.8**	61.43
	MaF	71.6	72.4	**74.0**	51.5	61.8	**66.2**	47.1	**62.4**	59.5
Full	Acc	79.8	79.9	**80.1**	59.7	**60.4**	57.0	62.4	65.9	**69.9**
	MiF	85.7	85.8	**86.0**	70.5	70.8	**72.2**	68.0	71.6	**75.4**
	MaF	85.7	85.8	**86.0**	76.6	**76.4**	73.5	67.4	70.8	**74.8**

method, but it differs from our method in that only one masked position is used in the template to predict label words. In Table 6, we show more details of the differences between the design of type-specific prompts in PTR and that in PLET.

Our model PTR is implemented based on the toolkit Transformers (Wolf et al., 2020) and OpenPrompt (Ding et al., 2021b). For a fair comparison, we follow the same settings as PLET: BERT_base is adopted as our backbone model and optimized among the learning rate $\{1e-5, 3e-5, 5e-5\}$ and a linear warmup for the first 500 steps. The training batch size used is 16 for all models. The weight decay is set to $1e-2$. In the supervised setting, each model is trained for 10 epochs. In the few-shot setting, each model is trained for 30 epochs. We tune models under the 1-shot, 2-shot, 4-shot, 8-shot, 16-shot and fully-supervised settings, where K-shot means K training examples per entity type. According to baseline's settings, we randomly sample the data five times using different random seeds, and get the average results. For all the above baselines, we report the results in the paper of Ding et al. (2021a).

Comparison between PTR and baselines

Table 8 shows the performance of PTR and baseline models on the three datasets. From the table, we can find that:

(1) Both PLET and PTR significantly outperform the vanilla fine-tuning method in all experimental scenarios. This is consistent with the conclusion of the relation extraction experiments in Tables 4 and 5.

(2) By comparing the results under different shot settings, we can find that PLET and PTR have achieved improvements over vanilla fine-tuning when the number of training samples is small, indicating that much knowledge to solve entity typing comes from PLMs themselves, using prompts can better stimulate the entity typing information distributed in PLMs to solve these many-class classification problems,

(3) The results show that by providing another masked position for the model to explicitly learn the structured hierarchy information of entity types, our approach works better than PLET which only encodes all type-specific information into one masked position. In general, by encoding the prior knowledge of the entity type schema into a rule orienting both coarse-grained-level and fine-grained-level entity information, following this rule to design prompts can guide PLMs to better detect entity types according to the type hierarchy.

4.3. The results on intent classification

In this section, we will present the experimental results on the task intent classification.

Table 9
Statistics of intent classification datasets.

Dataset	#train	#dev	#test	#intent
CLINC150	15,000	3,000	4,500	150
BANKING77	8,622	1,540	3,080	77
HWU64	8,954	1,076	1,076	64

Datasets

As shown in Table 9, we carry out our experiments on the following three datasets:

CLINC150 (Larson et al., 2019): a dataset covers fine-grained 150 intents in 10 domains, such as banking, work, travel, cooking, as well as some meta information.

BANKING77 (Casanueva et al., 2020): a dataset created for fine-grained intent detection in the banking domain.

HWU64 (Liu et al., 2019a): a dataset covers fine-grained 64 intents in 21 domains. All these domains are mainly related to reflect human–home robot interaction.

More details of these datasets are shown in the Table. For all the above three datasets, we use the accuracy (%) as the metric for evaluation.

Baselines and implementation details

In the intent classification experiment, we follow the same settings as in the paper of CPFT (Zhang et al., 2021a). We use RoBERTa_base as the backbone, set the batch size to 16. We tune all models under the 5-shot (5 training examples per intent) and 10-shot settings (10 training examples per intent), since both 5-shot and 10-shot are widely used in previous works. We optimize our models among the learning rate $\{1e-5, 3e-5, 5e-5\}$, and the whole tuning process takes 30 epochs. In this part, we also randomly sample the data using five different seeds and report the average results.

To evaluate the effectiveness of PTR, we compare PTR with 5 latest intent classification models, including: (1) RoBERTa_base is the vanilla fine-tuning version based on RoBERTa_base implemented by Zhang et al. (2020a). (2) USE (Yang et al., 2020) is a large-scale multilingual model pre-trained on 16 languages for intent classification. (3) ConveRT (Casanueva et al., 2020) is a model pre-trained on conversational data using retrieval-based response selection task. (4) ConvBERT (Mehri et al., 2020) extends BERT by further fine-tuning on open-domain dialogue corpus and performing task-adaptive self-supervised training prior to fine-tuning on intent task. (5) CPFT (Zhang et al., 2021a)

Table 10
For some typical intents of CLINC150, BANKING77 and HWU64, their intent-specific label words for the PTR and PTINT templates. The PTR template is "$< S_1 >$. This sentence referred to $[M]_1$ is for $[M]_2$", and the PTINT template is "$< S_1 >$. This sentence is for $[M]_1$".

Dataset	Class Label	PTR		PTINT
		$[M]_1$	$[M]_2$	$[M]_1$
CLINC150	income	work	income	income
	transfer	banking	transfer	transfer
	text	utility	text	text
BANKING77	activate_my_card	credit/card	activating	card/activating
	card_linking	credit/card	linking	card/linking
	terminate_account	information	terminating	account/terminating
HWU64	alarm_query	alarm	alarm/query	alarm/query
	email_query	email	email/query	email/query
	datetime_query	datetime	datetime/query	datetime/query

Table 11
The accuracy scores (%) on the three datasets under the 5-shot and 10-shot settings. In the "Extra Data" column, "w/o" indicates that no additional data is used for pre-training and tuning, yet "w/" means that extra data is used for enhancing pre-training.

Model	Extra Data	CLINC150		BANKING77		HWU64	
		5	10	5	10	5	10
Fine-tuning methods							
Fine-Tuning (Zhang et al., 2021a)	w/o	88.0	91.6	74.0	84.3	75.6	82.9
USE (Yang et al., 2020)	w/	87.8	90.9	76.3	84.2	77.8	83.8
ConveRT (Casanueva et al., 2020)	w/	89.2	**92.6**	75.3	83.3	77.0	82.7
ConvBERT (Mehri et al., 2020)	w/	–	92.1	–	83.6	–	83.8
CPFT (Zhang et al., 2021a)	w/o	88.2	91.6	76.8	84.8	76.0	83.0
Prompt tuning methods							
PTINT	w/o	90.0	89.8	79.5	85.7	82.2	85.8
PTR	**w/o**	**90.2**	90.8	**79.7**	**86.0**	**82.6**	**85.9**

uses contrastive pre-training and fine-tuning for intent classification. Here we compare PTR with the CPFT version without contrastive pre-training.

Considering that prompt tuning has not been widely explored on intent classification, here we implement a simplified one mask version of PTR, named "PTINT". In Table 10, we show some details of the differences between the design of our intent-specific prompts and that in PTINT.

Comparison between PTR and baselines

Table 11 shows the performance of PTR and baseline models on the three datasets. From the table, we can find that:

(1) Both PTINT and PTR significantly outperform the vanilla fine-tuning method in all experimental scenarios, indicating the effectiveness of using prompt for tuning PLMs. Besides applying prompt tuning methods, enhancing pre-training with large-scale intent-related data can also significantly improve the performance of intent classification. Intuitively, prompt tuning aims to simulate task-specific knowledge from PLMs, yet these methods for enhancing pre-training focus on injecting more task-specific knowledge into PLMs.

(2) In most scenarios, prompt-based methods can achieve better results than these models for enhancing pre-training, indicating that much intent-specific knowledge is already stored in PLMs, how to take use of existing latent knowledge is important and promising. In fact, enhancing pre-training and performing prompt tuning together may bring better performance, we left this for our future work.

(3) Since intent types are less structured than entity types and relation types, PTINT has performed very well and the improvement over PTINT brought by PTR is weak. The overall results are still consistent with those on relation classification and entity typing and show the effectiveness of building prompts with rules. In this paper, the rules are built heuristically. Although these heuristics are effective enough, they may still not be optimal. It is worth exploring automatically finding better rules in the future.

4.4. *The convergence of PTR*

In order to further compare the learning efficiency of different prompt tuning methods, we select two datasets ReTACRED and FewNERD, and then give the performance curve of each model during the training process on these datasets.

As shown in Fig. 3, besides effectiveness, PTR can lead to faster convergence. Compared with other prompt tuning methods, although the templates finally presented by PTR are more complex and bring much regularization to the model, the convergence of PTR does not become slower.

On the one hand, the prompts of PTR can introduce more information, which is beneficial to stimulate the knowledge of PLMs in a more effective and efficient manner. On the other hand, let PLMs learn according to prior task knowledge, this setting is very close to curriculum learning (Bengio et al., 2009), which is also the reason for the better convergence of PTR. To a certain extent, all these results indicate the potential positive effect of prompt tuning and rules on convergence.

5. Related work

Recent PLMs like GPT (Radford et al., 2018), BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b) and T5 (Raffel et al., 2020) provide a new approach to utilize large-scale unlabeled data for NLP tasks. Although these PLMs can capture rich knowledge (Jawahar et al., 2019; Hewitt and Manning, 2019; Petroni et al., 2019; Yenicelik et al., 2020) from massive corpora, a fine-tuning process on task-specific data is still required to transfer PLMs' knowledge to downstream tasks. From dialogue (Zhang et al., 2020b), summarization (Zhang et al., 2019; Liu and Lapata, 2019), question answering (Adiwardana et al., 2020), to text classification (Baldini Soares et al., 2019; Peng et al., 2020; Ding et al., 2021c), fine-tuned PLMs have been demonstrated their effectiveness on various NLP tasks. Besides fine-tuning PLMs for specific tasks, recent studies have also explored better optimization

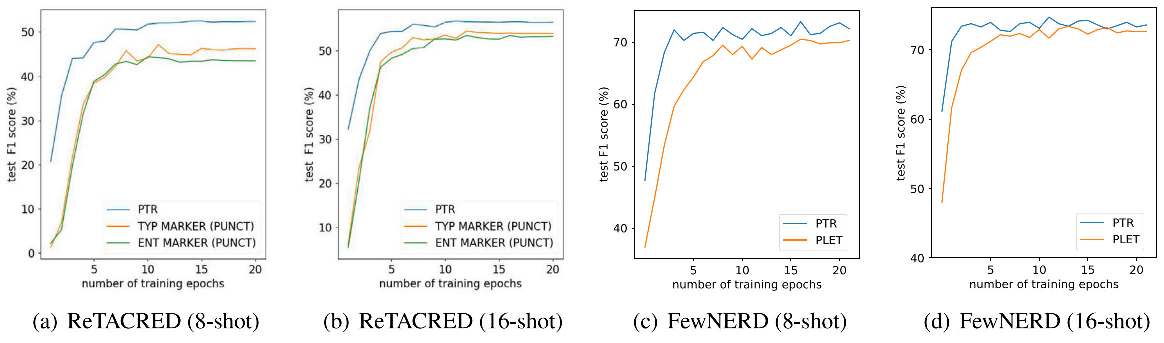

Fig. 3. Changes in F_1 scores (%) with increasing number of training epochs on the dataset ReTACRED and FewNERD.

and regularization techniques to improve fine-tuning (Lee et al., 2019; Dodge et al., 2020).

In the pre-training phase, sequential language modeling and masked language modeling are used to learn PLMs. In the fine-tuning phase, the optimization objectives are task-specific and different tasks may have quite different objective forms. In GPT-3 (Brown et al., 2020), prompts have been introduced and drawn much attention. By leveraging language prompts as contexts, downstream tasks can be expressed as some objectives similar to pre-training objectives. Through a series of research work on knowledge probing (Trinh and Le, 2018; Petroni et al., 2019; Davison et al., 2019), language prompts have been proven to effectively stimulate knowledge from PLMs. Moreover, human-picked prompts have achieved promising results on few-class classification tasks such as sentiment classification and natural language inference (Schick and Schütze, 2021; Liu et al., 2021b; Li et al., 2021).

To avoid labor-intensive prompt design, automatic prompt search has been extensively explored. As a typical prompt consists of two parts: a template and a set of label words, Schick et al. (2020), Schick and Schütze (2021) first explore automatic identification of label words for human-picked templates. Shin et al. (2020) further explore gradient-guided search to automatically generate both templates and label words. Gao et al. (2020) take seq-to-seq models to generate prompt candidates, and then use each of them for prompt tuning and verify their effectiveness on development sets. Compared with human-picked prompts, most auto-generated prompts cannot achieve comparable performance. Recently, some continuous prompts have also been proposed (Li and Liang, 2021; Lester et al., 2021), which directly use a series of learnable continuous embeddings as prompt templates and get rid of the trouble of designing prompts. These completely continuous prompts work well on those large-scale PLMs with billions of parameters, yet cannot stably work on normal PLMs. Some recent works further explore pre-training prompts (Wei et al., 2021; Sanh et al., 2021; Gu et al., 2021), which makes prompts perform better on downstream tasks.

For some many-class classification tasks, there are also some preliminary works using prompt-based methods, including methods for entity-related tasks (Ma et al., 2021; Chen et al., 2021b; Ding et al., 2021a) and information extraction tasks (Lin et al., 2021; Zhang et al., 2021b; Chen et al., 2021a). These prompt-based methods have obtained promising results, yet these methods focus more on using the language patterns learned by PLMs to design prompts, ignoring the rich prior knowledge of tasks such as the structured information about the class schema. In this paper, we explore to encode the rich prior knowledge into a few simple rules that can help handle classification tasks. Based on designing sub-prompts, PTR can combine sub-prompts into complete task-specific prompts based on the predefined rules. As compared with the above-mentioned prompt-based methods, PTR achieves a good balance among model efficiency, model effectiveness, model generalization, and human workload. Similar to PTR, some later prompt-based methods (Chen et al., 2022; Zhang et al., 2022) also utilize rules to enhance model performance.

6. Conclusion

In this paper, we propose prompt tuning with rules (PTR) for many-class text classification tasks. By composing sub-prompts into task-specific prompts according to rules, prior task knowledge can be encoded into prompt tuning. Meanwhile, the introduction of sub-prompts can further alleviate the difficulty of designing templates and label word sets. The experimental results on relation classification, entity typing, and intent classification show that PTR can outperform existing competitive vanilla fine-tuning and prompt tuning baselines without introducing any additional model layers, manual annotations, and augmented data. In the future, we will further combine the existing exploration of auto-generated prompts and pre-training augmentation with PTR, which will lead to a more practical approach.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is supported by the National Key Research and Development Program of China (No. 2020AAA0106500) and the National Natural Science Foundation of China (NSFC No. 62236004). Weilin Zhao is supported by the Tsinghua University Initiative Scientific Research Program.

References

Adiwardana, Daniel, Luong, Minh-Thang, So, David R., Hall, Jamie, Fiedel, Noah, Thoppilan, Romal, Yang, Zi, Kulshreshtha, Apoorv, Nemade, Gaurav, Lu, Yifeng, et al., 2020. Towards a human-like open-domain chatbot. arXiv preprint arXiv:2001.09977, URL https://arxiv.org/abs/2001.09977.

Alt, Christoph, Gabryszak, Aleksandra, Hennig, Leonhard, 2020. TACRED revisited: A thorough evaluation of the TACRED relation extraction task. In: Proceedings of ACL. pp. 1558–1569, URL https://www.aclweb.org/anthology/2020.acl-main.142.

Baldini Soares, Livio, FitzGerald, Nicholas, Ling, Jeffrey, Kwiatkowski, Tom, 2019. Matching the blanks: Distributional similarity for relation learning. In: Proceedings of ACL. pp. 2895–2905, URL https://www.aclweb.org/anthology/P19-1279.

Bengio, Yoshua, Louradour, Jérôme, Collobert, Ronan, Weston, Jason, 2009. Curriculum learning. In: Proceedings of ICML. pp. 41–48. http://dx.doi.org/10.1145/1553374.1553380.

Brown, Tom B., Mann, Benjamin, Ryder, Nick, Subbiah, Melanie, Kaplan, Jared, Dhariwal, Prafulla, Neelakantan, Arvind, Shyam, Pranav, Sastry, Girish, Askell, Amanda, et al., 2020. Language models are few-shot learners. In: Proceedings of NIPS. pp. 1877–1901, URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Casanueva, Iñigo, Temčinas, Tadas, Gerz, Daniela, Henderson, Matthew, Vulić, Ivan, 2020. Efficient intent detection with dual sentence encoders. In: Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI. pp. 38–45, URL https://aclanthology.org/2020.nlp4convai-1.5.

Chen, Xiang, Xie, Xin, Zhang, Ningyu, Yan, Jiahuan, Deng, Shumin, Tan, Chuanqi, Huang, Fei, Si, Luo, Chen, Huajun, 2021a. AdaPrompt: Adaptive prompt-based finetuning for relation extraction. arXiv preprint arXiv:2104.07650, URL https://arxiv.org/abs/2104.07650.

Chen, Xiang, Zhang, Ningyu, Li, Lei, Xie, Xin, Deng, Shumin, Tan, Chuanqi, Huang, Fei, Si, Luo, Chen, Huajun, 2021b. Lightner: A lightweight generative framework with prompt-guided attention for low-resource ner. arXiv preprint arXiv:2109.00720, URL https://arxiv.org/pdf/2109.00720.

Chen, Xiang, Zhang, Ningyu, Xie, Xin, Deng, Shumin, Yao, Yunzhi, Tan, Chuanqi, Huang, Fei, Si, Luo, Chen, Huajun, 2022. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In: Proceedings of WWW. pp. 2778–2788. http://dx.doi.org/10.1145/3485447.3511998.

Davison, Joe, Feldman, Joshua, Rush, Alexander M., 2019. Commonsense knowledge mining from pretrained models. In: Proceedings of EMNLP-IJCNLP. pp. 1173–1178, URL https://arxiv.org/abs/1909.00505.

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, Toutanova, Kristina, 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of NAACL-HLT. pp. 4171–4186, URL https://www.aclweb.org/anthology/N19-1423.pdf.

Ding, Ning, Chen, Yulin, Han, Xu, Xu, Guangwei, Xie, Pengjun, Zheng, Hai-Tao, Liu, Zhiyuan, Li, Juanzi, Kim, Hong-Gee, 2021a. Prompt-learning for fine-grained entity typing. arXiv preprint arXiv:2108.10604, URL https://arxiv.org/abs/2108.10604.

Ding, Ning, Hu, Shengding, Zhao, Weilin, Chen, Yulin, Liu, Zhiyuan, Zheng, Hai-Tao, Sun, Maosong, 2021b. Openprompt: An open-source framework for prompt-learning. arXiv preprint arXiv:2111.01998, URL https://arxiv.org/pdf/2111.01998.pdf.

Ding, Ning, Wang, Xiaobin, Fu, Yao, Xu, Guangwei, Wang, Rui, Xie, Pengjun, Shen, Ying, Huang, Fei, Zheng, Hai-Tao, Zhang, Rui, 2021c. Prototypical representation learning for relation extraction. In: Proceedings of ICLR. URL https://openreview.net/forum?id=aCgLmfhIy_f.

Ding, Ning, Xu, Guangwei, Chen, Yulin, Wang, Xiaobin, Han, Xu, Xie, Pengjun, Zheng, Haitao, Liu, Zhiyuan, 2021d. Few-NERD: A few-shot named entity recognition dataset. In: Proceedings of ACL-IJCNLP. pp. 3198–3213, URL https://aclanthology.org/2021.acl-long.248.

Dodge, Jesse, Ilharco, Gabriel, Schwartz, Roy, Farhadi, Ali, Hajishirzi, Hannaneh, Smith, Noah, 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. arXiv preprint arXiv:2002.06305, URL https://arxiv.org/abs/2002.06305.

Gao, Tianyu, Fisch, Adam, Chen, Danqi, 2020. Making pre-trained language models better few-shot learners. arXiv preprint arXiv:2012.15723, URL https://arxiv.org/pdf/2012.15723.

Gu, Yuxian, Han, Xu, Liu, Zhiyuan, Huang, Minlie, 2021. Ppt: Pre-trained prompt tuning for few-shot learning. arXiv preprint arXiv:2109.04332, URL https://arxiv.org/abs/2109.04332.

Han, Xu, Gao, Tianyu, Yao, Yuan, Ye, Deming, Liu, Zhiyuan, Sun, Maosong, 2019. OpenNRE: An open and extensible toolkit for neural relation extraction. In: Proceedings of EMNLP-IJCNLP. pp. 169–174. http://dx.doi.org/10.18653/v1/D19-3029, URL https://www.aclweb.org/anthology/D19-3029.

Hewitt, John, Manning, Christopher D., 2019. A structural probe for finding syntax in word representations. In: Proceedings of NAACL. pp. 4129–4138, URL https://www.aclweb.org/anthology/N19-1419".

Jawahar, Ganesh, Sagot, Benoît, Seddah, Djamé, 2019. What does BERT learn about the structure of language? In: Proceedings of ACL. pp. 3651–3657, URL https://www.aclweb.org/anthology/P19-1356.

Joshi, Mandar, Chen, Danqi, Liu, Yinhan, Weld, Daniel S., Zettlemoyer, Luke, Levy, Omer, 2020. Spanbert: Improving pre-training by representing and predicting spans. TACL 8, 64–77, URL https://www.aclweb.org/anthology/2020.tacl-1.5.

Larson, Stefan, Mahendran, Anish, Peper, Joseph J., Clarke, Christopher, Lee, Andrew, Hill, Parker, Kummerfeld, Jonathan K., Leach, Kevin, Laurenzano, Michael A., Tang, Lingjia, Mars, Jason, 2019. An evaluation dataset for intent classification and out-of-scope prediction. In: Proceedings of EMNLP-IJCNLP. pp. 1311–1316, URL https://aclanthology.org/D19-1131.

Lee, Cheolhyoung, Cho, Kyunghyun, Kang, Wanmo, 2019. Mixout: Effective regularization to finetune large-scale pretrained language models. In: Proceedings of ICLR. URL https://openreview.net/forum?id=HkgaETNtDB.

Lester, Brian, Al-Rfou, Rami, Constant, Noah, 2021. The power of scale for parameter-efficient prompt tuning. arXiv preprint arXiv:2104.08691, URL https://arxiv.org/abs/2104.08691.

Li, Chengxi, Gao, Feiyu, Bu, Jiajun, Xu, Lu, Chen, Xiang, Gu, Yu, Shao, Zirui, Zheng, Qi, Zhang, Ningyu, Wang, Yongpan, et al., 2021. Sentiprompt: Sentiment knowledge enhanced prompt-tuning for aspect-based sentiment analysis. arXiv preprint arXiv:2109.08306, URL https://arxiv.org/pdf/2109.08306.

Li, Xiang Lisa, Liang, Percy, 2021. Prefix-tuning: Optimizing continuous prompts for generation. arXiv preprint arXiv:2101.00190, URL https://arxiv.org/abs/2101.00190.

Lin, Jiaju, Jian, Jin, Chen, Qin, 2021. Eliciting knowledge from language models for event extraction. arXiv preprint arXiv:2109.05190, URL https://arxiv.org/pdf/2109.05190.

Liu, Xingkun, Eshghi, Arash, Swietojanski, Pawel, Rieser, Verena, 2019a. Benchmarking natural language understanding services for building conversational agents. arXiv preprint arXiv:1903.05566, URL https://arxiv.org/abs/1903.05566.

Liu, Yang, Lapata, Mirella, 2019. Text summarization with pretrained encoders. In: Proceedings of EMNLP. pp. 3730–3740, URL https://www.aclweb.org/anthology/D19-1387/.

Liu, Yinhan, Ott, Myle, Goyal, Naman, Du, Jingfei, Joshi, Mandar, Chen, Danqi, Levy, Omer, Lewis, Mike, Zettlemoyer, Luke, Stoyanov, Veselin, 2019b. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, URL https://arxiv.org/abs/1907.11692.

Liu, Pengfei, Yuan, Weizhe, Fu, Jinlan, Jiang, Zhengbao, Hayashi, Hiroaki, Neubig, Graham, 2021a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. arXiv preprint arXiv:2107.13586, URL https://arxiv.org/abs/2107.13586.

Liu, Xiao, Zheng, Yanan, Du, Zhengxiao, Ding, Ming, Qian, Yujie, Yang, Zhilin, Tang, Jie, 2021b. GPT understands, too. arXiv preprint arXiv:2103.10385, URL https://arxiv.org/abs/2103.10385.

Ma, Ruotian, Zhou, Xin, Gui, Tao, Tan, Yiding, Zhang, Qi, Huang, Xuanjing, 2021. Template-free prompt tuning for few-shot NER. arXiv preprint arXiv:2109.13532, URL https://arxiv.org/pdf/2109.13532.

Mehri, Shikib, Eric, Mihail, Hakkani-Tur, Dilek, 2020. Dialoglue: A natural language understanding benchmark for task-oriented dialogue. arXiv preprint arXiv:2009.13570, URL https://arxiv.org/abs/2009.13570.

Pan, Sinno Jialin, Yang, Qiang, 2009. A survey on transfer learning. TKDE 22 (10), 1345–1359, URL https://ieeexplore.ieee.org/document/5288526.

Peng, Hao, Gao, Tianyu, Han, Xu, Lin, Yankai, Li, Peng, Liu, Zhiyuan, Sun, Maosong, Zhou, Jie, 2020. Learning from context or names? An empirical study on neural relation extraction. In: Proceedings of EMNLP. pp. 3661–3672. http://dx.doi.org/10.18653/v1/2020.emnlp-main.298, URL https://www.aclweb.org/anthology/2020.emnlp-main.298.

Peters, Matthew E., Neumann, Mark, Logan, Robert, Schwartz, Roy, Joshi, Vidur, Singh, Sameer, Smith, Noah A., 2019. Knowledge enhanced contextual word representations. In: Proceedings of EMNLP-IJCNLP. pp. 43–54, URL https://www.aclweb.org/anthology/D19-1005.

Petroni, Fabio, Rocktäschel, Tim, Riedel, Sebastian, Lewis, Patrick, Bakhtin, Anton, Wu, Yuxiang, Miller, Alexander, 2019. Language models as knowledge bases? In: Proceedings of EMNLP. pp. 2463–2473, URL https://www.aclweb.org/anthology/D19-1250.pdf.

Qiu, Xipeng, Sun, Tianxiang, Xu, Yige, Shao, Yunfan, Dai, Ning, Huang, Xuanjing, 2020. Pre-trained models for natural language processing: A survey. Sci. China Technol. Sci. 1–26, URL https://arxiv.org/abs/2003.08271.

Radford, Alec, Narasimhan, Karthik, Salimans, Tim, Sutskever, Ilya, 2018. Improving language understanding by generative pre-training. OpenAI URL https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf.

Raffel, Colin, Shazeer, Noam, Roberts, Adam, Lee, Katherine, Narang, Sharan, Matena, Michael, Zhou, Yanqi, Li, Wei, Liu, Peter J., 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. JMLR 21, 1–67, URL https://www.jmlr.org/papers/volume21/20-074/20-074.pdf.

Ren, Xiang, Wu, Zeqiu, He, Wenqi, Qu, Meng, Voss, Clare R., Ji, Heng, Abdelzaher, Tarek F., Han, Jiawei, 2017. Cotype: Joint extraction of typed entities and relations with knowledge bases. In: Proceedings of WWW. pp. 1015–1024, URL http://delivery.acm.org/10.1145/3060000/3052708/p1015-ren.pdf?ip=183.173.78.194&id=3052708&acc=ACTIVE%20SERVICE&key=BF85BBA5741FDC6E%2E587F3204F5B62A59%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&_acm_=1557445685_65e90f7a9015ba328f00d325f207878e.

Sanh, Victor, Webson, Albert, Raffel, Colin, Bach, Stephen H., Sutawika, Lintang, Alyafeai, Zaid, Chaffin, Antoine, Stiegler, Arnaud, Scao, Teven Le, Raja, Arun, et al., 2021. Multitask prompted training enables zero-shot task generalization. arXiv preprint arXiv:2110.08207, URL https://arxiv.org/pdf/2110.08207.

Schick, Timo, Schmid, Helmut, Schütze, Hinrich, 2020. Automatically identifying words that can serve as labels for few-shot text classification. In: Proceedings of COLING. pp. 5569–5578. http://dx.doi.org/10.18653/v1/2020.coling-main.488, URL https://www.aclweb.org/anthology/2020.coling-main.488.

Schick, Timo, Schütze, Hinrich, 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In: Proceedings of EACL. pp. 255–269, URL https://www.aclweb.org/anthology/2021.eacl-main.20.

Shimaoka, Sonse, Stenetorp, Pontus, Inui, Kentaro, Riedel, Sebastian, 2017. Neural architectures for fine-grained entity type classification. In: Proceedings of EACL. pp. 1271–1280, URL https://aclanthology.org/E17-1119.

Shin, Taylor, Razeghi, Yasaman, Logan IV, Robert L., Wallace, Eric, Singh, Sameer, 2020. Eliciting knowledge from language models using automatically generated prompts. In: Proceedings of EMNLP. pp. 4222–4235, URL https://www.aclweb.org/anthology/2020.emnlp-main.346.pdf.

Stoica, George, Platanios, Emmanouil Antonios, Póczos, Barnabás, 2021. Re-TACRED: Addressing shortcomings of the TACRED dataset. arXiv preprint arXiv:2104.08398, URL https://arxiv.org/abs/2104.08398.

Trinh, Trieu H., Le, Quoc V., 2018. A simple method for commonsense reasoning. arXiv preprint arXiv:1806.02847, URL https://arxiv.org/abs/1806.02847.

Wei, Jason, Bosma, Maarten, Zhao, Vincent Y., Guu, Kelvin, Yu, Adams Wei, Lester, Brian, Du, Nan, Dai, Andrew M., Le, Quoc V., 2021. Finetuned language models are zero-shot learners. arXiv preprint arXiv:2109.01652, URL https://arxiv.org/pdf/2109.01652.

Weischedel, Ralph, Brunstein, Ada, 2005. BBN pronoun coreference and entity type corpus. In: Linguistic Data Consortium. URL https://catalog.ldc.upenn.edu/LDC2005T33.

Weischedel, Ralph, Palmer, Martha, Marcus, Mitchell, Hovy, Eduard, Pradhan, Sameer, Ramshaw, Lance, Xue, Nianwen, Taylor, Ann, Kaufman, Jeff, Franchini, Michelle, El-Bachouti, Mohammed, Belvin, Robert, Houston, Ann, 2013. OntoNotes release 5.0. In: Abacus Data Network. URL https://hdl.handle.net/11272.1/AB2/MKJJ2R.

Wolf, Thomas, Chaumond, Julien, Debut, Lysandre, Sanh, Victor, Delangue, Clement, Moi, Anthony, Cistac, Pierric, Funtowicz, Morgan, Davison, Joe, Shleifer, Sam, et al., 2020. Transformers: State-of-the-art natural language processing. In: Proceedings of EMNLP. pp. 38–45, URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.pdf.

Yamada, Ikuya, Asai, Akari, Shindo, Hiroyuki, Takeda, Hideaki, Matsumoto, Yuji, 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In: Proceedings of EMNLP. pp. 6442–6454, URL https://www.aclweb.org/anthology/2020.emnlp-main.523.

Yang, Yinfei, Cer, Daniel, Ahmad, Amin, Guo, Mandy, Law, Jax, Constant, Noah, Hernandez Abrego, Gustavo, Yuan, Steve, Tar, Chris, Sung, Yun-hsuan, Strope, Brian, Kurzweil, Ray, 2020. Multilingual universal sentence encoder for semantic retrieval. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations. pp. 87–94, URL https://aclanthology.org/2020.acl-demos.12.

Yenicelik, David, Schmidt, Florian, Kilcher, Yannic, 2020. How does BERT capture semantics? A closer look at polysemous words. In: Proceedings of BlackboxNLP. pp. 156–162, URL https://www.aclweb.org/anthology/2020.blackboxnlp-1.15.

Zhang, Jianguo, Bui, Trung, Yoon, Seunghyun, Chen, Xiang, Liu, Zhiwei, Xia, Congying, Tran, Quan Hung, Chang, Walter, Yu, Philip, 2021a. Few-shot intent detection via contrastive pre-training and fine-tuning. In: Proceedings of EMNLP. pp. 1906–1912, URL https://aclanthology.org/2021.emnlp-main.144.

Zhang, Jianguo, Hashimoto, Kazuma, Liu, Wenhao, Wu, Chien-Sheng, Wan, Yao, Yu, Philip, Socher, Richard, Xiong, Caiming, 2020a. Discriminative nearest neighbor few-shot intent detection by transferring natural language inference. In: Proceedings of EMNLP. pp. 5064–5082, URL https://aclanthology.org/2020.emnlp-main.411.

Zhang, Ningyu, Li, Luoqiu, Chen, Xiang, Deng, Shumin, Bi, Zhen, Tan, Chuanqi, Huang, Fei, Chen, Huajun, 2021b. Differentiable prompt makes pre-trained language models better few-shot learners. arXiv preprint arXiv:2108.13161, URL https://arxiv.org/pdf/2108.13161.

Zhang, Yuhao, Qi, Peng, Manning, Christopher D., 2018. Graph convolution over pruned dependency trees improves relation extraction. In: Proceedings of EMNLP. pp. 2205–2215, URL http://aclweb.org/anthology/D18-1244.

Zhang, Yizhe, Sun, Siqi, Galley, Michel, Chen, Yen-Chun, Brockett, Chris, Gao, Xiang, Gao, Jianfeng, Liu, Jingjing, Dolan, Bill, 2020b. Dialogpt: Large-scale generative pre-training for conversational response generation. In: Proceedings of ACL. pp. 270–278, URL https://arxiv.org/abs/1911.00536.

Zhang, Rongzhi, Yu, Yue, Shetty, Pranav, Song, Le, Zhang, Chao, 2022. Prompt-based rule discovery and boosting for interactive weakly-supervised learning. In: Proceedings of ACL. pp. 745–758, URL https://arxiv.org/pdf/2203.09735.pdf.

Zhang, Jingqing, Zhao, Yao, Saleh, Mohammad, Liu, Peter J., 2019. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In: Proceedings of ICML. pp. 11328–11339, URL https://arxiv.org/abs/1912.08777.

Zhang, Yuhao, Zhong, Victor, Chen, Danqi, Angeli, Gabor, Manning, Christopher D., 2017. Position-aware attention and supervised data improve slot filling. In: Proceedings of EMNLP. pp. 35–45, URL https://nlp.stanford.edu/pubs/zhang2017tacred.pdf.

Zhou, Wenxuan, Chen, Muhao, 2021. An improved baseline for sentence-level relation extraction. arXiv preprint arXiv:2102.01373, URL https://arxiv.org/abs/2102.01373.