PromptNER: Prompt Locating and Typing for Named Entity Recognition

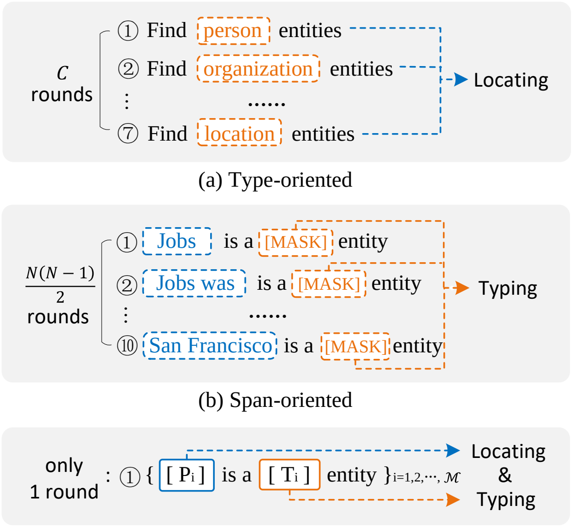
Yongliang Shen[1], **Zeqi Tan**[1], **Shuhui Wu**[1], **Wenqi Zhang**[1],
Rongsheng Zhang[2], **Yadong Xi**[2], **Weiming Lu**[1†], **Yueting Zhuang**[1]
[1]College of Computer Science and Technology, Zhejiang University
[2]Fuxi AI Lab, NetEase Inc.
{syl, luwm}@zju.edu.cn

Abstract

Prompt learning is a new paradigm for utilizing pre-trained language models and has achieved great success in many tasks. To adopt prompt learning in the NER task, two kinds of methods have been explored from a pair of symmetric perspectives, populating the template by enumerating spans to predict their entity types or constructing type-specific prompts to locate entities. However, these methods not only require a multi-round prompting manner with a high time overhead and computational cost, but also require elaborate prompt templates, that are difficult to apply in practical scenarios. In this paper, we unify entity locating and entity typing into prompt learning, and design a dual-slot multi-prompt template with the position slot and type slot to prompt locating and typing respectively. Multiple prompts can be input to the model simultaneously, and then the model extracts all entities by parallel predictions on the slots. To assign labels for the slots during training, we design a dynamic template filling mechanism that uses the extended bipartite graph matching between prompts and the ground-truth entities. We conduct experiments in various settings, including resource-rich flat and nested NER datasets and low-resource in-domain and cross-domain datasets. Experimental results show that the proposed model achieves a significant performance improvement, especially in the cross-domain few-shot setting, which outperforms the state-of-the-art model by +7.7% on average[1].

1 Introduction

Named entity recognition (NER) is a fundamental task in natural language processing that aims to identify specific types of entities in free text, such as person, location, and organization. Traditional sequence labeling methods (Ma and Hovy, 2016) have difficulty coping with nested entities, and recent works have transformed NER into other paradigms such as reading comprehension (Li et al., 2020; Shen et al., 2022), set prediction (Tan et al., 2021; Wu et al., 2022a) and sequence generation (Paolini et al., 2021; Yan et al., 2021; Lu et al., 2022). However, low-resource and cross-domain problems in practical scenarios still pose a great challenge to NER models.


C rounds
① Find person entities
② Find organization entities
⋮ ......
⑦ Find location entities
Locating
$\frac{N(N-1)}{2}$ rounds
① Jobs is a [MASK] entity
② Jobs was is a [MASK] entity
⋮ ......
⑩ San Francisco is a [MASK] entity
Typing
only 1 round
: ① { [Pᵢ] is a [Tᵢ] entity }$_{i=1,2,\cdots,\mathcal{M}}$
Locating & Typing

(a) Type-oriented

(b) Span-oriented

(c) PromptNER

Figure 1: A comparison of the type-oriented (a) and span-oriented (b) prompt learning with the proposed PromptNER (c). C, N and $\mathcal{M}$ denote the number of entity types, words and prompts, respectively.

Recently prompt learning (Liu et al., 2021a,b; Li and Liang, 2021; Lester et al., 2021) has received a lot of interest because of its excellent performance and data efficiency, and has been adopted in many classification and generation tasks (Gao et al., 2021; Schick and Schütze, 2021b; Ding et al., 2021a; Wu et al., 2022b). Prompt learning converts downstream tasks into language modeling tasks, where cloze questions are constructed as prompts to guide pre-trained language models to fill in the blanks. However, named entity recogni-

[†] Corresponding author.

[1] Our code will be available at https://github.com/tricktreat/PromptNER.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 12492–12507
July 9-14, 2023 ©2023 Association for Computational Linguistics

tion is a token-level tagging task, and it is difficult to apply prompt-based learning on NER directly (Liu et al., 2021a). Cui et al. (2021) proposes the template-based method, which constructs prompts for each potential entity span and then separately predicts their entity types. For example, given an input *"Jobs was born in San Francisco"*, Cui et al. (2021) enumerates each span to populate `[X]` of the template *"*`[X]` *is a* `[MASK]` *entity"*, and then determines the type of the filled span based on the prediction on the `[MASK]` slot. In contrast to entity typing over the enumerated spans, some methods (Li et al., 2020; Liu et al., 2022) design prompt templates from a symmetric perspective. They construct prompts for each entity type and then guide the model to locate specific types of entities. For example, Liu et al. (2022) constructs the prompt *"What is the location?"* for the `LOC` type, and then predicts all `LOC` entities in the sentence, e.g., *"San Francisco"*. We group these two types of methods into span-oriented and type-oriented prompt learning. As shown in Figure 1, they construct prompts based on the entity span or entity type, and then perform entity typing or entity locating. However, both groups of methods require multiple rounds of prompting. For an input with N words and C pre-fixed types, type-oriented and span-oriented prompt learning require C and $N(N-1)/2$ predictions, respectively. Moreover, each round of prediction is independent of the other, ignoring the latent relationships between different entities.

Different from the above methods that either perform multiple rounds of entity typing or entity locating through prompting, in this paper, we propose a prompt learning method for NER (**PromptNER**) that unifies entity locating and entity typing into one-round prompt learning. Specifically, we design the position slot `[P]` and the type slot `[T]` in the prompt template, which are used for prompting entity locating and typing accordingly. This manner is enumeration-free for entity span or entity type, and can locate and classify all entities in parallel, which improves the inference efficiency of the model. Since the correspondence between prompts and entities cannot be specified in advance, we need to assign labels to the slots in the prompts during training. Inspired by Carion et al. (2020), we treat the label assignment process as a linear assignment problem and perform bipartite graph match problem between the prompts and the entities. We further extend the traditional bipartite graph matching and design a one-to-many dynamic template filling mechanism so that an entity can be predicted by multiple prompts, which can improve the utilization of prompts. To summarize, our main contributions are as follows:

- We unify entity locating and entity typing for NER in prompt learning by filling both position and type slots in the dual-slot multi-prompt template. Our model eliminates the need to enumerate entity types or entity spans and can predict all entities in one round.
- For the model training, we design a dynamic template filling mechanism to assign labels for the position and type slots by an extended bipartite graph matching.
- We conduct experiments in a variety of settings, and we achieve significant performance improvements on both standard flat and nested NER datasets. In the cross-domain few-shot setting, our model outperforms the previous state-of-the-art models by +7.7% on average.

2 Related Work

2.1 Named Entity Recognition

Named Entity Recognition (NER) is a basic task of information extraction (Tjong Kim Sang and De Meulder, 2003; Wadden et al., 2019; Shen et al., 2021b; Tan et al., 2022). Current named entity recognition methods can be divided into four categories, including tagging-based, span-based, hypergraph-based, and generative-based methods. Traditional tagging-based methods (Ma and Hovy, 2016) predict a label for each word, which is difficult to cope with nested entities. Some works propose various strategies for improvement. For example, Alex et al. (2007) and Ju et al. (2018) use cascading or stacked tagging layers, and Wang et al. (2020) designs the tagging scheme with a pyramid structure. The span-based methods (Sohrab and Miwa, 2018) model NER as a classification task for spans directly, with the inherent ability to recognize nested entities. Due to the high cost of exhausting all spans, Zheng et al. (2019) and Shen et al. (2021a) propose boundary-aware and boundary-regression strategies based on span classification, respectively. Some other methods (Yu et al., 2020; Li et al., 2022) perform classification on inter-word dependencies or interactions, which are essentially span classification, and can also be considered as span-based methods. The generative-based methods (Yan et al., 2021; Lu et al., 2022; Zhang et al., 2022) are more general. They model the NER task as a sequence generation task that can unify the prediction of flat and nested entities.

In addition, some works focus on the NER task in practical settings, including the few-shot NER (Ding et al., 2021b) and the cross-domain NER (Liu et al., 2021c). For example, Chen et al. (2021) and Zhou et al. (2022) design data augmentation methods augment labeled data on low-resource domains. Some works (Ziyadi et al., 2020; Wiseman and Stratos, 2019) use the instance learning to perform a nearest neighbor search based on entity instances or token instances, and others (Ding et al., 2021b; Huang et al., 2021) use prototype networks at the token level or span level to handle such low-resource settings.

2.2 Prompt Learning

Prompt learning constructs prompts by injecting the input into a designed template, and converts the downstream task into a fill-in-the-blank task, then allows the language model to predict the slots in the prompts and eventually deduce the final output. Due to the data efficiency, prompt learning is currently widely used for many classification and generation tasks (Shin et al., 2020; Gao et al., 2021; Schick and Schütze, 2021b,a; Ding et al., 2021a). Some works investigate prompt learning on the extraction tasks. Cui et al. (2021) first applies prompt learning to NER. It proposes a straightforward way to construct separate prompts in the form of "[X] *is a* [MASK] *entity*" by enumerating all spans. The model then classifies the entities by filling the [MASK] slot. Since these methods need to construct templates and perform multiple rounds of inference, Ma et al. (2022) proposes a template-free prompt learning method using the mutual prediction of words with the same entity type. However, it requires constructing sets of words of the same entity type, which is difficult in low-resource scenarios. Lee et al. (2022) introduces demonstration-based learning in low-resource scenarios, they concatenate demonstrations in the prompts, including entity-oriented demonstrations and instance-oriented demonstrations. Another class of query-based methods (Li et al., 2020; Mengge et al., 2020; Liu et al., 2022) can also be categorized as prompt learning. In contrast to the above methods, they construct a type-related prompt (query), e.g. "*Who is the person ?*", and then lets the model locate all PER entities in the input. Different from all of the above, we unify entity locating and entity typing in prompt learning, and predict all entities in one round using a dual-slot multi-prompt template.

3 Method

In this section, we first introduce the task formulation in § 3.1, and then describe our method. The overview of the PromptNER is shown in Figure 2, and we will introduce the prompt construction in § 3.2 and the model inference in § 3.3, including the encoder and the entity decoding module. The training of the model requires assigning labels to the slots of the prompt, and we will introduce the dynamic template filling mechanism in § 3.4.

3.1 Task Formulation

Following Cui et al. (2021) and Lee et al. (2022), we transform the NER task into a fill-in-the-blank task. Given a sentence X of length N, we fill a fixed number $\mathcal{M}$ of prompts and X into a predefined template to construct the complete input sequence $\mathcal{T}$. The model then fills the position slots [P] and type slots [T] of all prompts and decodes the named entities in the sentence.

3.2 Prompt Construction

Different from the previous methods, we unify entity locating and entity typing into one-round prompt learning. Therefore, we have two improvements in prompt construction. First, each prompt has two slots, entity position slot and entity type slot, which are used for entity locating and entity typing respectively. Second, our model fills slots for a predefined number of prompts simultaneously and extracts all entities in a parallel manner. Specifically, the constructed input sequence consists of two parts: $\mathcal{M}$ prompts and the input sentence X. By default, each prompt has only two tokens: a position slot [P] and a type slot [T]. For a sentence $X =$ "*Jobs was born in San Francisco*", the default dual-slot multi-prompt input sequence can be represented as:

$$\mathcal{T} = \begin{array}{l} \{[\mathtt{P}_i] \textit{ is a } [\mathtt{T}_i] \textit{ entity} \}_{i=1,2,\cdots,\mathcal{M}} \\ [\mathtt{CLS}] \textit{ Jobs was born in San Francisco.} \end{array}$$

where " $[\mathtt{P}_i]$ *is a* $[\mathtt{T}_i]$ *entity* " is the i-th prompt, $[\mathtt{P}_i]$ and $[\mathtt{T}_i]$ denote its position and type slots and $\mathcal{M}$ denotes the number of prompts. Following

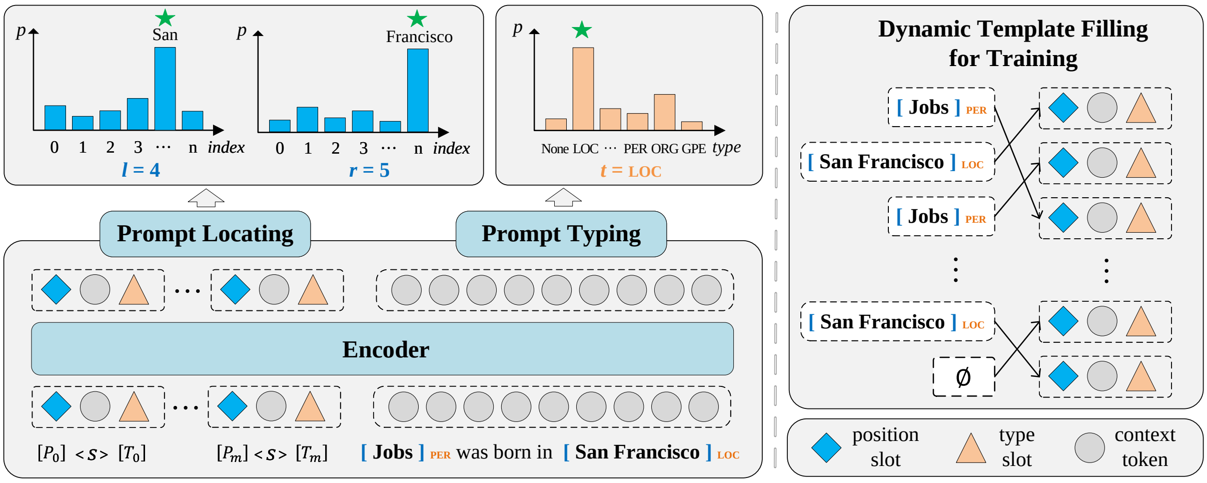


Figure 2: An overview of PromptNER. The left part describes the model's inference process and the right part describes the dynamic template filling mechanism during training. The model takes a dual-slot multi-prompt sequence as input and fills in the position slot '◆' and type slot '△' by prompt locating and prompt typing.

Lester et al. (2021); Gao et al. (2021), we also experiment with soft templates by replacing concrete contextual words with learnable tokens. In § 5.3 we compare the performance of the model using different templates.

3.3 Prompt Locating and Typing

With the input sequence $\mathcal{T}$ filled with the sentence X and $\mathcal{M}$ prompts, the model decodes the entities by filling the position slots $\texttt{[P}_i\texttt{]}_{i=1,2,\cdots,\mathcal{M}}$ and type slots $\texttt{[T}_i\texttt{]}_{i=1,2,\cdots,\mathcal{M}}$ of $\mathcal{M}$ prompts.

Encoder We first use BERT (Devlin et al., 2019) to encode the input sequence $\mathcal{T}$:

$$\mathbf{H}^{\mathcal{T}} = \text{BERT}\left(\mathcal{T}\right)$$

Note that in order to encode the sentence X independent of the prompts, we block the attention of the prompts to the sentence by a prompt-agnostic attention mask, which has a lower left submatrix of size $n \times k$ as a full $-inf$ matrix, where k is the length of the prompt sequence. Then by indexing on the corresponding position of $\mathbf{H}^{\mathcal{T}}$, we can obtain the encoding of the sentence X and the encodings of the two types of slots, denoted as $\mathbf{H}^X$, $\mathbf{H}^P$ and $\mathbf{H}^T$, where $\mathbf{H}^P, \mathbf{H}^T \in \mathbb{R}^{\mathcal{M}\times h}$ and $\mathbf{H}^X \in \mathbb{R}^{n\times h}$ and h is the hidden size.

To enhance the interaction of different prompts, we designed extra prompt interaction layers. Each interaction layer contains self-attention between slots with the same type (the key, query and value are the encodings of slots) and cross-attention from sentence to prompt slots (the query is the encodings of slots while the key and value are the sentence encodings). Thus the final encodings of position and type slots ($\delta \in \{P, T\}$) are computed as follows:

$$\hat{\mathbf{H}}^{\delta} = \text{PromptInteraction}\left(\mathbf{H}^{\delta} + \mathbf{E}_{id}, \mathbf{H}^X\right)$$

where $\mathbf{E}_{id} \in \mathbb{R}^{\mathcal{M}\times h}$ denote the learnable identity embeddings of $\mathcal{M}$ prompts, which can bind position slot and type slot within the same prompt.

Entity Decoding Now we can decode the corresponding entity for each prompt by prompt locating and prompt typing, i.e., filling the position slot and type slot of the prompt. For the i-th prompt, we put its type slot $\hat{\mathbf{H}}_i^T$ through a classifier and get the probabilities of different entity types as:

$$\mathbf{p}_i^t = \text{Classifier}\left(\hat{\mathbf{H}}_i^T\right)$$

where the classifier is a linear layer followed by the softmax function. For prompt locating, we need to determine whether the j-th word is the start or end word of the predicted entity by the i-th prompt. We first feed the position slot $\hat{\mathbf{H}}^P$ into a linear layer, and then add it with the word representation $\mathbf{H}^X$ of each position to get the fusion representations $\mathbf{H}^F$. We then perform binary classification to obtain the probability of the j-th word being the left boundary of the predicted entity for the i-th prompt:

$$\begin{aligned}\mathbf{H}^F &= \mathbf{W}_1\hat{\mathbf{H}}_i^P + \mathbf{W}_2\mathbf{H}^X \\ \mathbf{p}_{ij}^l &= \text{Sigmoid}\left(\mathbf{W}_3\mathbf{H}_{ij}^F\right)\end{aligned}$$

where $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3 \in \mathbb{R}^{h\times h}$ are learnable weights. In the same way, we can compute the probability $\mathbf{p}^r_{ij}$ of the j-th word being the right boundary. Then the probabilities of the entities predicted by the $\mathcal{M}$ prompts can be denoted as $\hat{\mathbf{Y}} = \{\hat{\mathbf{Y}}_i\}_{i=1}^{\mathcal{M}}$, where $\hat{\mathbf{Y}}_i = (\mathbf{p}^l_i, \mathbf{p}^r_i, \mathbf{p}^t_i)$ [2].

Inference During inference, we can get the left boundary, right boundary and type of the entity corresponding to the i-th prompt as $\left(\operatorname{argmax}\mathbf{p}^l_i, \operatorname{argmax}\mathbf{p}^r_i, \operatorname{argmax}\mathbf{p}^t_i\right)$. When two prompts yield identical entities, we keep only one; for conflicting candidates, such as entities with the same location but inconsistent types, we keep the entity with the highest probability.

3.4 Dynamic Template Filling

Since the correspondence between prompts and entities is unknown, we cannot assign labels to the slots in advance. To solve it, we treat slot filling as a linear assignment problem[3], where any entity can be filled to any prompt, incurring a cost, and we need to get the correspondence between the prompts and the entities with minimum overall cost. We propose a dynamic template filling mechanism to perform bipartite graph matching between prompts and the entities. Let us denote the gold entities as $\mathbf{Y} = \{(l_i, r_i, t_i)\}_{i=1}^{K}$, where K denotes the number of entities and l_i, r_i, t_i are the boundary indices and type for the i-th entity. We pad $\mathbf{Y}$ with $\varnothing$ to ensure that it has the same number $\mathcal{M}$ as the prompts. Then the permutation of the prompts corresponding to the optimal match is:

$$\sigma^\star = \underset{\sigma\in\mathfrak{S}(\mathcal{M})}{\arg\min} \sum_{i=1}^{\mathcal{M}} \mathcal{C}ost_{match}\left(\mathbf{Y}_i, \hat{\mathbf{Y}}_{\sigma(i)}\right)$$

where $\mathfrak{S}(\mathcal{M})$ is the set of all $\mathcal{M}$-length permutations and $\mathcal{C}ost_{match}\left(\mathbf{Y}_i, \hat{\mathbf{Y}}_{\sigma(i)}\right)$ is the pairwise match cost between the i-th entity and the prediction by the $\sigma(i)$-th prompt, we define it as $-\mathbb{1}_{\{t_i\neq\varnothing\}}\left[\mathbf{p}^t_{\sigma(i)}(t_i) + \mathbf{p}^l_{\sigma(i)}(l_i) + \mathbf{p}^r_{\sigma(i)}(r_i)\right]$, where $\mathbb{1}_{\{\cdot\}}$ denotes an indicator function.

Traditional bipartite graph matching is one-to-one, with each gold entity matching only one prompt, which leads to many prompts being matched to $\varnothing$, thus reducing the training efficiency. To improve the utilization of prompts, we extend the one-to-one bipartite graph matching to one-to-many, which ensures that a single gold entity can be matched by multiple prompts. To perform one-to-many matching, we simply repeat the gold entities to augment $\mathbf{Y}$ under a predefined upper limit U. In our experiments, we take $U = 0.9\mathcal{M}$. We use the Hungarian algorithm (Kuhn, 1955) to solve Equation 3.4 for the optimal matching $\sigma^\star$ at minimum cost. Then the losses for prompt locating ($\mathcal{L}_2$) and typing ($\mathcal{L}_1$) are computed as follows:

$$\mathcal{L}_1 = -\sum_{i=1}^{\mathcal{M}} \log \mathbf{p}^t_{\sigma^\star(i)}(t_i)$$

$$\mathcal{L}_2 = -\sum_{i=1}^{\mathcal{M}} \mathbb{1}_{t_i\neq\varnothing}\left[\log \mathbf{p}^l_{\sigma^\star(i)}(l_i) + \log \mathbf{p}^r_{\sigma^\star(i)}(r_i)\right]$$

and the final loss is the weighted sum $\mathcal{L} = \lambda_1\mathcal{L}_1 + \lambda_2\mathcal{L}_2$. By default we set $\lambda_1 = 1$ and $\lambda_2 = 2$.

4 Experiments

To verify the effectiveness of PromptNER in various settings, we conduct extensive experiments in flat and nested NER (§ 4.3) and low-resource NER, including in-domain few-shot setting (§ 4.4) and cross-domain few-shot setting (§ 4.5).

4.1 Implementation Details

If not marked, we use BERT-large (Devlin et al., 2019) as the backbone of the model. We use reserved tokens and sparse tokens of BERT, e.g. `[unused1]-[unused100]`, as position and type slots. The model has a hidden size $h = 1024$ and $\mathcal{I} = 3$ prompt interaction layers. Since the maximum number of entities per sentence does not exceed 50, we uniformly set the number of prompts $\mathcal{M} = 50$. In the dynamic template filling mechanism, we set the upper limit of the expanded labels $U = 0.9\mathcal{M} = 45$ for extended bipartite graph matching. For all datasets, we train PromptNER for 50-100 epochs and use the Adam (Kingma and Ba, 2015), with a linear warmup and linear decay learning rate schedule and a peak learning rate of $2e$-5. We initialize our prompt identity embeddings $\mathbf{E}_{id}$ with the normal distribution $\mathcal{N}(0.0, 0.02)$.

4.2 Warmup Training

Before employing PromptNER in the low-resource scenario, we use the open Wikipedia data to warm up the training for entity locating. PromptNER needs to elicit the language model to locate entities, while the pre-trained language model does not learn entity localization during pre-training. Therefore

[2] $\mathbf{p}^\alpha_i = [\mathbf{p}^\alpha_{i0}, \mathbf{p}^\alpha_{i1}, \ldots, \mathbf{p}^\alpha_{iN}]$, where $\alpha \in \{l, r\}$

[3] https://en.wikipedia.org/wiki/Assignment_problem

Model	ACE04			ACE05			CoNLL03		
	Pr.	Rec.	F1	Pr.	Rec.	F1	Pr.	Rec.	F1
Biaffine (Yu et al., 2020)	87.30	86.00	86.70	85.20	85.60	85.40	93.70	93.30	93.50
MRC (Li et al., 2020)	85.05	86.32	85.98	87.16	86.59	86.88	92.33	94.61	93.04
BARTNER (Yan et al., 2021)	87.27	86.41	86.84	83.16	86.38	84.74	92.61	93.87	93.24
Seq2Set (Tan et al., 2021)	88.46	86.10	87.26	87.48	86.63	87.05	-	-	-
Triaffine (Yuan et al., 2022)	87.13	87.68	87.40	86.70	86.94	86.82	-	-	-
UIE (Lu et al., 2022)	-	-	86.89	-	-	85.78	-	-	92.99
W^{2}NER (Li et al., 2022)	87.33	87.71	87.52	85.03	88.62	86.79	92.71	93.44	93.07
BuParser(Yang and Tu, 2022)	86.60	87.28	86.94	84.61	86.43	85.53	-	-	-
LLCP (Lou et al., 2022)	87.39	88.40	87.90	85.97	87.87	86.91	-	-	-
PIQN (Shen et al., 2022)	88.48	87.81	88.14	86.27	88.60	87.42	93.29	92.46	92.87
BS [BERT-large] (Zhu and Li, 2022)	-	-	87.85	-	-	87.82	-	-	93.08
BS [RoBERTa-large] (Zhu and Li, 2022)	-	-	88.52	-	-	88.14	-	-	**93.77**
PromptNER [BERT-large]	87.58	88.76	88.16	86.07	88.38	87.21	92.48	92.33	92.41
PromptNER [RoBERTa-large]	**88.64**	**88.79**	**88.72**	**88.15**	88.38	**88.26**	92.96	93.18	93.08

Table 1: Results in the standard *flat* and *nested* NER setting.

PromptNER needs to learn the prompt locating ability initially by Wiki warmup training. We choose accessible Wikipedia as our warm-up training data. Wikipedia contains a wealth of entity knowledge (Yamada et al., 2020; Wang et al., 2022) that is useful for entity-related tasks such as named entity recognition, relation extraction, entity linking, etc. We call entity-related hyperlinks in Wikipedia as wiki anchors. These anchors only have position annotations and lack type information, and we use these partially annotated noisy data to warm up the localization ability of PromptNER. Specifically, we fix the weight of BERT, train 3 epochs with a learning rate of $1e$-5 on the constructed wiki anchor data, and optimize the model only on the entity locating loss to warm up the entity decoding module. In low-resource scenarios (in-domain few-shot setting in § 4.4 and cross-domain few-shot setting in § 4.5), we initialize PromptNER with the warmed-up weights.

4.3 Standard Flat and Nested NER Setting

Datasets We adopt three widely used datasets to evaluate the performance of the model in the standard NER setting, including one flat NER dataset: CoNLL03 (Tjong Kim Sang and De Meulder, 2003) and two nested NER datasets: ACE04 (Doddington et al., 2004) and ACE05 (Walker et al., 2006). For ACE04 and ACE05, we use the splits of Lu and Roth (2015); Muis and Lu (2017) and the preprocessing protocol of Shibuya and Hovy (2020). Please refer to Appendix A.1 for detailed statistics on nested entities about ACE04 and ACE05. For CoNLL03, we follow Lample et al. (2016); Yu et al. (2020); Jin et al. (2023) to train the model on the concatenation of the train and dev sets.

Baselines We select recent competitive models as our baseline, including span-based (Yuan et al., 2022; Li et al., 2022), generation-based (Tan et al., 2021; Yan et al., 2021; Lu et al., 2022), MRC-based (Li et al., 2020; Shen et al., 2022; Jin et al., 2022), and parsing-based (Yu et al., 2020; Zhu and Li, 2022; Lou et al., 2022; Yang and Tu, 2022). These methods adopt different pre-trained language models as the encoder, thus in the experimental results, we provide the performance of PromptNER on BERT-large and RoBERTa-large.

Results Table 1 illustrates the performance of PromptNER as well as baselines on the flat and nested NER datasets. We observe that PromptNER outperforms most of the recent competitive baselines. When using RoBERTa-large as the encoder, PromptNER outperforms previous state-of-the-art models on the nested NER datasets, achieving F1-scores of 88.72% and 88.26% on ACE04 and ACE05 with +0.20% and +0.12% improvements. And on the flat NER dataset CoNLL03, PromptNER achieves comparable performance compared to the strong baselines. We also evaluate the performance of entity locating and entity typing separately on ACE04, please refer to Appendix A.2.

4.4 In-Domain Few-Shot NER Setting

Datasets and Baselines Following Cui et al. (2021), we construct a dataset with low-resource scenarios based on CoNLL03. We limit the number of entities of specific types by downsampling and meet the low-resource requirement on these types.

Models	ORG	PER	LOC*	MISC*	Overall
BERTTagger	75.32	76.25	61.55	59.35	68.12
TemplateNER	72.61	84.49	71.98	**73.37**	75.59
PromptNER	**76.96**	**88.11**	**82.69**	62.89	**79.75**

Table 2: Results in the in-domain few-shot NER setting. * indicates the low-resource entity type.

Specifically, we set LOC and MISC as low-resource types and PER and ORG as resource-rich types. We downsample the CoNLL03 training set to obtain 4,001 training samples, including 100 MISC, 100 LOC, 2496 PER, and 3763 ORG entities. We use this dataset to evaluate the performance of PromptNER under the in-domain few-shot NER setting. We choose BERTTagger (Devlin et al., 2019) and the low-resource friendly model TemplateNER (Cui et al., 2021) as our baselines.

Results As shown in Table 2, we achieve significant performance improvements on both low and rich resource types compared to BERTTagger. In particular, we achieve an average +12.34% improvement on low-resource types. Prompt design is the key to prompt learning (Liu et al., 2021a), and our method adaptively learns them by the dynamic template filling mechanism which can achieve better performance in low resource scenarios. Compared to TemplateNER, PromptNER performs better in the low-resource LOC type and overall, and slightly weaker in MISC type. We believe that entities of type MISC are more diverse and it is hard for PromptNER to learn a clear decision boundary from a small number of support instances.

4.5 Cross-Domain Few-Shot NER Setting

Datasets and Baselines In practical scenarios, we can transfer the model from the resource-rich domain to enhance the performance of the low-resource domain. In this setting, the entity types of the target domain are different from the source domain, and only a small amount of labeled data is available for training. To simulate the cross-domain few-shot setting, we set the source domain as the resource-rich CoNLL03 dataset, and randomly sample some training instances from the MIT movie, MIT restaurant, and ATIS datasets as the training data for the target domain. Specifically, we randomly sample a fixed number of instances for each entity type (10, 20, 50, 100, 200, 500 instances per entity type for MIT movie and MIT restaurant, and 10, 20, 50 instances per entity type for ATIS). If the number of instances of a type is less than the fixed number, we use all instances for training. We select several competitive methods with the same experimental setup as our baselines, including NeighborTagger (Wiseman and Stratos, 2019), Example-based (Ziyadi et al., 2020), MP-NSP (Huang et al., 2021), BERTTagger (Devlin et al., 2019), and TemplateNER (Cui et al., 2021).

Results Table 3 shows the performance of PromptNER in the cross-domain few-shot setting, along with some strong baselines. We observe that PromptNER achieves the best performance on all settings of fixed support instances for the three datasets. At the extreme 10-shot setting, PromptNER outperforms TemplateNER by +13.2%, +3%, and +14.2% on the MIT Movie, MIT Restaurant, and ATIS datasets, respectively. Overall, compared to the previous state-of-the-art model, PromptNER achieves a +7.7% improvement on average in all cross-domain few-shot settings. This shows that PromptNER can transfer the generalized knowledge learned in the resource-rich domain to the low-resource domain. Furthermore, PromptNER can decouple entity locating and typing via position and type slots, which is especially suitable for cross-domain scenarios with syntactic consistency and semantic inconsistency.

5 Analysis

5.1 Ablation Study

We conduct ablation experiments on ACE04 to analyze the effect of different modules of PromptNER. The experimental results are shown in Table 4, without the three practices, there is a different degradation of model performance. If we assign labels to slots simply by entity order or use one-to-one bipartite graph matching, the model performance decreases by 3.43% and 4.11%, respectively. We conclude that a one-to-many dynamic template-filling mechanism is important as it allows prompts fit to related entities adaptively. The one-to-many manner ensures that an entity can be predicted by multiple prompts, improving the model prediction tolerance. When encoding the input sequence, it is also important to keep the sentence encoding to be prompt agnostic, resulting in a +0.42% performance improvement.

5.2 Analysis of $\mathcal{M}$ and $\mathcal{I}$

We further investigate the effect of the number of prompts and the number of prompt interaction lay-

Methods	MIT Movie						MIT Restaurant						ATIS			Avg.
	10	20	50	100	200	500	10	20	50	100	200	500	10	20	50	
NeighborTagger	3.1	4.5	4.1	5.3	5.4	8.6	4.1	3.6	4.0	4.6	5.5	8.1	2.4	3.4	5.1	4.8
Example-based	40.1	39.5	40.2	40.0	40.0	39.5	25.2	26.1	26.8	26.2	25.7	25.1	22.9	16.5	22.2	30.4
MP-NSP	36.4	36.8	38.0	38.2	35.4	38.3	46.1	48.2	49.6	49.6	50.0	50.1	71.2	74.8	76.0	49.2
BERTTagger	28.3	45.2	50.0	52.4	60.7	76.8	27.2	40.9	56.3	57.4	58.6	75.3	53.9	78.5	92.2	56.9
TemplateNER	42.4	54.2	59.6	65.3	69.6	80.3	53.1	60.3	64.1	67.3	72.2	75.7	77.3	88.9	93.5	68.3
PromptNER	**55.6**	**68.2**	**76.5**	**80.4**	**82.9**	**84.5**	**56.1**	**62.6**	**69.3**	**71.3**	**74.4**	**77.4**	**91.5**	**94.3**	**95.5**	**76.0**

Table 3: Results in the cross-domain few-shot NER setting. We transfer the model from the general domain (CoNLL03) to specific target domains with only a few labeled instances: Movie Review, Restaurant Review, ATIS.

Model	Pr.	Rec.	F1
DEFAULT	**87.58**	**88.76**	**88.16**
w/o Dyn. Template Filling	86.19	83.32	84.73
w/o Extended Labels	84.46	83.65	84.05
w/o Prompt-agnostic Mask	87.59	87.90	87.74

Table 4: Ablation study. (1) *w/o Dyn. Template Filling*: static template filling, filling the slots of the prompts according to the occurrence order of the entities; (2) *w/o Extended Labels*: no label expansion in the dynamic template filling mechanism, i.e., using the traditional one-to-one bipartite graph matching; (3) *w/o Prompt-agnostic Mask*: using the original BERT for encoding.

ers on PromptNER. From Figure 3, we can observe that the number of prompts is more appropriate between 50 and 60. Too few would make it difficult to cover all entities, and too many would exceed the maximum length of the encoding and impair the model performance. In addition, as the number of interaction layers increases, we can observe a significant performance improvement in Figure 3. This suggests that the interaction between prompts can model the connection between entities. Considering the size and efficiency of the model, we choose $\mathcal{M}$=50, $\mathcal{I}$=3 as the default setting.

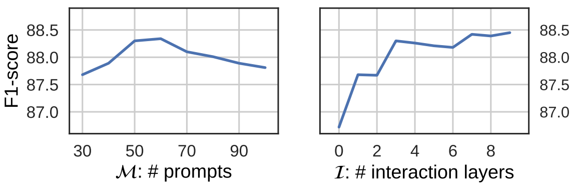


Figure 3: F1-scores under different number of prompts $\mathcal{M}$ and interaction layers $\mathcal{I}$ on ACE04 dataset

5.3 Analysis of Prompt Templates

Templates are important for prompt learning (Gao et al., 2021; Ding et al., 2021b). In this section, we conduct experiments on ACE04 to analyze the effect of different templates, as shown in Table 5. Contrary to intuition, inserting hard or soft contextual tokens in the prompts does not improve the model performance. We argue that adding contextual tokens to our multi-prompt template significantly grows the input sequence (each additional token increases the total length by $\mathcal{M}$), and the long sequence may exceed the maximum encoding length of BERT. Comparing hard and soft templates, we find that soft templates are more useful, which is consistent with Ding et al. (2021b).

Type	Template	F1
Hard	$\{$[P$_i$] *is a* [T$_i$]$\}_{i=1,2,\cdots,50}$ *entity* [CLS]*Jobs was born in San Francisco.*	87.96
Soft	$\{$[P$_i$]<s>[T$_i$]$\}_{i=1,2,\cdots,50}$[CLS]*Jobs was born in San Francisco.*	88.05
Default	$\{$[P$_i$] [T$_i$]$\}_{i=1,2,\cdots,50}$ [CLS] *Jobs was born in San Francisco.*	88.16

Table 5: A comparison of different templates. <s> is a learnable contextual token and the default template contains only slots without any contextual tokens.

5.4 Inference Efficiency

Theoretically, for a sentence with N words and C potential entity types, type-oriented (Li et al., 2020) and span-oriented (Cui et al., 2021) prompt learning need to be run C and $N(N-1)/2$ times. And the generation-based methods (Yan et al., 2021) generate entity sequences in an autoregressive manner. Assuming that the length of the entity sequence is T, it takes T steps to decode all entities. In contrast, PromptNER can locate and typing the entities in parallel through dual-slot multi-prompt learning, it only needs one run to decode all the entities. Under the same experimental setup, we compare their inference efficiency on CoNLL03, as shown in Table 6. Empirically, PromptNER achieves the fastest inference efficiency compared to the baselines, with 48.23×, 1.86× and 2.39× faster than TemplateNER, MRC and BARTNER, respectively.

Model	Complexity	SpeedUp
TempNER (Cui et al., 2021)	$O(N^2)$	1.00×
MRC (Li et al., 2020)	$O(C)$	25.86×
BARTNER (Yan et al., 2021)	$O(T)$	20.17×
PromptNER	$O(1)$	48.23×

Table 6: A comparison of inference efficiency on the test set of CoNLL03. All experiments were conducted with one NVIDIA GeForce RTX 3090 graphics card.

6 Conclusion

In this paper, we unify entity locating and entity typing in prompt learning for NER with a dual-slot multi-prompt template. By filling position slots and type slots, our proposed model can predict all entities in one round. We also propose a dynamic template filling mechanism for label assignment, where the extended bipartite graph matching assigns labels to the slots in a one-to-many manner. We conduct extensive experiments in various settings including flat and nested NER and low-resource in-domain and cross-domain NER, and our model achieves superior performance compared to the competitive baselines.

Limitations

We discuss here the limitations of the proposed PromptNER. First, although PromptNER performs well on flat and nested NER, it cannot recognize discontinuous entities. The discontinuous entity can be divided into multiple fragments, while each position slot of PromptNER can only fill one. A simple alternative is to expand the position slots in prompts to accommodate discontinuous entities. Second, named entity recognition requires pretrained language models (PLMs) with the essential ability to sense the structure and semantics of entities, which can enhance entity locating and entity typing in low-resource scenarios. However, since PLMs prefer to learn semantic rather than structured information in the pre-training stage, PromptNER needs to be warmed up by Wiki training when applied to low-resource scenarios. Finally, since the number of prompts is determined during training, there is a limit to the number of entities that the model can recognize. If the number of entities in a sentence exceeds the pre-specified value when testing, PromptNER will perform poorly.

Acknowledgments

This work is supported by the Key Research and Development Program of Zhejiang Province, China (No. 2023C01152, No. 2022C01011), the Fundamental Research Funds for the Central Universities (No. 226-2023-00060), and MOE Engineering Research Center of Digital Library.

References

Beatrice Alex, Barry Haddow, and Claire Grover. 2007. Recognising nested named entities in biomedical text. In *Biological, translational, and clinical language processing*, pages 65–72, Prague, Czech Republic. Association for Computational Linguistics.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *Computer Vision – ECCV 2020*, pages 213–229, Cham. Springer International Publishing.

Shuguang Chen, Gustavo Aguilar, Leonardo Neves, and Thamar Solorio. 2021. Data augmentation for cross-domain named entity recognition. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5346–5356, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using BART. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1835–1845, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. 2021b. Few-NERD: A few-shot named entity recognition dataset. 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 3198–3213, Online. Association for Computational Linguistics.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. 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 3816–3830, Online. Association for Computational Linguistics.

Jiaxin Huang, Chunyuan Li, Krishan Subudhi, Damien Jose, Shobana Balakrishnan, Weizhu Chen, Baolin Peng, Jianfeng Gao, and Jiawei Han. 2021. Few-shot named entity recognition: An empirical baseline study. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10408–10423, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Weiqiang Jin, Biao Zhao, and Chenxing Liu. 2023. Fintech key-phrase: A new chinese financial high-tech dataset accelerating expression-level information retrieval. In *Database Systems for Advanced Applications*, pages 425–440, Cham. Springer Nature Switzerland.

Weiqiang Jin, Biao Zhao, Hang Yu, Xi Tao, Ruiping Yin, and Guizhong Liu. 2022. Improving embedded knowledge graph multi-hop question answering by introducing relational chain reasoning. *Data Mining and Knowledge Discovery*.

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A neural layered model for nested named entity recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1446–1459, New Orleans, Louisiana. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3th International Conference on Learning Representations, ICLR 2021*.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Dong-Ho Lee, Akshen Kadakia, Kangmin Tan, Mahak Agarwal, Xinyu Feng, Takashi Shibuya, Ryosuke Mitani, Toshiyuki Sekiya, Jay Pujara, and Xiang Ren. 2022. Good examples make a faster learner: Simple demonstration-based learning for low-resource NER. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2687–2700, Dublin, Ireland. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jingye Li, Hao Fei, Jiang Liu, Shengqiong Wu, Meishan Zhang, Chong Teng, Donghong Ji, and Fei Li. 2022. Unified named entity recognition as word-word relation classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10965–10973.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. 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 4582–4597, Online. Association for Computational Linguistics.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859, Online. Association for Computational Linguistics.

Andy T. Liu, Wei Xiao, Henghui Zhu, Dejiao Zhang, Shang-Wen Li, and Andrew Arnold. 2022. Qaner: Prompting question answering models for few-shot named entity recognition.

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

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. Gpt understands, too.

Zihan Liu, Yan Xu, Tiezheng Yu, Wenliang Dai, Ziwei Ji, Samuel Cahyawijaya, Andrea Madotto, and Pascale Fung. 2021c. Crossner: Evaluating cross-domain named entity recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13452–13460.

Chao Lou, Songlin Yang, and Kewei Tu. 2022. Nested named entity recognition as latent lexicalized constituency parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6183–6198, Dublin, Ireland. Association for Computational Linguistics.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 857–867, Lisbon, Portugal. Association for Computational Linguistics.

Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022. Unified structure generation for universal information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5755–5772, Dublin, Ireland. Association for Computational Linguistics.

Ruotian Ma, Xin Zhou, Tao Gui, Yiding Tan, Linyang Li, Qi Zhang, and Xuanjing Huang. 2022. Template-free prompt tuning for few-shot NER. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5721–5732, Seattle, United States. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Xue Mengge, Bowen Yu, Zhenyu Zhang, Tingwen Liu, Yue Zhang, and Bin Wang. 2020. Coarse-to-Fine Pre-training for Named Entity Recognition. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6345–6354, Online. Association for Computational Linguistics.

Aldrian Obaja Muis and Wei Lu. 2017. Labeling gaps between words: Recognizing overlapping mentions with mention separators. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2608–2618, Copenhagen, Denmark. Association for Computational Linguistics.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages. In *9th International Conference on Learning Representations, ICLR 2021*.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Yongliang Shen, Xinyin Ma, Zeqi Tan, Shuai Zhang, Wen Wang, and Weiming Lu. 2021a. Locate and label: A two-stage identifier for nested named entity recognition. 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 2782–2794, Online. Association for Computational Linguistics.

Yongliang Shen, Xinyin Ma, Yechun Tang, and Weiming Lu. 2021b. A trigger-sense memory flow framework for joint entity and relation extraction. In *Proceedings of the Web Conference 2021*, WWW '21, page 1704–1715, New York, NY, USA. ACM.

Yongliang Shen, Xiaobin Wang, Zeqi Tan, Guangwei Xu, Pengjun Xie, Fei Huang, Weiming Lu, and Yueting Zhuang. 2022. Parallel instance query network for named entity recognition. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 947–961, Dublin, Ireland. Association for Computational Linguistics.

Takashi Shibuya and Eduard Hovy. 2020. Nested named entity recognition via second-best sequence learning and decoding. *Transactions of the Association for Computational Linguistics*, 8:605–620.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Mohammad Golam Sohrab and Makoto Miwa. 2018. Deep exhaustive model for nested named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2843–2849, Brussels, Belgium. Association for Computational Linguistics.

Zeqi Tan, Yongliang Shen, Xuming Hu, Wenqi Zhang, Xiaoxia Cheng, Weiming Lu, and Yueting Zhuang. 2022. Query-based instance discrimination network for relational triple extraction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7677–7690, Abu

Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zeqi Tan, Yongliang Shen, Shuai Zhang, Weiming Lu, and Yueting Zhuang. 2021. A sequence-to-set network for nested named entity recognition. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 3936–3942. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

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 5784–5789, Hong Kong, China. Association for Computational Linguistics.

Christopher Walker, Stephanie Strassel, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. linguistic. In *Linguistic Data Consortium, Philadelphia 57*.

Jue Wang, Lidan Shou, Ke Chen, and Gang Chen. 2020. Pyramid: A layered model for nested named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5918–5928, Online. Association for Computational Linguistics.

Xinyu Wang, Yongliang Shen, Jiong Cai, Tao Wang, Xiaobin Wang, Pengjun Xie, Fei Huang, Weiming Lu, Yueting Zhuang, Kewei Tu, Wei Lu, and Yong Jiang. 2022. DAMO-NLP at SemEval-2022 task 11: A knowledge-based system for multilingual named entity recognition. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pages 1457–1468, Seattle, United States. Association for Computational Linguistics.

Sam Wiseman and Karl Stratos. 2019. Label-agnostic sequence labeling by copying nearest neighbors. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5363–5369, Florence, Italy. Association for Computational Linguistics.

Shuhui Wu, Yongliang Shen, Zeqi Tan, and Weiming Lu. 2022a. Propose-and-refine: A two-stage set prediction network for nested named entity recognition. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 4418–4424. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Yiquan Wu, Yifei Liu, Weiming Lu, Yating Zhang, Jun Feng, Changlong Sun, Fei Wu, and Kun Kuang. 2022b. Towards interactivity and interpretability: A rationale-based legal judgment prediction framework. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4787–4799.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online. Association for Computational Linguistics.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various NER subtasks. 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 5808–5822, Online. Association for Computational Linguistics.

Songlin Yang and Kewei Tu. 2022. Bottom-up constituency parsing and nested named entity recognition with pointer networks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2403–2416, Dublin, Ireland. Association for Computational Linguistics.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics.

Zheng Yuan, Chuanqi Tan, Songfang Huang, and Fei Huang. 2022. Fusing heterogeneous factors with triaffine mechanism for nested named entity recognition. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3174–3186, Dublin, Ireland. Association for Computational Linguistics.

Shuai Zhang, Yongliang Shen, Zeqi Tan, Yiquan Wu, and Weiming Lu. 2022. De-bias for generative extraction in unified NER task. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 808–818, Dublin, Ireland. Association for Computational Linguistics.

Changmeng Zheng, Yi Cai, Jingyun Xu, Ho-fung Leung, and Guandong Xu. 2019. A boundary-aware neural model for nested named entity recognition. 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 357–366, Hong Kong, China. Association for Computational Linguistics.

Ran Zhou, Xin Li, Ruidan He, Lidong Bing, Erik Cambria, Luo Si, and Chunyan Miao. 2022. MELM: Data augmentation with masked entity language modeling for low-resource NER. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2251–2262, Dublin, Ireland. Association for Computational Linguistics.

Enwei Zhu and Jinpeng Li. 2022. Boundary smoothing for named entity recognition. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7096–7108, Dublin, Ireland. Association for Computational Linguistics.

Morteza Ziyadi, Yuting Sun, Abhishek Goswami, Jade Huang, and Weizhu Chen. 2020. Example-based named entity recognition. *CoRR*, abs/2008.10570.

A Appendix

A.1 Statistics of the nested NER datasets

In Table 7, we present statistics for the standard nested datasets: ACE04 and ACE05. We report the number of sentences (#S), the number of sentences containing nested entities (#NS), the average sentence length (AL), the number of entities (#E), the number of nested entities (#NE), the nesting rate (NR), and the maximum and the average number of entities (#AE) in sentences on the two datasets.

	ACE04			ACE05		
	Train	Dev	Test	Train	Dev	Test
#S	6198	742	809	7285	968	1058
#NS	2718	294	388	2797	352	339
#E	22204	2514	3035	24827	3234	3041
#NE	10159	1092	1417	10039	1200	1186
NR	45.75	43.44	46.69	40.44	37.11	39.00
AL	21.41	22.13	22.03	18.82	18.77	16.93
#ME	28	22	20	28	23	20
#AE	3.58	3.38	3.75	3.41	3.34	2.87

Table 7: Statistics for ACE04 and ACE05 datasets.

A.2 Analysis of Entity Locating and Typing

Our work unifies entity locating and entity typing in prompt learning, and in this section we compare the performance of the model on the two subtasks with some strong baselines. Following Shen et al. (2022), we consider entity locating correct when the left and right boundaries are correctly predicted. Based on the accurately located entities, we then evaluate the performance of entity typing. Figure 8 shows the performance comparison on ACE04, PromptNER significantly outperforms the baseline for both tasks, achieving +0.59% and +0.56% improvement in entity locating and entity typing compared to Shen et al. (2022).

Model	Pr.	Rec.	F1
Entity Locating			
Seq2set (Tan et al., 2021)	92.75	90.24	91.48
Locate&label (Shen et al., 2021a)	92.28	90.97	91.62
PIQN (Shen et al., 2022)	92.56	91.89	92.23
PromptNER	91.86	**93.80**	**92.82**
Entity Typing			
Seq2set (Tan et al., 2021)	95.36	86.03	90.46
Locate&label (Shen et al., 2021a)	95.40	86.75	90.87
PIQN (Shen et al., 2022)	95.59	87.81	91.53
PromptNER	95.15	**89.22**	**92.09**

Table 8: Analysis of entity locating and typing.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
the limitation section

☑ A2. Did you discuss any potential risks of your work?
the limitation section

☑ A3. Do the abstract and introduction summarize the paper's main claims?
the abstract section and introduction section

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 4.2, Section 4.3, Section 4.4

☑ B1. Did you cite the creators of artifacts you used?
Section 4.2, Section 4.3, Section 4.4

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Section 4.2, Section 4.3, Section 4.4

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Section 4.2, Section 4.3, Section 4.4

□ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 4.2, Section 4.3, Section 4.4

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4.2, Section 4.3, Section 4.4 and Section A.2

C ☑ Did you run computational experiments?

Section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4.1 and Section 5.3

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4.1

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4.1

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

□ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

□ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

□ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

□ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

□ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.