# LPNER: Label Prompt for Few-shot Nested Named Entity Recognition

**Jiaoyun Yang** JIAOYUN@HFUT.EDU.CN
**Zhihan Zhu** ZHUZHIHAN@MAIL.HFUT.EDU.CN
**Hong Ming** MING_HONG@MAIL.HFUT.EDU.CN
*Hefei University of Technology, Hefei, China*

**Lili Jiang** LILI.JIANG@CS.UMU.SE
*Umeå University, Umeå, Sweden*

**Ning An** NING.G.AN@ACM.ORG
*Hefei University of Technology, Hefei, China*

**Editors:** Vu Nguyen and Hsuan-Tien Lin

## Abstract

Few-shot Named Entity Recognition (NER) aims to identify named entities using very little annotated data. Recently, prompt-based few-shot NER methods have demonstrated significant effectiveness. However, most existing methods employ multi-round prompts, which significantly increase time and computational costs. Furthermore, current single-round prompt methods are mainly designed for flat NER tasks and are not effective in handling nested NER tasks. Additionally, these methods do not to fully utilize the semantic information of entity labels through prompts. To address these challenges, we propose a novel Label-Prompt-based few-shot nested NER method named LPNER, which not only handles nested NER tasks but also efficiently extracts semantic information of entities through label prompts, thereby achieving more efficient and accurate NER. LPNER first designs a specialized prompt based on a span strategy to enhance label semantics and effectively combines multiple span representations using special mark to obtain enhanced span representations integrated with label semantics. Then, entity prototypes are constructed through prototype network for classifying candidate entity spans. We conducted extensive experiments on five nested datasets: ACE04, ACE05, GENIA, GermEval, and NEREL. In 1-shot and 5-shot tasks, LPNER's $F_1$ scores mostly outperform baseline models.

**Keywords:** Nested named recognition; Few-shot learning; Prompt learning; Label semantics.

## 1. Introduction

Named Entity Recognition (NER), as a fundamental task in natural language processing, aims to locate and classify named entities such as locations, persons, and organizations from unstructured text. However, traditional fully supervised NER methods rely on a large amount of labeled data (Huang et al., 2015), and data annotation is a labor-intensive and time-consuming task requiring rich domain knowledge and expert experience. Therefore, few-shot named entity recognition, which aims to identify entities using a very limited number of labeled examples, poses a challenging and practical research problem that can alleviate manual workload and address cross-domain challenges. Few-shot NER tasks are typically categorized into few-shot flat NER and few-shot nested NER. In few-shot nested NER, entities are often nested (Finkel and Manning, 2009), meaning a single token may belong to

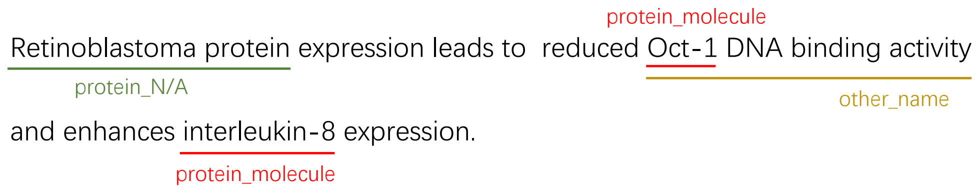


Figure 1: Example for nested entities from GENIA dataset: an entity "Oct-1" is nested in another entity "Oct-1 DNA binding activity".

multiple entities, as shown in Figure 1. This makes few-shot nested NER more challenging than few-shot flat NER.

To address the issue of small-sample situations, prompt learning has emerged as a new and popular paradigm in natural language processing, demonstrating significant potential. For few-shot NER tasks, several prompt-based methods have been proposed. Most of these methods use discrete prompts, manually constructing specific prompt templates to transform the NER task into a cloze task, thereby better leveraging pre-trained language models. For instance, templateNER (Cui et al., 2021) converts the NER task into a cloze task by defining a template like "<candidate_span> is a <entity_type>", and then uses a fine-tuned BART model to decode the prompt input for entity classification of text spans. QaNER (Liu et al., 2022) addresses the computational complexity and prompt robustness issues in templateNER by designing a simple prompt template "what is the <entity_type>", transforming the NER task into a Question Answering task. Similarly, Hou et al. (2022) also concerned with computational complexity, introduces an inverse prompt paradigm through "<entity_type> refers to _" for reverse prediction from entity type to entity span. It can be observed that since NER is a token-level task, for a prompt input with n tokens and k entity types, these prompt-based methods require n(n-1)/2 or k rounds of prompt predictions, leading to low computational efficiency.

To address the issues of computational complexity and efficiency, some researchers, inspired by metric learning, have cleverly combined prompt learning with metric learning to achieve entity span classification in a single round of prompts. COPNER (Huang et al., 2022) introduces class-specific word prompts as similarity measures and contrastive learning supervision signals, thus leveraging prompts to measure the similarity between samples and entity categories and to optimize token representations. Similarly, PromptNER (Zhang et al., 2023) adopts a scheme of constructing prompts from class-specific words, using the prompts not only for entity classification but also introducing k-nearest neighbor search based on ground truth entities in the support set as a basis for entity classification. PMRC (Huang et al., 2024) proposes a template based on label words and an example-based template, inserting special tokens "[ENT_START]" and "[ENT_END]" in prompts as boundary markers for various entity types, which serve as anchors for entity classification in subsequent processes. These methods achieve classification for all entities in a single round of prompts.

However, for the few-shot nested NER task, existing prompt-based learning methods have the following challenges: 1) Some methods use prompts that are only applicable to flat NER and cannot be adapted to nested NER, as they rely on sequence labeling, which lead to prompt overlap issues when handling nested entities; 2) Methods that can be applied to nested NER either require multiple rounds of prompts or fail to fully utilize label semantic information.

To solve these issues, we propose a novel Label-Prompt-based method for few-shot nested NER, which called LPNER. This method requires only one round of prompting and leverages prompts to fully exploit the semantic information of labels, enabling the label prototypes constructed from the support set samples to more accurately represent the distribution of various entity classes in the feature space. Firstly, for ease of model comprehension, we manually define a class-specific mapping, which converts entity labels into class-specific words in natural language for subsequent prompt construction. Then, to incorporate the semantic features of labels, we introduce a special mark. This special mark is applied to the entity span representations obtained from prompts and the original text, resulting in the final fused span representations that incorporate the semantic information of the labels.

In summary, our main contributions of this paper are summarized as follows:

- **Label Prompt Design for few-short nested NER:** We introduced a label prompt specifically for few-shot nested NER, allowing the model to better capture and utilize label semantics, leading to enhanced performance on this challenging task.
- **Efficient Integration of Prompt Learning and Prototype Networks:** Our LP-NER combines prompt learning with prototype networks, achieving few-shot nested NER with a single round of prompt prediction. This approach significantly reduces computational complexity and increases processing speed compared to methods requiring multiple rounds of prediction.
- **Demonstrated Effectiveness Across Multiple Datasets:** We validated our method through extensive experiments on five nested NER datasets (ACE04, ACE05, GENIA, GermEval, NEREL), consistently achieving higher $F_1$ scores than most existing methods.

## 2. Related Work

### 2.1. Few-shot Named Entity Recognition

For few-shot NER tasks, traditional NER methods struggle to remain effective due to the lack of training data (Wang et al., 2020). Thus, two main techniques have been proposed for few-shot NER tasks: transfer learning and meta-learning.

Transfer learning is a method of transferring knowledge from a solved task to an unsolved one. In the context of NER tasks, the approach involves training a model on a large NER dataset (often containing various entity types) initially, and then applying the trained model to a smaller dataset to help the model acquire missing semantic information.

Meta-learning aims to learn a generic model that can quickly adapt to new tasks with few training samples, without the need for retraining from scratch. The meta-learning methods primarily focus on small-sample learning and can be broadly categorized into three types: metric-based methods (Vinyals et al., 2016), optimization-based methods (Ravi and Larochelle, 2016), and memory-based methods (Li et al., 2020). Existing few-shot NER methods typically emphasize metric learning, utilizing representations learned in the semantic space to compute similarity for named entity recognition. For instance, ProtoNet (Snell et al., 2017) applies prototype networks to learn prototype representations for each entity class. NNShot (Yang and Katiyar, 2020) directly uses word embeddings as representations and applies nearest neighbor classification for inference.

### 2.2. Prompt Learning

GPT-3 (Brown et al., 2020) uses manual prompts to provide explicit task instructions and contextual information to the model, enabling it to better understand the tasks and generate high-quality outputs. This approach has achieved significant success across various natural language processing tasks. With the emergence of GPT-3 and other large language models, prompting learning has become increasingly popular and has demonstrated outstanding performance in NLP tasks. Templates play a crucial role in prompting learning, as appropriate templates can provide guiding information for downstream tasks, helping models better adapt to various NLP tasks and improve their performance and generalization capabilities on specific tasks. Current research focuses on manually constructing templates (Petroni et al., 2019), discrete prompt templates (Shin et al., 2020), and continuous prompt templates (Lester et al., 2021). Because prompting learning is well-suited for small-sample tasks with scarce training data, some studies have begun to introduce prompting learning into few-shot NER tasks. For example, templateNER (Cui et al., 2021) proposed a template-based prompting learning method, QaNER (Liu et al., 2022) transforms the NER task into a Question Answering (QA) task.

## 3. Problem Formulation

In this section, we formally introduce the problem formulation of few-shot nested named entity recognition.

Unlike token-level classification based on sequence labeling, we define the few-shot nested named entity recognition task as a span-based entity classification task. Given an input sequence $X = \{x_i\}_{i=1}^{L}$ consisting of $L$ tokens, we generate a set $M = \{\left(s_j, e_j\right)\}_{j=1}^{N}(1 \leq s_j \leq e_j \leq L)$ containing all possible spans, where $s_j$ and $e_j$ represent the start and end positions of the $j^{th}$ span in the sentence, and $N$ denotes the number of possible spans. Subsequently, we train a classification model that maps each span to one entity label from the label set $T_X$.

Suppose we have a source domain dataset $\mathcal{D}_{source}$ and a target domain dataset $\mathcal{D}_{target}$. During the training procedure, following Ding et al. (2021), we sample batches of training episode data from the source domain dataset $\mathcal{D}_{source}$. For each episode data $\varepsilon_{train} = \{S_{train}, Q_{train}, T_{train}\} \in \mathcal{D}_{source}$, where $S_{train}$ and $Q_{train}$ represent the support set and query set respectively, and $S_{train} \cap Q_{train} = \emptyset$. $T_{train}$ denotes the set of entity types in a training episode data. For each example $(X, M, \mathcal{Y})$ in the support set or query set, where $\mathcal{Y} = \{y_j\}_{j=1}^{N}$ is the set of entity labels and $y_j \in T_{train}$ is the label of the $j^{th}$ span $(s_j, e_j)$. Then, we train the model on these training episode data. During the test procedure, $\mathcal{D}_{target}$ is divided into a support set $\mathcal{D}_{target}^{spt}$ and a query set $\mathcal{D}_{target}^{qry}$. $\mathcal{D}_{target}^{spt}$ follows the $N$-way

$K$-shot setting, containing a few labeled sentences with $K$ entities for each entity category. The remaining data in $\mathcal{D}_{target}$ are unlabeled sentences used to construct $\mathcal{D}^{qry}_{target}$. Therefore, we first fine-tune our model on $\mathcal{D}^{spt}_{target}$ in the target domain, then make inferences for entity spans in $\mathcal{D}^{qry}_{target}$. It should be noted that the entity categories in $\mathcal{D}^{spt}_{target}$ are the same as those in $\mathcal{D}^{qry}_{target}$, but the sentences appearing in $\mathcal{D}^{spt}_{target}$ will not appear again in $\mathcal{D}^{qry}_{target}$.

## 4. Methodology

In this section, we will formally introduce our proposed LPNER, its various components, and target domain adaption procedures.

### 4.1. Model Framework

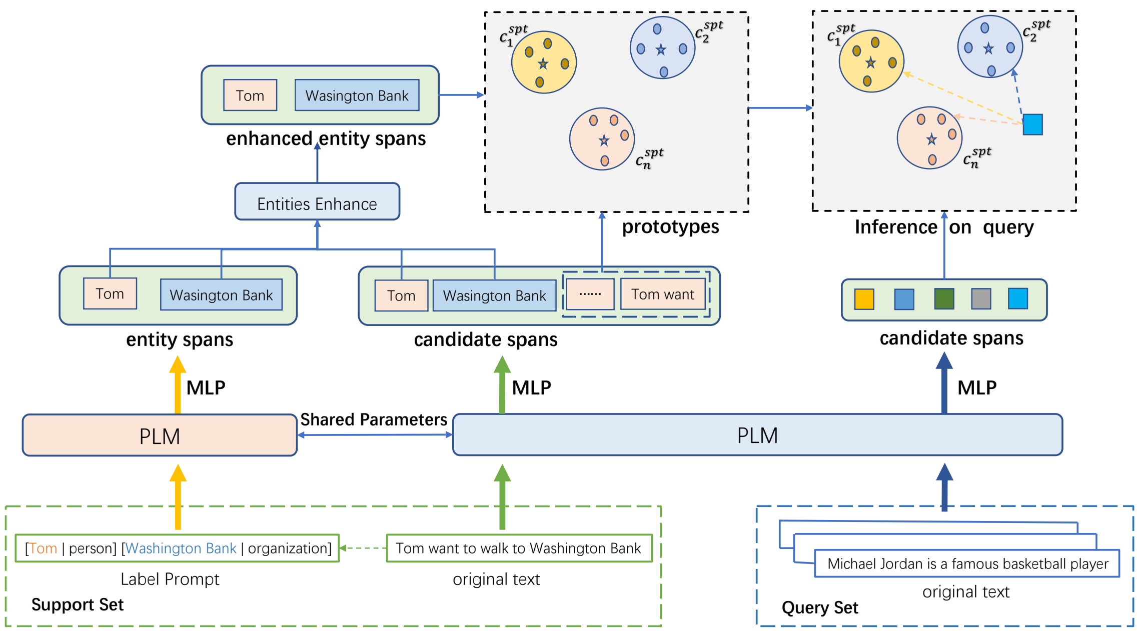


Figure 2: The architecture of our model LPNER. Given an original input from the support set, we first construct label prompts based on the prompt template. We then encode the label prompts and the raw input to obtain the corresponding token sequences. Using a Multi-Layer Perceptron (MLP), we transform the token representations into span representations. Next, we obtain semantically enhanced span representations through weighted summation and construct a prototypical network. Finally, we classify the test samples in the query set based on metric learning.

The architecture of LPNER is illustrated in Figure 2. Firstly, we apply a Pre-trained Language Model (PLM) to obtain semantic representations for the input sequence and each span in the prompt. Then, by applying special mark, we perform weighted fusion on the corresponding spans in the input sequence and prompt to obtain the final fused span representations that incorporate the semantic information of the labels. Next, LPNER computes entity class prototypes in the support set and calculates similarity scores between spans in the query set and prototypes based on distance metrics.

### 4.2. Prompt Construction

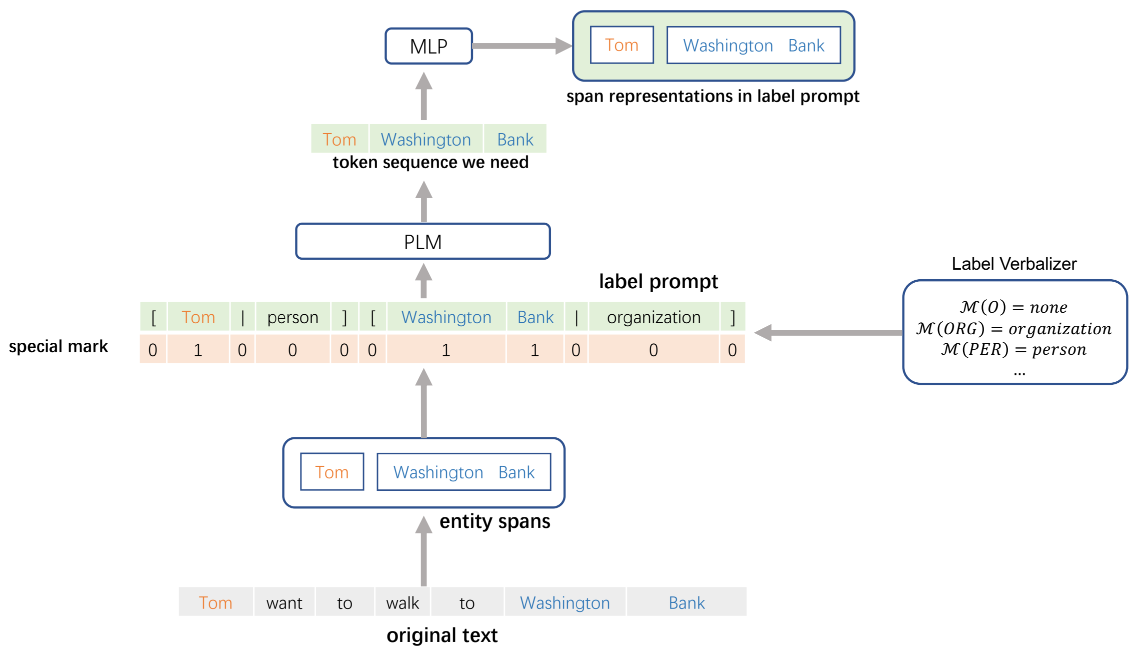


Figure 3: The illustration of prompt construction. First, we construct the label prompt by filling a predefined template with known entities from the training data using a Label Mapper. This Label Prompt is then encoded using a PLM. Subsequently, we use special mark to extract the token representations that we need and input them into an MLP to obtain the entity span representations within the prompt.

As shown in Figure 3, our method first employs a pre-defined prompt template to construct corresponding label prompt for each input sentence. Since the labels in the dataset are often not comprehensible to pre-trained language models, e.g., LOC and ORG, we manually define a class-specific Verbalizer $\mathcal{M}_{ver}$ to map each entity label in the dataset to a unique natural language form of the class-specific word, facilitating the understanding by pre-trained language models. For instance, in the vast majority of NER datasets, "LOC" typically represents the "location" entity. Thus, according to $\mathcal{M}_{ver}(\text{LOC}) = $ "location", the label LOC is mapped to the class-specific word "location". The specific mapping is introduced in Table 8 in the Appendix A. These class-specific words encompass general semantic information about the relevant entity classes.

Next, we can utilize these class-specific words to generate corresponding label prompts $X_{prompt}$ for each input sentence:

$$X_{prompt} = F_{prompt}(X, M, \mathcal{Y}) \tag{1}$$

Specifically, The function $F_{prompt}$ serves as a prompt function utilized to populate the prompt template "[<candidate_entity>|<entity_type>]". As previously described, $X$ represents the input sentence, $M$ and $\mathcal{Y}$ respectively denote the position information of the entity span and its corresponding ground-truth label. For each entity appearing in the input sentence, we retrieve the corresponding label from $\mathcal{Y}$ and map the label to a unique class-specific word to construct the label prompt. For example, given the input sentence $X$ = "Tom was born in 1999", which contains entities "Tom" and "1999", the corresponding label prompt $X_{prompt}$ would be "[Tom|person][1999|date]". In the label prompt, we only need to retrieve part of the token representations rather than all token representations. Hence, following ProML (Chen et al., 2022), we introduce a special mark $m \in [0, 1]^{|X_{prompt}|}$ to filter out those token representations that are not used later through a simple filtering operation where m==0. Additionally, since the label prompt can only be applied when the ground-truth label is available, in a few-shot learning setting, we only apply this prompt to the support set and not to the query set.

### 4.3. Training in Source Domain

During the training process, we adopt the episode training strategy to sample mini-batches from the source domain dataset $\mathcal{D}_{source}$, where each mini-batch contains a few-shot episode data. First, we use a PLM to encode the input sentence $X = \{x_1, \ldots, x_k\}$ and the label prompt $X_{prompt} = \{x'_1, \ldots, x'_l\}$:

$$\mathrm{H} = [h_1, \ldots, h_k] = \mathrm{PLM}([x_1, \ldots, x_k]) \tag{2}$$

$$\mathrm{H}_{prompt} = [h'_1, \ldots, h'_l] = \mathrm{PLM}([x'_1, \ldots, x'_l]) \tag{3}$$

Next, we use mark $m$ to obtain the token representations required in the label prompt:

$$\mathrm{H}'_{prompt} = \mathrm{H}_{prompt}[m == 1] \tag{4}$$

For a span with words $\{w_i\}_{\mathrm{i=s}}^{\mathrm{e}}$, we first concat the start and end token embeddings, and then feed them into MLP to get the span representation:

$$s = MLP(h_s \oplus h_e) \tag{5}$$

Therefore, we can get the corresponding span representations set $S$ and $S_{prompt}$ from H and $\mathrm{H}'_{prompt}$ respectively:

$$S = \left\{v_1, ..., v_{|s|}\right\} \tag{6}$$

$$S_{prompt} = \left\{u_1, ..., u_{|S_{prompt}|}\right\} \tag{7}$$

Where $|S|$ represent the number of all candidate spans in the input sentence and $|S_{prompt}|$ represent the number of entity spans in the label prompt.

Then, in order to obtain the final entity span representations that incorporate label semantics, for each entity span $u_i$ ($i \leq |S_{prompt}|$) in $S_{prompt}$, We weight $u_i$ and the corresponding entity span representation $v_i$ in $S$:

$$u'_i = \alpha * u_i + \beta * v_i \tag{8}$$

Where $u'_i$ represents a entity span that incorporates label semantics, $\alpha$ and $\beta$ are learnable hyper-parameters, which $\alpha + \beta = 1$.

Next, we can compute the prototype $c_t$ for each entity type by averaging the representations of all spans in the support set that share the same entity type $t$:

$$c_t = \frac{1}{|u'_t|} \sum u'_t \tag{9}$$

Where $|u'_t|$ represents the span number of the entity type $t$.

During the training stage, since the labels for spans in the query set are visible, the prototypical learning loss is calculated as:

$$\mathcal{L}_{proto} = \frac{1}{|M_q|} \sum_{s^{qry} \in M_q} - \log p(y_{s^{qry}} | s^{qry}) \tag{10}$$

Where $M_q$ represents the span set in the query set, $s^{qry}$ represents the original span representation in the query set and $p(y_{s^{qry}} | s^{qry})$ is the probability distribution which can be calculated as:

$$p(y_{s^{qry}} | s^{qry}) = softmax\left(-d\left(s^{qry}, c_{y_{s^{qry}}}\right)\right) \tag{11}$$

Where $d(.,.)$ represents a distance function.

### 4.4. Adapting to Target Domain

Since the source domain and target domain belong to different domains, LPNER requires certain domain-transferring capabilities. Therefore, after training the model on the source domain, we make an adaptation to the target domain.

During the domain adaptation procedure, LPNER is fine-tuned with relevant support sets. This fine-tuning procedure is similar to the training procedure. Specifically, we first obtain the enhanced span representations based on the label prompt and further compute the prototypes in the support set. After that, we fine-tune the model by utilizing the prototypical learning loss. Different from using $s^{qry}$ in the training procedure, we fine-tune the model using the original span representations of the input sentence $S = \left\{v_1, ..., v_{|s|}\right\}$ in the support set since the labels of the query set are unknown.

In the inference phase, LPNER obtains prototypes and original span representations in the query set. For each span $s^{qry}$ in the query set, LPNER utilizes nearest neighbor inference to find the nearest prototype in the PLM representation space and assigns the corresponding label to this span:

$$y^{pred}_{s^{qry}} = argmaxp(y | s^{qry}) \tag{12}$$

## 5. Experiments

In this section, we mainly introduce our experiments. We will introduce the experimental datasets, baselines and experimental settings, and discuss the experimental results.

### 5.1. Datasets

To evaluate the performance of LPNER, we use six datasets across different domains: ACE04 (Mitchell et al., 2005), ACE05 (Walker et al., 2006), GENIA (Kim et al., 2003), GermEval (Benikova et al., 2014), NEREL (Loukachevitch et al., 2021), and Few-NERD (Ding et al., 2021). Among these datasets, the first five datasets are nested NER datasets, and the last one is a flat NER dataset. The details of the dataset are shown in Table 1.

Table 1: Statistics of Datasets.

| Dataset | language | Types | Sentences | Entities/Nest entities |
|---|---|---|---|---|
| ACE04 | English | 7 | 6.8k | 27.8k / 12.7k |
| ACE05 | English | 7 | 13.6k | 50.2k / 18.3k |
| GENIA | English | 36 | 18.5k | 55.7k / 30.0k |
| GermEval | German | 12 | 18.4k | 41.1k / 6.1k |
| NEREL | Russian | 29 | 8.9k | 56.1k / 18.7k |
| FewNERD | English | 66 | 188.2k | - |

### 5.2. Baselines

To compare the performance of LPNER, we consider three different types of methods as our baselines:

1) Rich-resource-based nested NER methods: NER-DP (Yu et al., 2020) uses the idea of graph-based dependency parsing and applies a biaffine model to establish the dependency of the start and end words for each span. IoBP (Wang et al., 2022b) is an extension of the second-best path recognition method, which eliminates the impact of the best path. PO-TreeCRFs (Fu et al., 2021) treats nested NER as constituency parsing with partially observed trees.

2) prompt-learning-based few-shot NER methods: COPNER (Huang et al., 2022) is a few-shot flat NER approach, which combines contrastive learning and prompt guiding. ProML (Chen et al., 2022) combines prompt learning and nearest neighbor inference to solve few-shot flat NER. TemplateNER (Cui et al., 2021) proposes a template-based method for NER, treating NER as a language model ranking problem in a sequence-to-sequence framework.

3) contrastive-learning-based and metric-learning-based few-shot NER methods: CONTaiNER (Das et al., 2021) is a contrastive-learning-based few-shot flat NER method. ProtoNet (Snell et al., 2017) utilizes Prototypical Network. SLNER (Ren et al., 2023) utilizes two encoders: one encodes text spans with enhanced span representations using biaffine and self-attention, and the other encodes label names for label representations. NNShot (Yang and Katiyar, 2020) is a few-shot flat NER method that utilizes nearest neighbor inference based metric-learning. ESD (Wang et al., 2021) constructs prototypes by applying intra-span and cross-span attention to enhance span representation. SpanProto (Wang et al., 2022a) applies a two-stage strategy to recognize entities, including a span extractor stage to determine candidate entity spans and a mention classifier stage to identify entity labels.

### 5.3. Training and Inference Settings

During the training procedure, we used the Few-NERD dataset as the source dataset. We randomly sampled 10500 5-way 5-shot subtasks from the Few-NERD inter-domain subset, among which 10000 subtasks as the training set and 500 subtasks as the validation set. And we validate our model for every 1000 subtasks.

During the inference procedure, we also require support set and query set for model fine-tuning and inference. Therefore, we first sample several sentences from the target domain dataset as the support set and the remaining sentence as the query set. During the sampling process, we refer to the idea of $K$-$2K$ sampling strategy and allow some entity categories to have more than $K$ entities. After that, we fine-tuned our model on the support set and then tested it on the query set. In this experiment, we adopted a few-shot setting of 1-shot and 5-shot, sampling support sets from the test subset of the ACE04, ACE05, GENIA, GermEval, and NEREL datasets, respectively.

### 5.4. Implementation Details

We choose $\text{BERT}_{\text{base\_multilingual}}$ from HuggingFace as the default Pre-trained Language Model. And we select Adam as the optimizer to optimize the model. The learning rate of the encoder is 5e-5. $\alpha$ and $\beta$ are learnable hyper-parameters, we initialize them to 0.5 and 0.5 respectively. We set 10 different random seeds from 0 to 9 to get ten results and report the average micro-F1 with standard deviations. We implement our model with PyTorch 1.8.2, and train the model with a single NVIDIA Tesla A10 GPU.

### 5.5. Experimental Results

#### 5.5.1. Main Results

Table 2: $F_1$ performance on ACE04, ACE05, GENIA, GermEval, and NEREL nested NER datasets with 1-shot setting (%).

| Model | ACE04 | ACE05 | GENIA | GermEval | NEREL |
|---|---|---|---|---|---|
| NER-DP | 4.01±2.75 | 6.48±5.34 | 15.26±2.78 | 7.12±2.61 | 15.86±5.77 |
| IoBP | 10.63±6.70 | 15.68±4.48 | 16.09±2.07 | 3.32 ±2.04 | 8.61±1.23 |
| PO-TreeCRFs | 10.55±4.79 | 18.02±11.93 | 22.37±5.08 | 8.87±8.08 | 22.06±6.55 |
| COPNER | 9.19±6.34 | 11.21±9.92 | 7.47±1.70 | 27.81±9.89 | 26.56±3.42 |
| ProML | 22.05±5.01 | 22.74±9.81 | 6.65±1.56 | 23.40±5.61 | 21.93±3.39 |
| TemplateNER | 10.16±5.25e-03 | 16.51±4.83e-03 | 19.01±5.00e-03 | 13.50±9.92e-03 | 7.49±0.02 |
| CONTaiNER | 6.87±2.89 | 11.46±3.30 | 18.47±2.36 | 29.18±7.05 | 26.61±1.75 |
| ProtoNet | 25.55±8.23 | 25.61±11.25 | 19.76±1.73 | 33.20±9.00 | 38.70±4.62 |
| SLNER | 11.14±3.91 | 16.58±9.39 | 12.48±4.32 | 23.82±7.74 | 29.76±4.57 |
| NNShot | 22.01±7.92 | 23.93±10.74 | 24.25±2.89 | 28.58±6.76 | 38.58±1.30 |
| ESD | 23.41±6.19 | 24.85±11.17 | 21.73±3.64 | 34.00±8.75 | 28.56±5.18 |
| SpanProto | 24.90±5.80 | **29.92**±8.27 | **29.01**±3.55 | 34.12±6.64 | 44.20±3.55 |
| LPNER | **25.67**±7.05 | 25.01±10.83 | 26.32±3.88 | **39.45**±6.55 | **45.11**±3.78 |

Table 3: $F_1$ performance on ACE04, ACE05, GENIA, GermEval, and NEREL nested NER datasets with 5-shot setting (%).

| Model | ACE04 | ACE05 | GENIA | GermEval | NEREL |
|---|---|---|---|---|---|
| NER-DP | 11.48±4.05 | 15.58±8.54 | 31.89±4.01 | 24.89±3.92 | 42.25±2.42 |
| IoBP | 14.14±6.06 | 34.36±6.62 | 31.67±3.31 | 12.86±2.60 | 18.50±1.46 |
| PO-TreeCRFs | 29.77±7.97 | 33.83±10.54 | 35.13±3.33 | 45.83±3.88 | 52.25±2.40 |
| COPNER | 17.25±11.56 | 26.21±11.05 | 16.67±2.62 | 32.40±9.87 | 37.46±4.15 |
| ProML | 26.50±6.46 | 38.44±9.16 | 10.31±0.93 | 28.45±7.10 | 30.23±2.18 |
| TemplateNER | 14.46±0.02 | 19.16±9.29e-03 | 20.86±0.02 | 18.31±0.03 | 10.49±0.06 |
| CONTaiNER | 14.19±3.09 | 15.52±4.96 | 19.90±1.21 | 37.05±1.01 | 44.37±1.27 |
| ProtoNet | 40.18±6.19 | 41.52±5.14 | 38.01±2.75 | 47.95±4.06 | 50.22±1.28 |
| SLNER | 23.40±2.45 | 34.72±2.76 | 27.01±2.50 | 25.84±3.72 | 39.81±0.47 |
| NNShot | 37.74±5.55 | 36.69±6.23 | 35.57±2.43 | 41.26±2.50 | 46.54±1.93 |
| ESD | 39.13±5.09 | 41.30±5.37 | 27.54±3.17 | 34.75±6.03 | 47.68±2.20 |
| SpanProto | 40.10±5.98 | 41.65±7.89 | 41.84±2.66 | 51.11±5.89 | 56.16±2.15 |
| LPNER | **42.67**±7.55 | **46.62**±5.82 | **44.99**±2.20 | **59.30**±2.26 | **61.54**±1.77 |

Table 2 and Table 3 shows the average results over ten experiments of our method compared with these baselines introduced in 5.2 under the 1-shot and 5-shot settings, respectively. We can observe that our method has improved compared to most of these baselines.

In the 1-shot setting, LPNER performs well on some datasets. For the ACE04 dataset, it surpasses all baselines with a modest improvement of 0.12%, but it underperforms on ACE05, likely due to limited labeled data restricting its ability to capture deeper semantics. On the GENIA dataset, LPNER falls short of SpanProto, potentially due to the complexity and similarity of biomedical labels, which are harder to differentiate with minimal data. However, LPNER performs strongly on GermEval and NEREL, achieving $F_1$ scores of 39.45% and 45.11%, with improvements of 5.33% and 0.91%, respectively.

In the 5-shot setting, LPNER consistently excels across datasets. It outperforms all baselines on ACE04 and ACE05 with improvements of 2.49% and 4.97%, respectively. For the GENIA dataset, LPNER achieves state-of-the-art results with a 3.15% improvement over SpanProto. It also performs exceptionally well on GermEval and NEREL, with $F_1$ scores of 59.30% and 61.54%, marking improvements of 8.19% and 5.38%, respectively.

From the experimental results, it can be observed that LPNER demonstrates more significant performance improvements in the 5-shot setting. This is likely because, in the 1-shot setting, the extremely limited labeled data may prevent LPNER from fully capturing label semantics, hindering its ability to achieve optimal performance on certain datasets. In contrast, the increased number of labeled examples in the 5-shot setting enables LPNER to better leverage semantic information, leading to superior results.

#### 5.5.2. Comparison of LPNER and ChatGPT

In order to compare the performance of LPNER with that of Large Language Models (LLM), we cited the experimental results of ChatGPT in Han et al. (2023), and the results are shown in Table 4. For the ACE04 and ACE05 datasets, LPNER improved by 4.15% and 10.45% respectively compared with ChatGPT. For the GENIA dataset, ChatGPT performed better, 3.83% higher than LPNER. We think that this is due to the model characteristics of ChatGPT itself and the selection of some optimal prompt sample examples, which significantly improves the model performance and makes ChatGPT more adaptable on the GENIA dataset.

Table 4: Performance comparison between ChatGPT and LPNER on different datasets on 5-shot setting.

| Model | ACE04 | ACE05 | GENIA |
|---|---|---|---|
| ChatGPT | 38.52±2.51 | 36.17±1.78 | **48.82**±1.31 |
| LPNER | **42.67**±7.55 | **46.62**±5.82 | 44.99±2.20 |

### 5.5.3. Analysis of Prompt Template

Table 5: Statistics of different prompt templates.

| templateID | template |
|---|---|
| template1 | [<candidate_entity> is a <entity_type> entity] |
| template2 | [<candidate_entity> belongs to <entity_type> category] |
| template3 | [the entity type of <candidate_entity> is <entity_type>] |
| template4 | [<candidate_entity> is marked as <entity_type>] |

Table 6: Results of different prompt templates in 1-shot and 5-shot settings on the GermEval dataset.

| dataset | template1 | template2 | template3 | template4 |
|---|---|---|---|---|
| GermEval(1-shot) | 40.71±6.65 | 40.39±8.89 | 39.32±9.62 | 38.07±8.37 |
| GermEval(5-shot) | 59.20±2.25 | 58.77±3.13 | 60.66±2.83 | 58.79±2.74 |

To analyze the impact of different prompt templates on model performance, we additionally designed four prompt templates as shown in Table 5. We then conducted experiments using these templates on the GermEval dataset under 1-shot and 5-shot settings. Tabel 6 illustrates the effect of various templates on the model's $F_1$ score. As can be seen from the Tabel 6, when the prompt template changes, the performance of the model will change slightly, but it is generally stable, indicating that our method is robust to prompts to a certain extent.

### 5.5.4. Ablation Study

As shown in Table 7, in order to analyze the effectiveness of Label Prompt (LP) in our LPNER, we conduct an ablation study on the ACE04, ACE05, GENIA, GermEval and NEREL 1-shot setting. According to Table 7, the results suggest that the LP can effectively improve the $F_1$ score. Specifically, Label Prompt achieved $F_1$ score improvements of 1.20%, 0.88%, 3.24%, 4.96%, and 2.86% on the ACE04, ACE05, GENIA, GermEval, and NEREL datasets. The results of the above ablation experiments show that Label Prompt is of positive impact to the improvement of performance.

Table 7: Ablation study of $F_1$ performance on different datasets 1-shot setting (%). "w/o LP" means removing the Label Prompt.

| | ACE04 | ACE05 | GENIA | GermEval | NEREL |
|---|---|---|---|---|---|
| LPNER | 25.67±7.05 | 25.01±10.83 | 26.32±3.88 | 39.45±6.55 | 45.11±3.78 |
| w/o LP | 24.47±7.76 | 24.13 ±10.73 | 23.08±5.44 | 34.49±7.62 | 42.25±3.20 |

## 6. Conclusion

In this paper, we propose LPNER, a novel Label-Prompt-based approach for nested named entity recognition, specifically designed for few-shot learning scenarios. By leveraging prompt templates to integrate label semantics and text information, LPNER consistently outperforms baseline methods across multiple datasets, including ACE04, ACE05, GENIA, GermEval, and NEREL, in both 1-shot and 5-shot settings. While effective, our approach has some limitations, particularly in the manual design of prompts and the need for label-to-class-specific-word mappings, which may impact performance. Addressing these limitations could further enhance the model's adaptability and effectiveness.

Futuremore, the label prompt method proposed in this paper can also be applied to other fields, such as static, dynamic, and multimodal knowledge graphs reasoning (Liang et al., 2024). For static graphs, label prompts can enhance the model's ability to accurately identify and label entities and relationships. In dynamic graphs, including temporal knowledge graphs (Wang et al., 2023), time-sensitive prompts can capture temporal variations in data, improving the accuracy of temporal knowledge graph completion. Additionally, in multimodal graphs, label prompts can provide context across different data modalities, further enhancing the model's reasoning capabilities.

## 7. Acknowledgements

This work was partially supported by the National Natural Science Foundation of China (No. 62072153), the Anhui Provincial Key Technologies R&D Program (No. 2022h11020015) and the 111 center (No. B14025).

## References

Darina Benikova, Chris Biemann, and Marc Reznicek. Nosta-d named entity annotation for german: Guidelines and dataset. In *LREC*, pages 2524–2531, 2014.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Yanru Chen, Yanan Zheng, and Zhilin Yang. Prompt-based metric learning for few-shot ner. *arXiv preprint arXiv:2211.04337*, 2022.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. Template-based named entity recognition using bart. *arXiv preprint arXiv:2106.01760*, 2021.

Sarkar Snigdha Sarathi Das, Arzoo Katiyar, Rebecca J Passonneau, and Rui Zhang. Container: Few-shot named entity recognition via contrastive learning. *arXiv preprint arXiv:2109.07589*, 2021.

Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Hai-Tao Zheng, and Zhiyuan Liu. Few-nerd: A few-shot named entity recognition dataset. *arXiv preprint arXiv:2105.07464*, 2021.

Jenny Rose Finkel and Christopher D Manning. Nested named entity recognition. In *Proceedings of the 2009 conference on empirical methods in natural language processing*, pages 141–150, 2009.

Yao Fu, Chuanqi Tan, Mosha Chen, Songfang Huang, and Fei Huang. Nested named entity recognition with partially-observed treecrfs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12839–12847, 2021.

Ridong Han, Tao Peng, Chaohao Yang, Benyou Wang, Lu Liu, and Xiang Wan. Is information extraction solved by chatgpt? an analysis of performance, evaluation criteria, robustness and errors. *arXiv preprint arXiv:2305.14450*, 2023.

Yutai Hou, Cheng Chen, Xianzhen Luo, Bohan Li, and Wanxiang Che. Inverse is better! fast and accurate prompt for few-shot slot tagging. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 637–647, 2022.

Jin Huang, Danfeng Yan, and Yuanqiang Cai. Pmrc: Prompt-based machine reading comprehension for few-shot named entity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18316–18326, 2024.

Yucheng Huang, Kai He, Yige Wang, Xianli Zhang, Tieliang Gong, Rui Mao, and Chen Li. Copner: Contrastive learning with prompt guiding for few-shot named entity recognition. In *Proceedings of the 29th International conference on computational linguistics*, pages 2515–2527, 2022.

Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182, 2003.

Brian Lester, Rami Al-Rfou, and Noah Constant. 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, 2021.

Jing Li, Billy Chiu, Shanshan Feng, and Hao Wang. Few-shot named entity recognition via meta-learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(9):4245–4256, 2020.

Ke Liang, Lingyuan Meng, Meng Liu, Yue Liu, Wenxuan Tu, Siwei Wang, Sihang Zhou, Xinwang Liu, Fuchun Sun, and Kunlun He. A survey of knowledge graph reasoning on graph types: Static, dynamic, and multi-modal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Andy T Liu, Wei Xiao, Henghui Zhu, Dejiao Zhang, Shang-Wen Li, and Andrew Arnold. Qaner: Prompting question answering models for few-shot named entity recognition. *arXiv preprint arXiv:2203.01543*, 2022.

Natalia Loukachevitch, Ekaterina Artemova, Tatiana Batura, Pavel Braslavski, Ilia Denisov, Vladimir Ivanov, Suresh Manandhar, Alexander Pugachev, and Elena Tutubalina. Nerel: A russian dataset with nested named entities, relations and events. *arXiv preprint arXiv:2108.13112*, 2021.

Alexis Mitchell, Stephanie Strassel, Shudong Huang, and Ramez Zakhary. Ace 2004 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 1:1–1, 2005.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? 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 2463–2473, 2019.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International conference on learning representations*, 2016.

Zhe Ren, Xizhong Qin, and Wensheng Ran. Slner: Chinese few-shot named entity recognition with enhanced span and label semantics. *Applied Sciences*, 13(15):8609, 2023.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 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, 2020.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57:45, 2006.

Jianing Wang, Chengyu Wang, Chuanqi Tan, Minghui Qiu, Songfang Huang, Jun Huang, and Ming Gao. Spanproto: A two-stage span-based prototypical network for few-shot named entity recognition. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3466–3476, 2022a.

Jiapu Wang, Boyue Wang, Meikang Qiu, Shirui Pan, Bo Xiong, Heng Liu, Linhao Luo, Tengfei Liu, Yongli Hu, Baocai Yin, et al. A survey on temporal knowledge graph completion: Taxonomy, progress, and prospects. *arXiv preprint arXiv:2308.02457*, 2023.

Peiyi Wang, Runxin Xu, Tianyu Liu, Qingyu Zhou, Yunbo Cao, Baobao Chang, and Zhifang Sui. An enhanced span-based decomposition method for few-shot sequence labeling. *arXiv preprint arXiv:2109.13023*, 2021.

Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

Yiran Wang, Hiroyuki Shindo, Yuji Matsumoto, and Taro Watanabe. Nested named entity recognition via explicitly excluding the influence of the best path. *Journal of Natural Language Processing*, 29(1):23–52, 2022b.

Yi Yang and Arzoo Katiyar. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. *arXiv preprint arXiv:2010.02405*, 2020.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. Named entity recognition as dependency parsing. *arXiv preprint arXiv:2005.07150*, 2020.

Mozhi Zhang, Hang Yan, Yaqian Zhou, and Xipeng Qiu. Promptner: A prompting method for few-shot named entity recognition via k nearest neighbor search. *arXiv preprint arXiv:2305.12217*, 2023.

## Appendix A. Label Mapping

Table 8: Original labels and their corresponding class-specific words in the datasets.

| Dataset | original label | class-specific word |
|---|---|---|
| Few-NERD | O | none |
| | organization-company | company |
| | ... | ... |
| GermEval | OTH | other |
| | PER | person |
| | ... | ... |
| GENIA | other_name | other |
| | virus | virus |
| | ... | ... |
| NEREL | NATIONALITY | nationality |
| | ... | ... |
| ACE | ORG | organization |
| | ... | ... |