A Cueing Strategy for Prompt Tuning in Relation Extraction

Anonymous ACL submission

Abstract

Traditional relation extraction models predict confidence scores for each relation type based on a condensed sentence representation. In prompt tuning, prompt templates is used to tune pre-trained language models (PLMs), which outputs relation types as verbalized type tokens. This strategy shows great potential to support relation extraction because it is effective to take full use of rich knowledge in PLMs. However, current prompt tuning models are directly implemented on a raw input. It is weak to encode contextual features and semantic dependencies of a relation instance. In this paper, we designed a cueing strategy which implants task specific cues into the input. It controls the attention of prompt tuning, which enable PLMs to learn task specific contextual features and semantic dependencies of a relation instance. We evaluated our method on two public datasets. Experiments show great improvement. It exceeds state-of-the-art performance by more than 4.8% and 1.4% in terms of F1-score on the SemEval corpus and the ReTACRED corpus[1].

1 Introduction

Relation extraction (RE) identifies predefined semantic relationships between two named entities in a sentence. It is a specific information extraction task characterized by two properties. First, extracting entity relations depends on global features of a sentence, but a sentence usually contains several named entities. Second, relation types are asymmetric. The order of entities in a relation instance should be considered. Therefore, relation extraction should verify all entity pairs in a sentence, which leads to a serious data imbalance problem. Because all relation instances share the same context, it is important to learn contextual features and semantic dependencies relevant to considered entities. In deep neural networks, many techniques have been developed to do so, for example, position embedding (Zeng et al., 2015), multi-channel (Chen et al., 2020), neuralized feature engineer (Chen et al., 2021) and entity indicators (Qin et al., 2021; Zhou and Chen, 2021). These models are common in that entities relevant features (e.g., entity positions or types) are encoded into a task specific representation. Then it is fed into a deep architecture for classification, which outputs confidence scores for every relation type.

In prompt tuning, prompts are defined as templates with slots that take values from a verbalized type token set. These prompts are concatenated with an input (a sentence), then fed into PLMs to predict masked slots, the same as a cloze-style schema (Schick and Schütze, 2020). Because prompt tuning is effective to take use of knowledge within PLMs, it has been successfully applied in tasks such as text classification and natural language inference (Schick and Schütze, 2020). For example, in semantic recognition, an input is first concatenated with a prompt (e.g., "It was [MASK]"). Then, it is fed into PLMs for predicting the masked token (e.g., *Glad* or *Sad*). In relation extraction, let e_1 and e_2 represent two named entities. A "person:parent" relation can be identified by a prompt template "the [MASK] e_1 [MASK] the [MASK] e_2" (Han et al., 2021). If a PLM output three type tokens as "person", "is parent of" and "person" respectively, it indicates a "person:parent" relation between $e_1$and e_2.

In traditional type classification models, PLMs are mainly used to support token embedding. A deep architecture is usually designed to compress every relation instance into an abstract representation (a vector in common). The classification only depends on a single representation of the whole input, which undoubtedly results in a serious semantic loss. Furthermore, the process to initialize PLMs is implemented as a masked token prediction task (Devlin et al., 2018). There is a gap be-

[1]Our codes to implement the cueing strategy will be available online.

tween pre-training objectives and fine tuning objectives. On the other hand, the effectiveness of prompt tuning is heavily depends on the quality of prompt templates. In related work, many prompts have been designed for PLMs tuning (Brown et al., 2020). However, current prompt tuning models are often directly implemented on a raw input concatenated with predefined prompt templates. Rare work has been done to tune PLMs for learning task specific features about considered entities.

As discussed above, due to the properties of relation extraction, it is very important to learn contextual features and semantic dependencies relevant to considered entities. Motivated by techniques developed in type classification and prompt tuning, in this paper, we designed a cueing strategy which implants task specific cues into the input. It controls the attention of prompt tuning to learn task specific contextual features and semantic dependencies in a sentence. By combining the cueing strategy with prompt tuning, it enables PLMs encoding semantic dependencies between type tokens and contextual words. Furthermore, the predicting process is similar as that of PLMs tuning, it is helpful to bridge the gap between PLMs and relation extraction. Our study shows remarkable improvement in the performance. It reveals a meaningful mechanism that is essential for relation extraction and prompt tuning. Contributions of this paper are listed as follows:

1 Several cueing strategies are designed for tuning PLMs. It is effective to reinforce the attention of neural networks to learn task specific features.

2 Our method is evaluated on two public datasets. Experiments are also conducted to analyse the attention mechanism of cueing for relation extraction.

The remainder of this paper is organized as follows. Section 2 introduces related work. The cueing strategy is presented in Section 3. Section 4 conducts experiments to evaluate our cueing strategy. The conclusion is given in Section 5.

2 Related Work

Relation extraction has always been regarded as a classification problem. In the early stage, shallow architectures are widely used (Zhao and Grishman, 2005; Chen et al., 2015). Because manually designed rules are required to extract features of a relation instance, these models are expensive in human labour and the migration between different domains is difficult. On the other hand, deep architectures adopt multi-stacked network layers implementing designed feature transformation, e.g, convolutional networks (Zeng et al., 2014; Nguyen and Grishman, 2015) or recurrent networks (Zhou et al., 2016; Wang et al., 2016). They have the advantage to automatically extract high order abstract representation from raw input.

For learning better relation representations, PLMs (e.g., ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018)) are widely adopted for embedding tokens into distributed representations. Therefore, in relation extraction, PLMs tuning has achieved great success(Torfi et al., 2020). PLMs usually consist of billions of parameters automatically learned from external resources. They encode rich knowledge of sentences that are valuable for downstream tasks (Brown et al., 2020). Therefore, in the training process, PLMs are tuned with annotated examples for learning task relevant representations. In this field, there are two paradigm to tune PLMs: fine tuning and prompt tuning.

In fine-tuning paradigm, PLMs are used to map every token into a distributed representation, e.g., BERT (Devlin et al., 2018), ALBERT (Lan et al., 2019), RoBERTa (Liu et al., 2019). Because PLMs are pre-trained from external resources with unsupervised methods, they are effective to relief the feature sparsity problem (Soares et al., 2019; Li and Tian, 2020). Based on PLMs, Zhao et al. (2021) proposed a graph neural network (GNN) for joint entity and relation extraction. Chen et al. (2021) combined neural network with feature engineering and proposed a neuralized feature engineering method. Cohen et al. (2020) used the schema of question answering to verify the feasibility of relation extraction. Lyu et al. (2021) proposed an entity type restriction, where the entity types are exploited to restrict candidate relations.

Prompt tuning has received considerable attention in recent years, and has achieved great success (Liu et al., 2021). In this paradigm, relation extraction is implemented as a mask language model, which involves to two issues: template designing and verbalizer constructing. In related work, Han et al. (2021) proposed a PTR model, which applies logic rules to construct prompts with several sub-prompts. It is able to encode prior knowledge of each class into prompt tuning. Shin et al.

(2020) proposed a gradient-guided methods to create prompts automatically. Gao et al. (2020) presented a prompt model, which take sequence-to-sequence models to generate prompt candidates. Chen et al. (2020) proposed a knowledge-aware prompt-tuning approach. It jointly optimize the representation of a virtual prompt template and answer words with knowledge constraints.

3 Methodology

To provide a formalized discussion, the task of extracting entity relations is formalized as follows:

A relation instance is defined as a 3-tuple $I = \langle r, e_1, e_2 \rangle$, which contains a relation mention r and two named entities e_1 and e_2. Relation mention r is a token sequence $r = [t_1, t_2, \cdots, t_n]$. Entities $e_k = [t_i, \cdots, t_j]$ ($k \in \{1, 2\}$) is a substring of r. Let $\mathbf{Y} = \{y_0, y_1, \cdots, y_M\}$ be a relation type set. It is composed of M positive relation types and one negative relation types y_0. Let $\mathbf{I} = \{I_1, I_2, \cdots\}$ represent a relation instance set. Then, relation extraction is represented as a map between $\mathbf{I}$ and $\mathbf{Y}$, denoted as:

$$f : \mathbf{I} \rightarrow \mathbf{Y} \tag{1}$$

where f is a function which can be a shallow model (e.g., a support vector machine, a maximum entropy classifier) or a deep neural network (e.g., a convolutional network or a recurrent network).

3.1 Classification Paradigm

In a traditional model, a deep architecture (denoted as $\mathcal{N}$) is implemented on the original input r to extract its representation. To encode external knowledge, the network $\mathcal{N}$ can be embedded with a PLM to support token embedding. It is denoted as $\mathcal{N}_{\mathcal{M}}$. The output of $\mathcal{N}_{\mathcal{M}}$ is represented as $\mathbf{H} = [H_1, H_2, \cdots, H_n]$, where H_i is an abstract representation of token t_i. $\mathbf{H}$ is often transformed into a vector, then fed into a classifier ($\mathcal{C}$) to make a prediction. The process is formalized as:

$$P(\mathbf{Y}|I) = Softmax\Big(\mathcal{C}\Big(\mathcal{N}_{\mathcal{M}}(r)\Big)\Big) \tag{2}$$

Directly implementing a deep network on r usually cause serious performance degradation, because the network know nothing about the position of considered entities. To handle this problem, Qin et al. (2021) and Zhong et al. (2020) implant entity cues to the input to control the attention of a deep network for learning task specific representation. It is formalized as:

$$\begin{aligned} Cueing(e_k) &= [\langle c_k \rangle, e_k, \langle / c_k \rangle] \\ Cueing(r) &= [\ddot{r}|_{e_k/Cueing(e_k), k=\{1,2\}}] \end{aligned} \tag{3}$$

where, $\langle c_k \rangle$ and $\langle / c_k \rangle$ are specific tokens representing the start and end boundaries of entity e_k ($k = \{1, 2\}$). They are named as entity cues.

In Equation (3), the first equation concatenates two tokens on both sides of e_k. In the second equation, $e_k/Cueing(e_k)$ denotes to the string replacement operation, where e_k is replaced by $Cueing(e_k)$. Therefore, the function $Cueing(r)$ implant entity cues into both side of the considered entity pair. With this settings, Equation (2) can be revised as:

$$P(\mathbf{Y}|I) = Softmax\Big(\mathcal{C}\Big(\mathcal{N}_{\mathcal{M}}\big(Cueing(r)\big)\Big)\Big) \tag{4}$$

In the above equation, after entity cues have been implanted into the input, it enables the deep network focusing on considered entity pair. Then, the classification is based on a sentence representation relevant to considered entities. It is effective to learn contextual features and semantic dependencies of a relation instance.

3.2 Prompt Tuning Paradigm

In prompt tuning, class types are verbalized into a token set $\mathbf{V} = \{person, parent, true, \cdots\}$. It is composed of entity types, relation types or category labels (e.g., "true" or "false"). Elements of $\mathbf{V}$ are referred as "type tokens". Then, a prompt is defined as a template with slots can be filled by verbalized type tokens (e.g., "It is a [MASK]"). It is concatenated with a raw input and fed into a deep network for predicting the distribution of type tokens in the position of " [MASK]".

The design of prompt templates heavily depends on the property of a task. At current, it is an art instead of a science. In this paper, we follow the work of Han et al. (Han et al., 2021), where a relation prompt is defined as a template with three slots: "$\mathcal{P}(e_1, e_2)$ = the [MASK]$_1$ e_1 is [MASK]$_2$ to [MASK]$_3$ e_2", where, [MASK] takes values from $\mathbf{V}$. The prompt is concatenated with the input and fed into a deep neural network to learn token representations $\mathbf{H}$. It is represented as:

$$[H_1, \cdots, H_L] = \mathcal{N}_{\mathcal{M}}\big(Cueing(r) + \mathcal{P}(e_1, e_2)\big) \tag{5}$$

In prompt tuning, instead of outputting a class label based on token representations $[H_1, \cdots, H_L]$, for each slot ([MASK]) in a prompt template, the normalized confidence score that $\mathcal{N}_{\mathcal{M}}$ assigns a type token $v \in \mathbf{V}$ to [MASK]$_i$ is computed as:

$$\mathcal{S}([MASK] = v|I) = H_v \cdot H_{M_i} \quad (6)$$

where, $H_{M_i} \in \mathbf{H}$ is the representation of a [MASK]$_i$ and H_v is the token type representation of $v \in \mathbf{V}$ in the employed PLMs. Then, given a relation instance I, the distribution of type token v in slot [MASK]$_i$ is computed as:

$$P(v|I) = \frac{\exp\Big(\mathcal{S}\big(\text{[MASK]}=v|I\big)\Big)}{\sum_{v' \in \mathbf{V}} \exp\Big(\mathcal{S}\big(\text{[MASK]}=v'|I\big)\Big)} \quad (7)$$

In prompt tuning, a right output requires that three slots are correct recognized. Because prompt tuning is effective to use rich knowledge in PLMs, it still show competitive performance.

3.3 Cueing Strategies

As discussed in Section 1, in relation extraction, it is very important to learn contextual features and semantic dependencies relevant to considered entities (Han et al., 2021; Chen et al., 2021; Zhong and Chen, 2020). In prompt tuning, researches are mainly focusing on designing prompt templates to tune PLMs for downstream tasks (Brown et al., 2020). We assume that, in prompt tuning, tuning PLMs attents to task specific information is also valuable. Instead of designing new prompt templates, we focus on designing and implanting entity cues for tuning PLMs to support relation extraction. Several cueing strategies have been proposed in this paper. They are listed as follows:

Cueing Types	Demonstration
$Cueing_o(e_k)$	$[e_k]$
$Cueing_a(e_k)$	$[\langle c_k\rangle, e_k, \langle /c_k\rangle]$
$Cueing_e(e_k)$	$[\{(entity)\ e_1\}]$, $[\{(entity)\ e_2\}]$
$Cueing_{ht}(e_k)$	$[\{(head)\ e_1\}]$, $[\langle\lceil tail\rfloor\ e_2\rangle]$

Table 1: Cueing Strategies

In Table 1, square brackets is used to indicate that the inner is a token sequence. $Cueing_o$ means that the input relation mention is unchanged. $Cueing_a$ replaces entity e_k ($k \in \{1, 2\}$) with a token sequence "$\langle c_k\rangle, e_k, \langle /c_k\rangle$". This is a traditional strategy used in related work (Chen et al., 2021; Qin et al., 2021). In $Cueing_e(e_k)$, each entity (e_k) in a relation mention is replaced by a string "$\{(entity)\ e_k\}$". Note that all pairs of closed braces and parentheses are also used as tokens to indicate the position of named entities. In $Cueing_{ht}(e_k)$, a "head" token and a "tail" token with different braces are used to distinguish different entities. In our experiments, we found that, in the latter two cueing strategies, the token "entity", "head" and "tail" and braces can be any words or braces. It has little influence on the final performance. But in $Cueing_{ht}(e_i)$, for different entities, the tokens and braces should be different.

Entity cues are implanted into the input. Then, the revised input is concatenated with prompt templates to tune PLMs for relation extraction. In Figure 1, we give examples to demonstrate the cueing strategy.

In Figure 1, "PTR prompt" is the prompt template proposed by Han et al. (Han et al., 2021), in which a template has three slots. [MASK]$_1$ and [MASK]$_3$ can take values in {"person", "country", $\cdots$ }. They denote to the type of named entities. [MASK]$_2$ takes values in {"was born in", "was located in", $\cdots$ }. It is used to indicate the relation between named entities. In "Naive prompt", three [MASK] are directly used without any contextual words. It is mainly used for comparison.

The cueing strategies listed in Table 1 are concatenated with both "PTR prompt" and "Naive prompt", where $\oplus$ denotes to the concatenating operation. For example, "$Cueing_{ht}(e_k)$+PTR" means that, given a relation instance $\langle r, e_1, e_2\rangle$, we first replace e_1 and e_2 in r by two string "$\{(head)\ e_1\}$" and "$\langle\lceil tail\rfloor\ e_2\rangle$". Then, the revised relation mention ($\ddot{r}|_{e_k/Cueing_{ht}(e_k), k=\{1,2\}}$) is concatenated with the PTR prompt. The output is fed into a PLM to predict type tokens in each [MASK]. If a PLM outputs "person", "is parent of", "person", then a "person:parent" relation is identified between e_1 and e_2.

4 Experiments

In this section, our strategies are verified on two popular evaluation datasets. Then, it is compared with several SOTA models. Experiments are also conducted to show the advantage of cueing strategies in few-shot learning.

4.1 Datasets and Experimental Settings

Our experiments are conducted on two evaluation datasets: SemEval 2010 Task 8 (Hendrickx et al., 2019) and ReTACRED (Stoica et al., 2021).

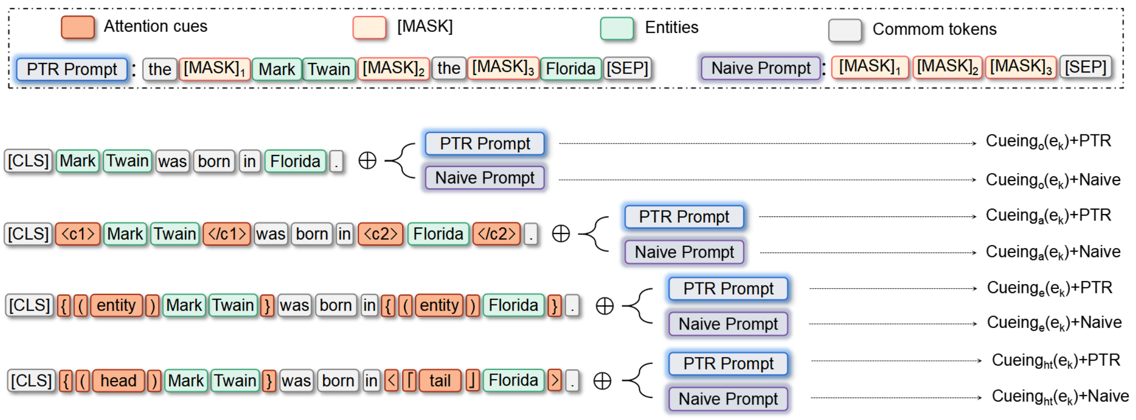

Attention cues
[MASK]
Entities
Common tokens
PTR Prompt : the [MASK]$_1$ Mark Twain [MASK]$_2$ the [MASK]$_3$ Florida [SEP]
Naive Prompt : [MASK]$_1$ [MASK]$_2$ [MASK]$_3$ [SEP]
[CLS] Mark Twain was born in Florida .
[CLS] <c1> Mark Twain </c1> was born in <c2> Florida </c2> .
[CLS] { (entity) Mark Twain } was born in { (entity) Florida } .
[CLS] { (head) Mark Twain } was born in ⟨ ⌈ tail ⌋ Florida ⟩ .
PTR Prompt
Naive Prompt
Cueing$_o$(e$_k$)+PTR
Cueing$_o$(e$_k$)+Naive
Cueing$_a$(e$_k$)+PTR
Cueing$_a$(e$_k$)+Naive
Cueing$_e$(e$_k$)+PTR
Cueing$_e$(e$_k$)+Naive
Cueing$_{ht}$(e$_k$)+PTR
Cueing$_{ht}$(e$_k$)+Naive

Figure 1: Examples of Cueing Strategies

The SemEval corpus is a classic and widely used dataset, which contains 8000 train data and 2717 test data, annotated with 19 relation types. The ReTACRED is a large relation extraction dataset, annotated with 40 relation types. ReTACRED officially divided the whole relation instances into 58465, 19584 and 13418 for training, developing and testing.

In our experiments, the RoBERTa$_{LARGE}$ (Liu et al., 2019) is adopted as our PLMs. The max length for each input is set as 128. The "Adam" is used as the model optimizer. The dropout is set to 0.1 to avoid the overfitting. Epochs, learning rate and batch size are set as 20, 2e-5, 32, respectively. To compare with related work, we follow experiment settings in Han et al. (2021). All performance is reported micro-averaged F1-score on positive relation types.

4.2 Comparing with Related Work

In this experiment, we adopt the $Cueing_{ht}(e_k)$ strategy in Table 1, where entities e_1 and e_2 in a relation mention are replaced by string "$\{(head)\ e_1\}$" and "$\langle\lceil tail\rfloor\ e_2\rangle$", respectively. The revised relation mention is concatenated with the PTR prompt. An example is illustrated in Figure 1. Every concatenated string is fed into a PLM to predict type tokens in masked slots. Our cueing strategy is compared with several related work. The first four are fine tuning models. The latter two are prompt tuning models. They are introduced as follows.

MTB (Soares et al., 2019): This model takes in pairs of relation statements in which two entity mentions are replaced by blanks. In the training process, relation representations are learned to be similar to themselves if these statements range over the same pairs of entities.

R-BERT (Wu and He, 2019): This model incorporates information from two considered entities to support the relation extraction task. It first locates considered entities. Then, encodings of entities are transferred through a pre-trained architecture for classification.

REDN (Li and Tian, 2020): This model proposes a special loss function to serve as a downstream task of PLMs for supervised relation extraction. It has the advantage to extract complicated relations, such as long-distance relation or overlapped relations on entity-pairs.

QA (Cohen et al., 2020): This is a span-prediction based system. In stead of using a single embedding to represent the relation, the extraction task is implemented as a span-prediction problem, the same as a Question Answering model.

PTR (Han et al., 2021): It is a prompt tuning based model, which applies logic rules to encode prior knowledge about the relation extraction task. In the ReTACRED, they reversed some relation types for predicting. To make a fair comparison, we only list their performance reported on unchanged dataset.

KnowPrompt (Xiang et al., 2020): This model focuses on incorporating knowledge into prompt tuning. It presents a knowledge-ware prompt-tuning approach with synergistic optimization and learnable template words.

Table 2 gives the performance of our strategy and related work. All performance is reported in F1 score (%).

Unlike fine tuning that outputs a class label for each input, in prompt tuning models, extracting a relation requires three type tokens to be correctly identified. In prompt tuning, the process to extract

Tuning	Methods	SemEval	ReTACRED
Fine	MTB	89.2	-
	R-BERT	89.25	-
	REDN	91.0	-
	QA	91.9	-
Prompt	PTR	89.9	90.9
	KnowPrompt	90.1	89.8
	Ours	**96.77**	**92.36**

Table 2: Comparing with Related Work

relation is the same as to train PLMs, it is helpful to bridge the gap between PLMs training and downstream tasks. As Table 2 showing, prompt tuning still achieves competitive performance. Furthermore, prompt tuning outputs type tokens based contextual features and semantic dependencies of a sentence, it is effective to take full use of rich knowledge in PLMs.

ReTACRED has a lot of relation types (40 relation types). Because the corpus contains a large number of training data, comparing with the SemEval corpus (19 relation types), it also achieved competitive performance. For related work in the SemEval corpus, fine tuning model REDN and QA has achieved the best performance. The reason is that both the REDN and QA models also address the gap between PLMs and downstream tasks. Therefore, they are also effective to use knowledge in PLMs.

Comparing with related work, our model achieves the state-of-the-art performance. It considerably outperforms related working exceeding state-of-the-art performance by more than 4.8% and 1.4% in terms of F1-score on the SemEval corpus and the ReTACRED corpus, respectively. The result shows that, instead of directly implementing prompt tuning on the raw input, implanting entity cues is valuable to support relation extraction. The conclusion reveals the mechanism of prompt tuning. It is significant to support future studies on both relation extraction and prompt tuning.

4.3 Ablation Study

Several cueing strategies have been presented in Table 1. In order to demonstrate the effectiveness of cueing strategies, we combine them with the naive prompt and PTR prompt to show the influence of cueing strategies on the performance. The sentence “Mark Twain was born in Florida" is used as an example to illustrate the combinations, in which "Mark Twain" is person entity and "Florida" is location entity. There is a “was born in” relation between them. The performance of different combinations was listed in Table 4. In the following, each of them is discussed separately.

(1) In Cueing$_o(e_k)$+Naive, every original input is directly concatenated with a naive prompt, which is composed with only tree masked token without any contextual words. This combination is mainly conducted for comparison. It can be seen as the baseline of prompt tuning. Because there is no “fixed” word in both the input and prompt, it is difficult to learn semantic dependencies between them. The result shows that it achieved the worst performance, especially in the ReTACRED corpus.

(2) Cueing$_o(e_k)$+PTR is the strategy used in Han et al. (2021). Every input is directly concatenated with the PTR prompt and fed into a PLM for predicting type tokens that can be filled in template slots. Because the PRT prompt contains contextual words, it encodes semantic information about the prompt which is helpful to take use of PLMs. Comparing with its naive version, it considerably improves the performance.

(3) In Cueing$_a(e_k)$+Naive, entity cues proposed in Chen et al. (2021) are implanted into the input to indicate the position of entities e_i ($i \in \{1,2\}$). The performance is clearly improved in the SemEval corpus, in which, comparing with (1), entity cues improve the performance above 10% in F1 score. Comparing with related work in Table 2, it already achieved the state of the art performance. The result indicates that entity cues are every powerful in prompt tuning based models. In the ReTACRED corpus, comparing (2) and (3) with (1), we can see that both entity cues and prompts are valuable to support relation extraction.

(4) Cueing$_a(e_k)$+PTR also outperforms its naive version. The different between (3) and (4) is that the PTR prompt contains contextual words. They are meaningful to encode semantic information of the prompt. It also enables PLMs encoding semantic dependencies between type tokens and contextual words.

(5) Cueing$_e(e_k)$ is a novel entity cues proposed in this paper. In this strategy, each e_k ($k \in \{1,2\}$) is replaced by a string $\{(entity)e_k\}$, where all braces and parentheses are also used as tokens. They are used to indicate the position of named entities. In this setting, e_1 and e_2 use the same

ID	Cueing+Prompt	Example	SemEval	ReTACRED
(1)	Cueing$_o(e_k)$+Naive	[CLS] Mark Twain was born in Florida. [MASK]$_1$ [MASK]$_2$ [MASK]$_3$ [SEP]	82.77	43.37
(2)	Cueing$_o(e_k)$+PTR	[CLS] Mark Twain was born in Florida. the [MASK]$_1$ Mark Twain [MASK]$_2$ [MASK]$_3$ Florida [SEP]	89.91	90.46
(3)	Cueing$_a(e_k)$+Naive	[CLS] $\langle c_1 \rangle$ Mark Twain $\langle /c_1 \rangle$ was born in $\langle c_2 \rangle$ Florida $\langle /c_2 \rangle$. [MASK]$_1$ [MASK]$_2$ [MASK]$_3$ [SEP]	93.37	90.62
(4)	Cueing$_a(e_k)$+PTR	[CLS] $\langle c_1 \rangle$ Mark Twain $\langle /c_1 \rangle$ was born in $\langle c_2 \rangle$ Florida $\langle /c_2 \rangle$. the [MASK]$_1$ Mark Twain [MASK]$_2$ [MASK]$_3$ Florida [SEP]	93.63	91.12
(5)	Cueing$_e(e_k)$+Naive	[CLS] { (entity) Mark Twain } was born in { (entity) Florida }. [MASK]$_1$ [MASK]$_2$ [MASK]$_3$ [SEP]	88.75	87.09
(6)	Cueing$_e(e_k)$+PTR	[CLS] { (entity) Mark Twain } was born in { (entity) Florida }. the [MASK]$_1$ Mark Twain [MASK]$_2$ [MASK]$_3$ Florida [SEP]	93.55	90.89
(7)	Cueing$_{ht}(e_k)$+Naive	[CLS] { (head) Mark Twain } was born in $\langle \lceil$ tail $\rfloor$ Florida $\rangle$. [MASK]$_1$ [MASK]$_2$ [MASK]$_3$ [SEP]	95.23	90.43
(8)	Cueing$_{ht}(e_k)$+PTR	[CLS] { (head) Mark Twain } was born in $\langle \lceil$ tail $\rfloor$ Florida $\rangle$. the [MASK]$_1$ Mark Twain [MASK]$_2$ [MASK]$_3$ Florida [SEP]	**96.77**	**92.36**

Table 3: Performance with Different Cueing Strategies

ID	Cueing	SemEval				ReTACRED			
		8	16	32	mean	8	16	32	mean
(1)	Cueing$_o(e_k)$+PTR	69.20	79.08	83.21	77.16	52.34	**58.05**	61.83	**57.41**
(2)	Cueing$_a(e_k)$+PTR	54.79	78.58	85.38	72.92	**52.85**	55.20	63.58	57.21
(3)	Cueing$_e(e_k)$+PTR	61.19	80.46	85.94	75.86	50.13	57.13	63.52	56.93
(4)	Cueing$_{ht}(e_k)$+PTR	**62.26**	**89.80**	**92.07**	**81.38**	50.36	57.98	**63.70**	57.35

Table 4: Performance of Few-shot

entity cues. It is ineffective to let PLMs to distinguish different entities. Comparing with (3), it worsens the performance.

(6) This cueing strategy also suffers from the same problem as its naive version in Row (5), where the same entity cues are used in e_1 and e_2. However, an interesting phenomenon is that the PTR prompt is helpful to compensate the problem. It considerably outperforms its naive version.

(7) In this cueing strategy, different entity cues are used to make a distinction between entities. In stead of specific tags (e.g., $\langle c_k \rangle$ or $\langle /c_k \rangle$), contextual words are also used as entity cues ("head" and "tail"). The result in SemEval shows that it has substantial influence on the performance. However, its performance is not robust. Comparing with the same naive version in Row (3), it achieves lower performance in the ReTACRED corpus.

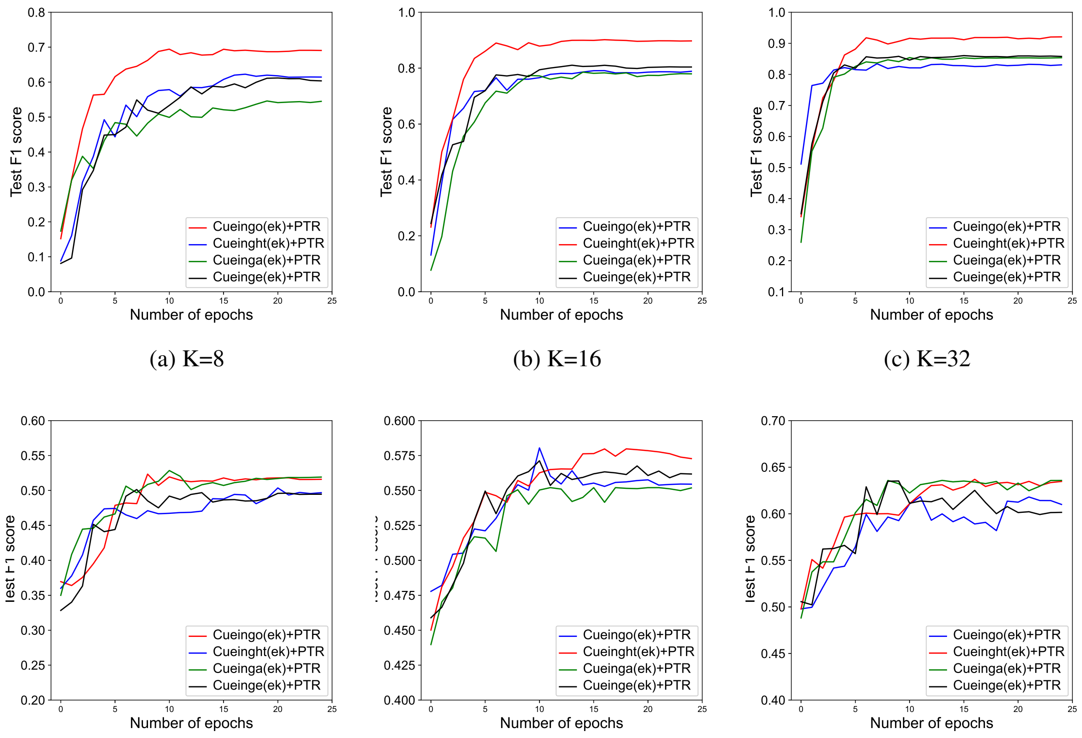
(8) In this cueing strategy, both entity cues and prompts contains contextual words. They are effective to encode contextual features and semantic dependencies of a relation instance, especially in ReTACRED. This setting achieves the highest performance in our experiments.

In all experiments, we also found that entity cues are more useful in the SemEval corpus. The reason is that, comparing with the relation definition in ReTACRED, entity types in the SemEval corpus are more discriminative for relation extraction. When both entity cues and prompts are used simultaneously, the relation extraction achieves more robust and superior performance.

4.4 Performance on Few-shot

Because PLMs encode rich knowledge of a sentence, which is valuable to support few-shot learning. In this experiment, we evaluate the ability of our method in few-shot learning.

We randomly sample K-shot (K $\in$ {8, 16, 32}) from each relation class the training dataset. They are used to tune PLMs, then evaluated on the whole testing dataset. In this experiment, we combine our cueing strategies with the PTR prompt. The result is shown in Table 4.



(a) K=8 (b) K=16 (c) K=32

(d) K=8 (e) K=16 (f) K=32

Figure 2: Influence of Training Epochs on Few-shot Learning

From the Table 4, we can see that few-shot learning still has competitive performance in the SemEval corpus. However, the performance is degenerated considerably in the ReTACRED corpus, because the latter has large number of relation types which are heavily depends semantic features of a relation instance.

Comparing strategy (2) with strategy (1), we found that, in few-shot learning, when the number of shot is small ($K \in \{8, 16\}$), implanting entity cues into the input may worsens the performance. The reason is that PLMs are initialized without implanted entity cues. Therefore, a few shots is incapable to tune PLMs for downstream tasks. In contrast, due to the incompatibility problem, it worsens the performance. When the value of K is increasing, benefiting from entity cues and prompts, the performance is improved rapidly.

In all experiments, when the number of shots is increased, the performance is improved steadily. In SemEval, few-shot learning has impressive performance. When $K = 32$, it has achieved competitive performance. However, in ReTACRED, comparing with Row (8) in Table 4, its performance degraded considerably. The reason is that the relation extraction task in ReTACRED is more challenging. Its performance heavily depends on the number of training data.

In Figure 2, the influence of training epochs on few-shot learning is demonstrated. In SemEval, the cueing strategy can lead to faster convergence. It has stable performance in the training process. However, in ReTACRED, the performance is unstable. The conclusion is the same as in Table 4.

5 Conclusion

Prompt tuning has shown great potential to support natural language processing. However, information extraction usually focuses on identifying specific semantic elements in a sentence. Therefore, in the tuning process, let PLMs attention to targeted elements is important to learn contextual features and semantic dependencies of a sentence. In this paper, we propose several cueing strategies to control the attention of a deep network. In our experiments, they achieve the state of the art performance in relation extraction. The result reveals the mechanism of prompt tuning in relation extraction. In our future work, the cueing strategy can be extended to support other NLP tasks. Furthermore, more studies can be conducted to reveal the mechanism of cueing strategies.

References

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

Yanping Chen, Kai Wang, Weizhe Yang, Yongbin Qing, Ruizhang Huang, and Ping Chen. 2020. A multi-channel deep neural network for relation extraction. *IEEE Access*, 8:13195–13203.

Yanping Chen, Weizhe Yang, Kai Wang, Yongbin Qin, Ruizhang Huang, and Qinghua Zheng. 2021. A neuralized feature engineering method for entity relation extraction. *NN*, 141:249–260.

Yanping Chen, Qinghua Zheng, and Ping Chen. 2015. Feature assembly method for extracting relations in chinese. *AI*, 228:179–194.

Amir DN Cohen, Shachar Rosenman, and Yoav Goldberg. 2020. Relation classification as two-way span-prediction. *arXiv preprint arXiv:2010.04829*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2021. Ptr: Prompt tuning with rules for text classification. *arXiv preprint arXiv:2105.11259*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid O Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2019. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. *arXiv preprint arXiv:1911.10422*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Cheng Li and Ye Tian. 2020. Downstream model design of pre-trained language model for relation extraction task. *arXiv preprint arXiv:2004.03786*.

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

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. *arXiv preprint arXiv:1907.11692*.

Shengfei Lyu and Huanhuan Chen. 2021. Relation classification with entity type restriction. *arXiv preprint arXiv:2105.08393*.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of ACL*, pages 39–48.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Yongbin Qin, Weizhe Yang, Kai Wang, Ruizhang Huang, Feng Tian, Shaolin Ao, and Yanping Chen. 2021. Entity relation extraction based on entity indicators. *Symmetry*, 13(4):539.

Timo Schick and Hinrich Schütze. 2020. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. arXiv:1906.03158.

George Stoica, Emmanouil Antonios Platanios, and Barnabás Póczos. 2021. Re-tacred: Addressing shortcomings of the tacred dataset. *Proceedings of AAAI*, 35(15):13843–13850.

Amirsina Torfi, Rouzbeh A Shirvani, Yaser Keneshloo, Nader Tavaf, and Edward A Fox. 2020. Natural language processing advancements by deep learning: A survey. *arXiv preprint arXiv:2003.01200*.

Linlin Wang, Zhu Cao, Gerard De Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Proceedings of ACL*, pages 1298–1307.

Shanchan Wu and Yifan He. 2019. Enriching pre-trained language model with entity information for relation classification. In *Proceedings of CIKM*, pages 2361–2364.

Chen Xiang, Zhang Ningyu, Xie Xin, Deng Shumin, Yao Yunzhi, Tan Chuanqi, Huang Fei, Si Luo, and Chen Huajun. 2020. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. *arXiv preprint arXiv:2104.07650*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of EMNLP*, pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, pages 2335–2344.

Kang Zhao, Hua Xu, Yue Cheng, Xiaoteng Li, and Kai Gao. 2021. Representation iterative fusion based on heterogeneous graph neural network for joint entity and relation extraction. *KBS*, 219:106888.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proceedings of ACL*, pages 419–426.

Zexuan Zhong and Danqi Chen. 2020. A frustratingly easy approach for joint entity and relation extraction. arXiv:2010.12812.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of ACL*, pages 207–212.

Wenxuan Zhou and Muhao Chen. 2021. An improved baseline for sentence-level relation extraction. *arXiv preprint arXiv:2102.01373*.