KiPT: Knowledge-injected Prompt Tuning for Event Detection

Haochen Li[1] , **Tong Mo**[1], **Hongcheng Fan**[1], **Jingkun Wang**[1],
Jiaxi Wang[1], **Fuhao Zhang**[2] **and Weiping Li**[1,*]
[1]Peking University, Beijing, China
[2]Chinese Academy of Surveying and mapping, Beijing, China
{haochenli, wjk, wangjxi, fanhongcheng}@pku.edu.cn
{wpli, motong}@ss.pku.edu.cn, zhangfh@ac.cn

Abstract

Event detection aims to detect events from the text by identifying and classifying event triggers (the most representative words). Most of the existing works rely heavily on complex downstream networks and require sufficient training data. Thus, those models may be structurally redundant and perform poorly when data is scarce. Prompt-based models are easy to build and are promising for few-shot tasks. However, current prompt-based methods may suffer from low precision because they have not introduced event-related semantic knowledge (e.g., part of speech, semantic correlation, etc.). To address these problems, this paper proposes a Knowledge-injected Prompt Tuning (KiPT) model. Specifically, the event detection task is formulated into a condition generation task. Then, knowledge-injected prompts are constructed using external knowledge bases, and a prompt tuning strategy is leveraged to optimize the prompts. Extensive experiments indicate that KiPT outperforms strong baselines, especially in few-shot scenarios.

1 Introduction

Events describe state changes of participating entities. The Event Detection (ED) task is one of the essential tasks in the Information Extraction field. Event triggers are the most representative words or phrases in events, and they are usually composed of verbs or nouns. There is a one-to-one correspondence between events and event triggers, so the ED task is equivalent to identifying and classifying event triggers.

The ED task has a wide range of applications, providing helpful information for downstream tasks such as text summarization, auto summarization, machine question and answer (QA), etc. Meanwhile, with the vigorous development of Internet news and social media, ED has become a practical approach for extracting information from massive texts. Therefore, the ED task has attracted increasing attention with great academic and applied value in recent years.

Most current ED models use a pre-trained language model to build complex downstream networks (including CNN, RNN, GCN, etc.) These methods perform well on public datasets, but they rely heavily on the fine-tuning strategy to train their downstream networks and introduce massive extra parameters. However, due to the scarcity and uneven distribution of the annotated data for ED, these methods may cause severe overfitting problems. Further, these methods may perform poorly in data-scarce scenarios because their extra parameters cannot be fully optimized.

Recently, prompt-based learning methods is a new trend in natural language processing. Researchers verified that pre-trained language models already have enough knowledge, so the complex downstream networks are unnecessary in many cases. Prompt-based learning methods make full use of the information in pre-trained language models by constructing prompts to guide the language model to solve NLP tasks. Specifically, prompt-based methods first transform the original input into prompt templates containing the initial input, the prompt tokens, and unfilled slots for output. Then, a pre-trained language model is employed to fill the unfilled slots to obtain a final string from which the final output can be derived.

Prompt learning-based methods eliminate complex downstream networks and massive extra parameters, so they have advantages in data scarcity scenarios. Meanwhile, the prompts' quality directly affects the models' performance. However, manually selecting the optimal prompts is time-consuming and labor-intensive. Considering this, prompt tuning strategies have been proposed by introducing continuous virtual tokens as trainable prompts that will be optimized through training.

[*]Weiping Li is the corresponding author.

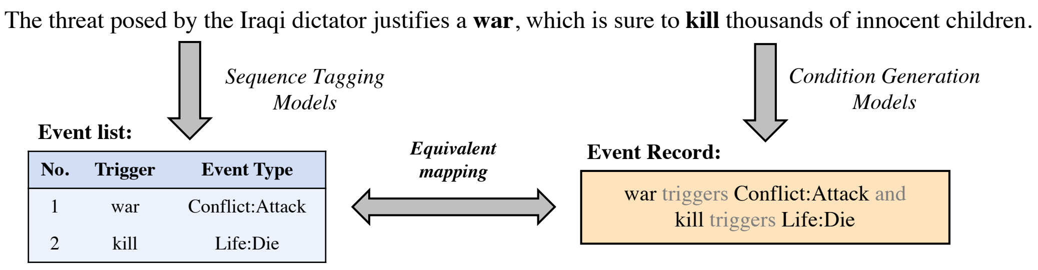

The threat posed by the Iraqi dictator justifies a war, which is sure to kill thousands of innocent children.
Sequence Tagging Models
Condition Generation Models
Event list:
No. Trigger Event Type
1 war Conflict:Attack
2 kill Life:Die
Equivalent mapping
Event Record:
war triggers Conflict:Attack and kill triggers Life:Die

Figure 1: An example of Event Detection (on the left is the event list output by the sequence tagging model; on the right is the event record output by the condition generation model, where the gray words act as structural tokens)

However, the above prompts may suffer from low precision because they have not introduced event-related semantic knowledge.

Event triggers are mostly verbs and nouns, and are usually semantically related to the core concept words of events. Therefore, semantic knowledge (such as part of speech, word semantic correlation, etc.) plays a crucial role in the ED task. To this end, this paper proposes a Knowledge-injected Prompt Tuning (KiPT) model to introduce event-related knowledge using external knowledge bases.

Specifically, we formulate the ED task into a condition generation task. Then, external knowledge bases and semantic tools are used to obtain the semantic knowledge associated with input sentences and events. Next, the semantic knowledge is injected into the prompts for ED. Finally, we use the knowledge-injected prompts to extract event triggers, and the prompts will be optimized through a prompt tuning strategy. Our method is direct and effective, and it can be easily transferred to other tasks.

The main contributions of this paper are summarized as follows:

- We introduce a knowledge injection method to inject event-related semantic knowledge into the prompt templates, which is the first in the ED task to the best of our knowledge;
- We propose a prompt-based learning model for ED called Knowledge-injected Prompt Tuning (KiPT), which leverages a prompt tuning strategy to optimize the prompts;
- Extensive experiments show that our model outperforms current prompt-based ED models and strong baselines, especially in data scarcity scenarios.

2 Related Work

Studies related to our work are mainly discussed from the following three aspects:

2.1 Event Detection Methods

The ED models can be divided into sequence tagging models and condition generation models.

Chen et al. and Nguyen et al. formulated ED as a sequence tagging task for the first time, and they used CNN and RNN to model sentence-level features; Liu et al. and Yan et al. used GCN to emphasize the semantic dependency. Nguyen and Nguyen jointly extracted entities, triggers, and arguments based on the shared hidden representations; Wadden et al. provided a graph propagation method to capture context relevant for entity, relation, and event; Lin et al. built an end-to-end information extraction system which employs global feature and beam search to extract globally optimal event structures;

Condition generation methods encode sentences using generative pre-trained language models such as BART(Lewis et al., 2019) and T5(Raffel et al., 2019). Li et al. utilized a conditional generation model with BART. Paolini et al. regarded event extraction as a translation task between augmented natural languages; Lu et al. constructed events as event trees and used a Seq2Structure model.

2.2 Prompt-based Learning Methods

Prompt-based learning methods use prompts to guide pre-trained language models to generate results, so the quality of the prompt templates is critical. The current prompt-base learning templates include:

Manually setting discrete prompt templates (actual words in the template): Schick and Schütze transferred the text classification task into a cloze-filling task by using manual prompt templates.

Petroni et al. converted triple completion tasks into cloze-filing questions through prompt templates.

Building trainable continuous prompt templates (virtual tokens in the template): Li and Liang used trainable prefix tokens as prompts and added soft tokens in each layer of the language model. Liu et al. replaced actual words with trainable soft tokens in the prompt template and introduced an extra prompt encoder.

However, neither of these methods introduces task-related knowledge, and they cannot optimize the prompt along with external knowledge.

2.3 Prompt-based Knowledge Injection

Some works have already attempted to introduce external knowledge in prompt-based learning. Hu et al. enhanced the mapping of model outputs and predefined categories by introducing extra knowledge in verbalizers. Chen et al. introduced virtual tokens to enhance category features for relation extraction tasks. Li et al. strengthened the model by introducing ground truths. Although these methods are instructive, they have strong task dependencies and are difficult to apply to Event Detection.

Our work aims to perform the ED task using a prompt-based model with event-related knowledge. To achieve this, a knowledge-injected prompt tuning method is proposed in this paper.

3 Methodology

This section first describes the definition of the ED task. Then, the proposed KiPT model is introduced in detail.

3.1 Task Description

Following the task description of Automatic Content Extraction [1], the standard task of ED includes event Trigger Identification (Trig-I) and Trigger Classification (Trig-C). Consider the example in Figure 1. After obtaining the input sentence, ED methods should first identify the triggers **"war"** and **"kill"** and then classify them into event types **"Conflict:Attack"** and **"Life:Die"**.

This paper formulates the standard ED task into a condition generation task to simplify the output. First, the structural word "triggers" is used to combine a trigger and its corresponding type like "**war** triggers **Conflict:Attack**". Then, if a sentence contains more than one event, the events are concatenated with another structural word "and". Considering the example in Figure 1, our constructed event record is **"war** triggers **Conflict:Attack** and **kill** triggers **Life:Die**". Therefore, given the input sentence, our model generates the above event record as output. It is worth mentioning that the output of our method is equivalent to the original ED task since event records can be easily split into event triggers and event types.

3.2 The Overall Structure of KiPT

The overall structure of KiPT is shown in Figure 2. First, given the input sentence x, the knowledge-injected prompt $Prompt(x)$ is constructed. Then, prompt template input T is built and fed into the pre-trained language model $\mathcal{LM}$ to obtain the output event record y.

We propose a trainable knowledge-injected prompt $Prompt(x)$ in KiPT. $Prompt(x)$ includes two parts: input-related **knowledge injection** $K(x)$ and input-irrelated **soft tokens** S. $K(x)$ and S are both trainable and will be optimized during the training process.

In the following subsections, the construction and tuning of the knowledge-injected prompt will be explained in detail.

3.3 Knowledge-injected Prompt Construction

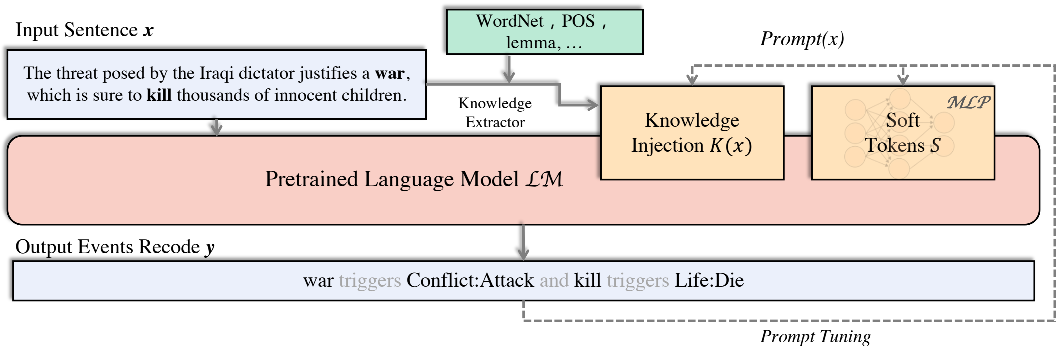
In this section, the definitions of the knowledge injection $K(x)$ and the soft tokens S are introduced first. Then, the combination of $K(x)$ and S into the prompt $Prompt(x)$ is described.

Knowledge Injection $K(x)$**:** Given each input sentence x, the token sequence $x_{1:n} = \{x_1, x_2, ..., x_n\}$ can be obtained by using a tokenizer. Then, a knowledge extractor is used to extract event-related knowledge $K(x)$ for x, and the knowledge extractor is constructed by using NLP analysis tools and external knowledge bases.

First, since event triggers are mostly verbs and nouns, each token's part of speech (POS) is essential. Thus, POS analysis is performed on each token x_i using a semantic analysis tool (**Stanford's Stanza**(Qi et al., 2020)). Then, Stanza's lemmatization module is employed to recover the lemma form $\hat{x_i}$ of each input token (for example, "died" -> "die"). After the POS and the lemma form of each token are obtained, the verbs and nouns are selected as a list of potential triggers $PT_{1:|PT|} = \{pt_1, pt_2, ..., pt_{|PT|}\}$.

Semantic correlations in knowledge bases also provide vital information for ED. Each type of

[1] https://www.ldc.upenn.edu/collaborations/past-projects/ace/annotation-tasks-and-specifications



Input Sentence *x*
WordNet , POS , lemma, ...
Prompt(x)
The threat posed by the Iraqi dictator justifies a **war**, which is sure to **kill** thousands of innocent children.
Knowledge Extractor
Knowledge Injection $K(x)$
Soft Tokens S
MLP
Pretrained Language Model $\mathcal{LM}$
Output Events Recode *y*
war triggers Conflict:Attack and kill triggers Life:Die
Prompt Tuning

Figure 2: The overall architecture of KiPT. Given the input sentence x, external knowledge are used to construct input-related prompt $Prompt(x)$. Then, the language model $\mathcal{LM}$ is used to generate output event record y.

event has some core concepts, e.g., the event type "Conflict: Attack" has concepts like "attack, fight, bomb, etc.", and the event type "Life: Die" has concepts like "kill, suicide, murder, etc." To exploit the semantic relevance of the core concepts of words and events, we make the following assumption:

Assumption 3.1 *If a word has strong semantic correlations with the core concepts of a specific event type, then the word is likely to trigger an event of that type.*

Based on the above assumption, a dictionary $Concept_{1:e}$ containing the core concepts of each event type (e represents the total number of event types) is manually constructed, and $concept_k$ represents the concept of the k^{th} type of event. Then, **WordNet**(Fellbaum and Miller, 1998) is introduced to obtain the semantic correlation between the input tokens and the core concepts, and whether they have a strong semantic correlation is judged by calculating the semantic similarity of two words in WordNet. For each word pt_j in the list of potential triggers PT and the core concept $concept_k$ for each event type, their semantic correlation $Sc(pt_j, concept_k)$ is calculated by using the *Wu-Palmer Similarity Algorithm*(Wei et al., 2015). If the semantic correlation $Sc(pt_j, concept_k)$ is above the similarity threshold θ_{sim}, the potential trigger word pt_j and the corresponding event type $event_k$ are both added to $K(x)$.

To sum up, the process of the knowledge extractor is as follows:

The final extracted knowledge injection $K(x)$ is a text sequence composed of potential triggers and their potential event types. It is denoted as $K(x)_{1:|K|} = \{k_1, k_2, ..., k_{|K|}\}$, where $|K|$ stands for the length of K. Further, the similarity threshold θ_{sim} may affect the performance of our model, which will be discussed in the Appendix.

Algorithm 1 Knowledge Extractor

```
Input: Sentence: $x_{1:n}$, Concept dict: $Concept_{1:e}$
Output: Knowledge Injection: $K(x)$
  for $x_i \in x$ do
    if $POS(x_i) \in [verb, noun]$ then
      Add $lemma(x_i)$ to potential triggers $PT$
    end if
  end for
  for $pt_j \in PT$ do
    for $concept_k \in Concept$ do
      if $Sc(pt_j, concept_k) > \theta_{sim}$ then
        Add $[pt_j, event_k]$ to $K(x)$
      end if
    end for
  end for
```

Soft Tokens S: Besides using knowledge injection $K(x)$, this paper also adds some trainable soft tokens to the prompts. Soft tokens are virtual tokens sharing the same dimension as actual words (e.g, 768 for T5-base) but without real meanings. Previous works have proved that trainable soft prompts are more flexible and effective than actual words (Liu et al., 2021; Li and Liang, 2021).

In our method, randomly initialized tensors are used as the soft prompt tokens, which will be optimized during the training of the language model. The soft tokens used in this paper are denoted as $S_{1:p} = \{s_1, s_2, ..., s_p\}$. The selection of p may slightly affect the performance of our model, and this will be discussed in Appendix.

Prompt Templates Construction: After knowledge injection $K(x)$ and soft tokens S are obtained, they are concatenated as $Prompt(x)$:

$$\begin{aligned} Prompt(x) &= [K(x); S] \\ &= \{k_1, ..., k_{|K|}, s_1, ..., s_p\} \end{aligned} \quad (1)$$

For each input x, $Prompt(x)$ is constructed as the knowledge-injected prompt. Then, a prompt template is built, which contains the input x, $Prompt(x)$, and the target event record y.

$$\textbf{Template} : Prompt(x)\ [x],\ Events:\ [y] \quad (2)$$

Where $[x]$ represents the slot for the input sentence, $[y]$ represents the slot for the target event records, and "$Events$: " is a fixed anchor token. Anchor tokens have been proved useful by previous works (Li and Liang, 2021; Han et al., 2021).

3.4 Knowledge-injected Prompt Tuning

The knowledge-injected prompt $Prompt(x)$ needs to be optimized through training mainly for the following two reasons:

(1) Some rule-based algorithms are used during the construction of the knowledge injection $K(x)$. However, these rules may be ineffective or even wrong in some cases, so these rules need to be softened through training;

(2) Soft tokens S are virtual tensors that are randomly initialized and have no original semantics. They need to be trained to approximate the distribution of actual words to achieve the role of prompts for language models.

To this end, we propose knowledge-injected prompt tuning to optimize $Prompt(x)$. Given a pre-trained language model $\mathcal{LM}$ and its vocabulary $\mathcal{V}$, the prompt template's input T is:

$$\begin{aligned} T &= [H^k; H^s; e(x)] \\ &= \{h_1^k, ..., h_{|K|}^k, h_1^s, ..., h_p^s, e(x_1), ..., e(x_n)\} \end{aligned} \quad (3)$$

where $e(x_i)$ indicates the embeddings for the input tokens; h_i^s and h_i^k stand for embedded prompts for the knowledge injection and the soft tokens, respectively. Note that $e(x_i)$ and h_i^k are initialized using the embeddings of the actual tokens from the $\mathcal{LM}$'s vocabulary $\mathcal{V}$, and h_i^s indicates randomly initialized tensors. The conditional probability of the event record output can be obtained by using the generative language model $\mathcal{LM}$.

Finally, given the golden event record y, gradient updates are performed by using the following log-likelihood loss function:

$$L = -\sum_{(x,y)\in\mathcal{D}} \log(y|H^k, H^s, e(x), \theta_{\mathcal{LM}}) \quad (4)$$

where $\mathcal{D}$ stands for the whole training dataset, and $\theta_{\mathcal{LM}}$ stands for the $\mathcal{LM}$'s parameters.

Previous research (Liu et al., 2021) has shown that there are semantic gaps between the embeddings of actual tokens and virtual tokens. Because the embeddings of actual words are highly discrete through pre-training, trainable soft prompt tokens are randomly distributed. They will only change in a small neighborhood during stochastic gradient descent.

Thus, following Liu et al.'s work, we use a lite muti-layer linear network ($\mathcal{MLP}$) as a prompt encoder to narrow the semantic distance between the embedding of actual words and prompt tokens:

$$\hat{h_i^s} = \mathcal{MLP}(h_i^s), i \in [1, p] \quad (5)$$

The loss function is improved as follows:

$$L = -\sum_{(x,y)\in\mathcal{D}} \log(y|H^k, H^s, e(x), \theta_{\mathcal{LM}}, \theta_{\mathcal{MLP}}) \quad (6)$$

where $\theta_{\mathcal{MLP}}$ stands for the parameters of the muti-layer linear network $\mathcal{MLP}$.

4 Experiments

This section describes our experimental settings and detailed experimental analysis.

First, we conduct overall experiments to verify the performance of KiPE with sufficient training data. Next, we conduct few-shot and zero-shot experiments to verify the performance of KiPT in data-scarce scenarios. Finally we perform an ablation analysis to explore the influence of each part in our prompts.

4.1 Experiment Setup

Datasets. Our work is evaluated on the most widely used datasets ACE 2005 (Automatic Content Extraction program of 2005)[2] and TAC 2015 (Event Nugget data of TAC 2015)[3]. The detailed descriptions of the datasets are presented in Table 1:

As for ACE 2005, following the previous works (Yan et al., 2019; Nguyen and Nguyen, 2019), the 599 documents are divided into 529 training documents, 30 development documents, and 40 test documents. And for TAC 2015,following Lu et al., the 458 documents are divided into 396 training documents, 31 development documents, and 31 test documents.

[2]https://catalog.ldc.upenn.edu/LDC2006T06

[3]https://catalog.ldc.upenn.edu/LDC2020T13

Datasets	ACE 2005	TAC 2015
Event Type	33	38
Total Documents	599	458
Total Events	5,055	7,530
Train/Dev/Test Split	529/30/40	396/31/31

Table 1: Statistics of ACE 2005 and TAC 2015

Metrics. The ED task has two subtasks: **Trigger Identification (Trig-I)** and **Trigger Classification (Trig-C)**. Their standard evaluation criteria are as follows:

- **Trig-I**: An event trigger is identified correctly if its span matches the gold trigger;
- **Trig-C**: An event trigger is classified correctly if both its span and event type match the gold trigger.

We use micro-averaged Precision (**P**), Recall (**R**), and F1 score (**F1**) in all the following evaluations.

Settings. In this paper, T5 (Yang et al., 2019) is utilized as the pre-trained language model $\mathcal{LM}$ in KiPT. Both T5-Base and T5-Large are used in the overall results, while only T5-Base is used in the rest of the experiments. Our model is optimized with AdamW for 30 epochs with a learning rate of 1e-4 and a weight decay of 1e-5 for T5, and a learning rate of 1e-3 and a weight decay of 1e-4 for other parameters. The batch size is set to 16 for T5-base and 8 for T5-large.

In our main KiPT model, the number of soft tokens p is set to 40, and the similarity threshold θ_{sim} is set to 0.8. The performance of the other options will be discussed in Appendix.

Baselines. This paper chooses 10 strong baselines for comparison: (1) Three sequence tagging models: **Joint3EE** jointly extracts entities, triggers, and arguments based on the shared hidden representations; **DYGIE++** provides a graph propagation method to capture relevant context for entity, relation, and event; **OneIE** builds an end-to-end information extraction system that employs global feature and beam search to extract globally optimal event structures; (2) Two QA-based models: **BERT_QA** first formulates ED as a QA task and generates questions from annotation guidelines; **MQAEE** uses multi-turn question strategy to build questions; (3) Three condition generation models: **BART-Gen** utilizes a conditional generation model with BART. Given the description of events, the corresponding triggers in the sentence are generated; **TANL** frames ED as a translation task between augmented natural languages; **Text2Event** constructs event structure and uses a Seq2Structure model. (4) Two prompt-based learning models: **PoKE** presents various joint prompt methods, which can elicit more complementary knowledge by modeling the interactions between different triggers or arguments; **GDAP** empowers the automatic exploitation of label semantics on prompt templates.

4.2 Overall Results

The overall results of our model on ACE 2005 and TAC 2015 are presented in Table 2 and Table 3, respectively. For both Trig-I and Trig-C tasks, KiPT (T5-large) outperforms all strong baselines on ACE 2005, reaching F1 values of 78.6% and 75.3%, respectively. Due to the reduced parameters of the pre-trained language model, the performance of KiPT (T5-base) drops slightly (-1.1% in Trig-I and -0.4% in Trig-C), but it still exceeds most of the baselines. On TAC 2015, OneIE and Text2Event are compared because they are the only two baselines experimented on TAC 2015. Although our model is slightly lower than OneIE in Trig-I, it outperforms all baselines in Trig-C.

From the overall results, it can be seen that:

(1) Our model significantly outperforms other prompt-based models (PoKE and GDAP) because of the utilization of event-related knowledge. The results indicate that prompt-based learning models tend to achieve higher Recall but lower Precision in both Trig-I and Trig-C tasks. We believe that prompt-based learning models only use the knowledge of pre-trained language models, which is more general but lacks task specificity.

KiPT improves this problem significantly by introducing event-related knowledge through knowledge injection, narrowing the gap between Precision and Recall. In the Trig-C task, the gap between Recall and Precision of KiPT (T5-base) is 4.0%, while that of PoKE and GDAP is 12.1% and 9.2%, respectively. That is why KiPT outperforms other prompt-based learning models.

Further, KiPT(T5-base) has better precision than KiPT(T5-large) but has lower recall and F1. The possible reason is that T5-large contains more general knowledge and reduces the proportion of event-related knowledge we introduced.

(2) Without introducing complex downstream networks, KiPT (T5-base) outperforms the best T5-base model PoKE by 5.3% in F1, and KiPT

Models	Trig-I			Trig-C			PLM
	P	R	F1	P	R	F1	
Sequence Tagging Models							
Joint3EE (Nguyen and Nguyen, 2019)	70.5	74.5	72.5	68.0	71.8	69.8	-
DYGIE++ (Wadden et al., 2019)	-	-	76.5	-	-	73.6	BERT-large
OneIE (Lin et al., 2020)	-	-	**78.6**	-	-	75.2	BERT-large
QA Models							
BERT_QA (Du and Cardie, 2020)	74.3	77.4	75.8	71.1	73.7	72.4	2×BERT-base
MQAEE (Li et al., 2020)	-	-	77.4	-	-	73.8	3×BERT-large
Condition Generation Models							
BART-Gen (Li et al., 2021b)	-	-	74.4	-	-	71.1	BART-large
TANL (Paolini et al., 2021)	-	-	-	-	-	68.5	T5-base
Text2Event (Lu et al., 2021)	-	-	-	69.6	74.4	71.9	T5-large
Prompt-based Learning Models							
PoKE (Lin et al., 2021)	-	-	-	64.1	76.2	69.6	T5-base
GDAP (Si et al., 2022)	-	-	-	65.6	74.7	69.9	T5-large
Our Model							
KiPT (T5-base)	**76.0**	79.1	77.5	**72.9**	76.9	74.9	T5-base
KiPT (T5-large)	75.4	**82.1**	**78.6**	71.6	**79.2**	**75.3**	T5-large

Table 2: Experimental results on ACE 2005. Trig-I indicates trigger identification tasks and Trig-C indicates trigger classification tasks. The column PLM indicates the pre-trained language models used in each model.

Model	Trig-I	Trig-C
OneIE	**68.4**	57.0
Text2Event	-	57.8
KiPT (T5-base)	66.3	58.1
KiPT (T5-large)	67.0	**58.3**

Table 3: Experimental results on TAC 2015. The F1 score is recorded for each model.

$k - shot$	OneIE	Text2Event	PoKE	KiPT
$k = 4$	22.8	39.6	44.2	**45.5**
$k = 8$	25.2	51.2	**53.1**	50.1
$k = 16$	28.6	52.1	53.8	**54.2**
$k = 32$	39.7	53.7	55.3	**56.4**
$k = 64$	48.6	57.8	59.1	**63.6**
All Data	**75.2**	71.9	69.6	74.9

Table 4: Experiment with few-shot settings. The average F1 scores of 10 experiments are used for each model.

(T5-large) outperforms the best T5-large model Text2Event by 3.4%. This proves the effectiveness of our prompt tuning strategy.

(3) Our model has a relatively small F1 drop from Trig-I to Trig-C: -2.6% for KiPT (T5-base) and -2.3% for KiPT (T5-large). As a comparison, the average drop of the QA model is -3.5% and that of the condition generation model is -3.3%. This indicates that KiPT binds potential triggers and their corresponding event types together through knowledge injection, so it is easier for it to classify trigger words correctly after identifying them.

Among all the baselines, OneIE has a close performance to KiPT, indicating that sequence tagging models still have competitive performance with enough training data. However, the introduction of complex downstream networks and massive parameters may face struggles when data is insufficient.

Next, we will perform data-scarce experiments.

4.3 Few-shot and Zero-shot Scenarios

To verify the advantages of KiPT in low-resource settings, we conduct few-shot and zero-shot experiments on ACE 2005. Three strong baselines are compared: sequence tagging model OneIE, condition generation model Text2Event, and prompt-based learning model PoKE.

Few-shot experiments: Referring to the few-shot settings of previous works (Gao et al., 2020; Lin et al., 2021), 4, 8, 16, 32, and 64 shot experiments are conducted to compare the performance of KiPT and baselines under scenarios with small data resources. Specifically, for each type of event, k samples are randomly selected from the initial training set. Then, after training, the models' per-

Models	Trig-I	Trig-C
OneIE	37.6	34.7
Text2Event	45.1	38.4
Poke	44.7	39.3
KiPT	**44.9**	**42.3**

Table 5: Experiment with zero-shot settings. The F1 scores for Trig-I and Tri-C are on 23 unseen event types.

Models	F1 score for Trig-C
KiPT (T5-base)	**74.9**
- $\mathcal{MLP}$	73.0 (-1.9)
- S	73.6 (-1.3)
- $K(x)$	72.2 (-2.7)
- $K(x)$ and S	71.0 (-3.9)

Table 6: Ablation study results on ACE 2005. '-' means the removal of the corresponding component.

formance is tested on the standard test set. Each $k-shot$ experiment is repeated 10 times, and the average results are recorded finally.

The results are shown in Table 4. It can be seen that KiPT outperforms all the strong baselines in most few-shot settings. Further analysis indicates that:

(1) Our model is still effective when the data is particularly sparse. For example, in the setting of $k = 4$, KiPT reaches 45.5% in F1 score, significantly surpassing all baselines;

(2) The lead of current prompted-based models will gradually shrink as the training data increase. For example, PoKE outperforms Text2Event by 4.6% and 1.3% when $k = 4$ and $k = 64$, respectively. However, benefiting from the trainable knowledge-injected prompts, KiPT's performance grows uniformly as the data increase. In detail, KiPT surpasses Text2Event by 5.9% and 5.8% when $k = 4$ and $k = 64$, respectively.

(3) OneIE perform poorly in all few-shot scenarios. This indicates that the models using complex downstream networks and extra parameters require sufficient training data to achieve their protential.

Zero-shot experiments: Following the settings of Lu et al., we selected the top 10 most popular event types as seen types and tested the zero-shot classification performance for the remaining 23 unseen types. The results are shown in Table 5, where KiPT overpasses all baselines, especially in the Trig-C task. This indicates that even for a new unseen event type, we can also obtain its event-related knowledge through our prompts, proving the transfer potential of our model.

4.4 Ablation Study

An ablation study is conducted to verify the effectiveness of each component of KiPT. Specifically, four groups of controlled experiments are designed: "$-\mathcal{MLP}$" means removing the prompt encoder proposed in equation 5; "$-S$" means only using knowledge injection in prompt construction; "$-K(s)$" means only using soft tokens as prompts; "$-K(x)$ and S" means removing all the prompts. The results of the ablation study are presented in Table 6. It can be seen that:

(1) When $\mathcal{MLP}$ is removed, the model's performance drops slightly (-1.9%). This indicates that although soft tokens can be optimized by training, the gap between virtual tensors and actual words may still cause an adverse effect, showing the necessity of our prompt encoder.

(2) Knowledge injection plays a more significant role than soft tokens within prompts. The removal of $K(x)$ caused a performance drop of 2.7%, over twice the removal of S (1.3%). This indicates that introducing additional semantic knowledge for prompt-based models is better than adding randomly initialized tokens.

(3) After all the prompts are removed, our model downgrades into a simple condition generation model, which causes a significant performance drop (-3.9%). Because we does not design downstream networks, the performance of the downgraded model is lower than that of condition generation baselines (Text2Event and BART-Gen). This validates the effectiveness of our knowledge-injected prompt tuning strategy.

5 Conclusion

This paper proposes a prompt-based learning method for ED by introducing knowledge-injected prompt tuning. External knowledge and soft tokens are used to construct knowledge-injected prompts, which can be optimized through training. Comprehensive experiments demonstrate that KiPT outperforms current prompt-based ED models and strong baselines, especially in data-scarce scenarios. Through our method, prompt-based models can introduce task-related knowledge more conveniently and effectively. In the future, we will explore more knowledge injection approaches and their applications in other tasks.

6 Acknowledgments

This work is supported by the National Key Research and Development Program of China (No. 2020YFC0833300).

References

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2022. KnowPrompt: Knowledge-aware Prompt-tuning with Synergistic Optimization for Relation Extraction. *arXiv:2104.07650 [cs].*

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event Extraction via Dynamic Multi-Pooling Convolutional Neural Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176, Beijing, China. Association for Computational Linguistics.

Xinya Du and Claire Cardie. 2020. Event Extraction by Answering (Almost) Natural Questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 671–683, Online. Association for Computational Linguistics.

C. Fellbaum and G. Miller. 1998. *WordNet : an electronic lexical database*. WordNet:An Electronic Lexical Database.

T. Gao, A. Fisch, and D. Chen. 2020. Making pre-trained language models better few-shot learners.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2021. PTR: Prompt Tuning with Rules for Text Classification. *arXiv:2105.11259 [cs].*

Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Juanzi Li, and Maosong Sun. 2021. Knowledgeable Prompt-tuning: Incorporating Knowledge into Prompt Verbalizer for Text Classification. *arXiv:2108.02035 [cs].*

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461.*

Chengxi Li, Feiyu Gao, Jiajun Bu, Lu Xu, Xiang Chen, Yu Gu, Zirui Shao, Qi Zheng, Ningyu Zhang, Yongpan Wang, and Zhi Yu. 2021a. SentiPrompt: Sentiment Knowledge Enhanced Prompt-Tuning for Aspect-Based Sentiment Analysis. *arXiv:2109.08306 [cs].*

Fayuan Li, Weihua Peng, Yuguang Chen, Quan Wang, Lu Pan, Yajuan Lyu, and Yong Zhu. 2020. Event Extraction as Multi-turn Question Answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 829–838, Online. Association for Computational Linguistics.

Sha Li, Heng Ji, and Jiawei Han. 2021b. Document-Level Event Argument Extraction by Conditional Generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 894–908, Online. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. *arXiv:2101.00190 [cs].*

J. Lin, J. Jian, and Q. Chen. 2021. Eliciting knowledge from language models for event extraction.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A Joint Neural Model for Information Extraction with Global Features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009, Online. Association for Computational Linguistics.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks. *arXiv:2110.07602 [cs].*

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018. Jointly Multiple Events Extraction via Attention-based Graph Information Aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, Brussels, Belgium. Association for Computational Linguistics.

Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. 2021. Text2Event: Controllable Sequence-to-Structure Generation for End-to-end Event Extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2795–2806, Online. Association for Computational Linguistics.

T. H. Nguyen, K. Cho, and R. Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.*

Trung Minh Nguyen and Thien Huu Nguyen. 2019. One for All: Neural Joint Modeling of Entities and Events. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:6851–6858.

G. Paolini, B. Athiwaratkun, J. Krone, J. Ma, A. Achille, R. Anubhai, Cnd Santos, B. Xiang, and S. Soatto. 2021. Structured prediction as translation between augmented natural languages.

F. Petroni, T. Rocktschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel. 2019. Language models as knowledge bases?

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

T. Schick and H Schütze. 2020. Exploiting cloze questions for few shot text classification and natural language inference.

Jinghui Si, Xutan Peng, Chen Li, Haotian Xu, and Jianxin Li. 2022. Generating Disentangled Arguments with Prompts: A Simple Event Extraction Framework that Works. *arXiv:2110.04525 [cs]*.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, Relation, and Event Extraction with Contextualized Span Representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5783–5788, Hong Kong, China. Association for Computational Linguistics.

Tingting Wei, Yonghe Lu, Huiyou Chang, Qiang Zhou, and Xianyu Bao. 2015. A semantic approach for text clustering using wordnet and lexical chains. *Expert Systems with Applications*, 42(4):2264–2275.

H. Yan, X. Jin, X. Meng, J. Guo, and X. Cheng. 2019. Event detection with multi-order graph convolution and aggregated attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. 2019. Exploring Pre-trained Language Models for Event Extraction and Generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5284–5294, Florence, Italy. Association for Computational Linguistics.

A Appendix

In the appendix, we discuss the influence of the soft prompt tokens' number p and the similarity threshold θ_{sim} on KiPT.

A.1 The selection of soft tokens' number p

We set up experiments for different numbers of soft tokens for detailed analysis. We set up experiments with soft tokens' number as 0, 10, 20, 40, 80, and 160 on both ACE 2005 and TAC 2015 datasets. The results are shown in Figure 3.

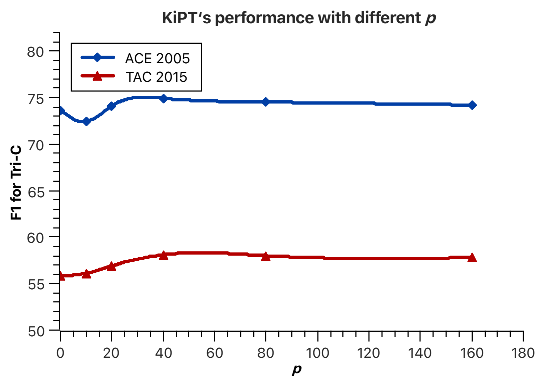


Figure 3: KiPT's performance with different p selection

We can observe that p and the KiPT's performance are not entirely positively correlated, and the model performance peaks when p is around 40. We believe that the small number of soft tokens will make the semantic information captured by the model insufficient, and it is challenging to classify events accurately. On the contrary, introducing too many soft tokens will dilute the original semantic information of the sentence, which will also lead to a decline in the model's performance.

A.2 The selection of similarity threshold θ_{sim}

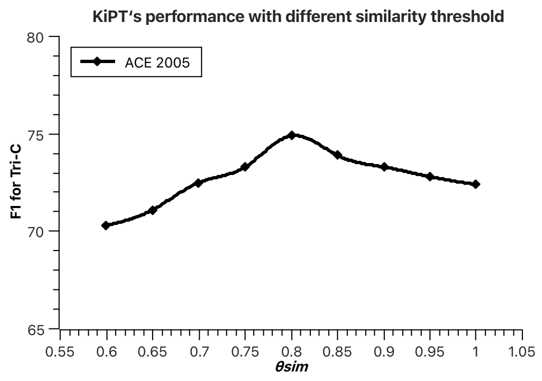


Figure 4: KiPT's performance with different θ_{sim}

The selection of similarity threshold θ_{sim} will directly influence the knowledge injection $K(s)$. Lower θ_{sim} will introduce more knowledge along with more noise, while higher θ_{sim} will inject knowledge more precisely but less broadly. KiPT performs best when θ_{sim} equals 0.8.