

# SpanProto: A Two-stage Span-based Prototypical Network for Few-shot Named Entity Recognition

Jianing Wang<sup>1</sup>, Chengyu Wang<sup>2</sup>, Chuanqi Tan<sup>2</sup>, Minghui Qiu<sup>2</sup>,  
Songfang Huang<sup>2</sup>, Jun Huang<sup>2</sup>, Ming Gao<sup>1,3\*</sup>

<sup>1</sup> School of Data Science and Engineering, East China Normal University, Shanghai, China

<sup>2</sup> Alibaba Group, Hangzhou, China

<sup>3</sup> KLASDS-MOE, School of Statistics, East China Normal University, Shanghai, China

lygwjn@gmail.com,

{chengyu.wcy, chuanqi.tcq, minghui.qmh}@alibaba-inc.com,

{songfang.hsf, huangjun.hj}@alibaba-inc.com,

mgao@dase.ecnu.edu.cn

## Abstract

Few-shot Named Entity Recognition (NER) aims to identify named entities with very little annotated data. Previous methods solve this problem based on token-wise classification, which ignores the information of entity boundaries, and inevitably the performance is affected by the massive non-entity tokens. To this end, we propose a seminal span-based prototypical network (SpanProto) that tackles few-shot NER via a *two-stage* approach, including span extraction and mention classification. In the span extraction stage, we transform the sequential tags into a global boundary matrix, enabling the model to focus on the explicit boundary information. For mention classification, we leverage prototypical learning to capture the semantic representations for each labeled span and make the model better adapt to novel-class entities. To further improve the model performance, we split out the false positives generated by the span extractor but not labeled in the current episode set, and then present a margin-based loss to separate them from each prototype region. Experiments over multiple benchmarks demonstrate that our model outperforms strong baselines by a large margin. <sup>1</sup>

## 1 Introduction

Named Entity Recognition (NER) is one of the crucial tasks in natural language processing (NLP), which aims at extracting mention spans and classifying them into a set of pre-defined entity type classes. Previous methods (Huang et al., 2015; Santoro et al., 2016; Ma and Hovy, 2016; Lample et al., 2016; Peters et al., 2017) present multiple

\* Corresponding author.

<sup>1</sup>All the codes and datasets will be released to the EasyNLP framework (Wang et al., 2022a). URL: <https://github.com/alibaba/EasyNLP>

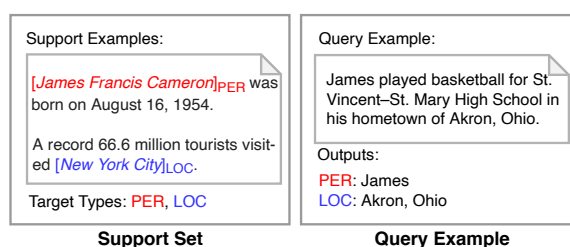


Figure 1: An example of the 2-way 1-shot NER problem. Given a support set with 2 target types where each type is associated with 1 labeled entity, the task is to identify entities in the query example.

deep neural architectures and achieve impressive performance. Yet, these conventional approaches heavily depend on the time-consuming and labor-intensive process of data annotation. Thus, an attractive problem of few-shot NER (Ding et al., 2021; Huang et al., 2021; Ma et al., 2022b) has been introduced which involves recognizing novel-class entities based on very few labeled examples (i.e. support examples). An example of the 2-way 1-shot NER problem is shown in Figure 1.

To solve the problem, multiple methods (Das et al., 2022; Hou et al., 2020; Ziyadi et al., 2020; Fritzier et al., 2019; Ma et al., 2022a) follow the sequence labeling strategy that directly categorizes the entity type at the token level by calculating the distance between each query token and the prototype of each entity type class or the support tokens. However, the effect of these approaches are disturbed by numerous non-entity tokens (i.e. “O” class) and the token-wise label dependency (i.e. “BIO” rules) (Wang et al., 2021; Shen et al., 2021). To bypass these issues, a branch of *two-stage* methods arise to decompose NER into two separate processes (Shen et al., 2021; Wang et al., 2021;

Ma et al., 2022b; Wu et al., 2022), including span extraction and mention classification. Specifically, they first extract multiple spans via a class-agnostic model and then assign the label for each predicted span based on metric learning. Despite the success, there could still be two remaining problems. 1) The performance of span extraction is the upper limit for the whole system, which is still far away from satisfaction. 2) Previous methods ignore false positives generated during span extraction. Intuitively, because the decomposed model is class-agnostic in the span extraction stage, it could generate some entities which have no available entity type to be assigned in the target type set. In Figure 1, the model could extract a span of “August 16, 1954” (may be an entity of time type in another episode), yet, the existing methods still assign it a label as “PER” or “LOC”<sup>2</sup>.

To address these limitations, we present a novel **Span-based Prototypical Network** (SpanProto) via a *two-stage* approach. For span extraction, we introduce a *Span Extractor*, which aims to find the candidate spans. Different from the recent work (Ma et al., 2022b) which models it as a sequence labeling task, we convert the sequential tags to a global boundary matrix, which denotes the sentence-level target label, enabling the model to learn the explicit span boundary information regardless of the token-wise label dependency. For mention classification, we propose a *Mention Classifier* which aims at assigning a pre-defined entity type for each recalled span. When training the mention classifier, we compute the prototype embeddings for each entity type class based on the support examples, and leverage the prototypical learning to adjust the span representations in the semantic space. To address the problem of false positives, we additionally design a margin-based loss to enlarge the semantic distance between false positives and all prototypes. We conduct extensive experiments over multiple benchmarks, including FewNERD (Ding et al., 2021) and CrossNER (Hou et al., 2020). Results show that our method consistently outperforms state-of-the-art baselines by a large margin. We summarize our main contributions as follows:

- We propose a novel *two-stage* framework named SpanProto that solves the problem

<sup>2</sup>Previous works suppose that all spans generated by the span extractor can be assigned with a type, which is unrealistic in the two-stage scenario.

of few-shot NER with two mainly modules, i.e. *Span Extractor*, and *Mention Classifier*.

- In SpanProto, we introduce a global boundary matrix to learn the explicit span boundary information. Additionally, we effectively train the model with prototypical learning and margin-based learning to improve the abilities of generalization and adaptation on novel-class entities.
- Extensive experiments conducted over two widely-used benchmarks illustrate that our method achieves the best performance.

## 2 Related Work

In this section, we briefly summarize the related work in various aspects.

**Few-shot Learning and Meta Learning.** Few-shot learning is a challenging problem which aims to learn models that can quickly adapt to different tasks with low-resource labeled data (Wang et al., 2020; Huisman et al., 2021). A series of typical meta-learning algorithms for few-shot learning consist of optimization-based learning (Kulkarni et al., 2016), metric-based learning (Snell et al., 2017), and augmentation-based learning (Wei and Zou, 2019), etc. Recently, multiple approaches have been applied in few-shot NLP tasks, such as text classification (Geng et al., 2020), question answering (Wang et al., 2022b) and knowledge base completion (Sheng et al., 2020). Our SpanProto is a typical  $N$ -way  $K$ -shot paradigm, which is based on meta learning to make the model better adapt to new domains with little training data available.

**Few-shot Named Entity Recognition.** Few-shot NER aims to identify and classify the entity type based on low-resource data. Recently, Ding et al. (2021) and Hou et al. (2020) provide well-designed few-shot NER benchmarks in a unified  $N$ -way  $K$ -shot paradigm. A series of approaches typically adopt the metric-based learning method to learn the representations of the entities in the semantic space, i.e. prototypical learning (Snell et al., 2017), margin-based learning (Levi et al., 2021) and contrastive learning (Gao et al., 2021). Existing approaches can be divided into two kinds, i.e., one-stage (Snell et al., 2017; Hou et al., 2020; Das et al., 2022; Ziyadi et al., 2020) and two-stage (Ma et al., 2022b; Wu et al., 2022; Shen et al., 2021). Generally, the methods in the kind of one-stage typically categorize the entity type by the token-level metric

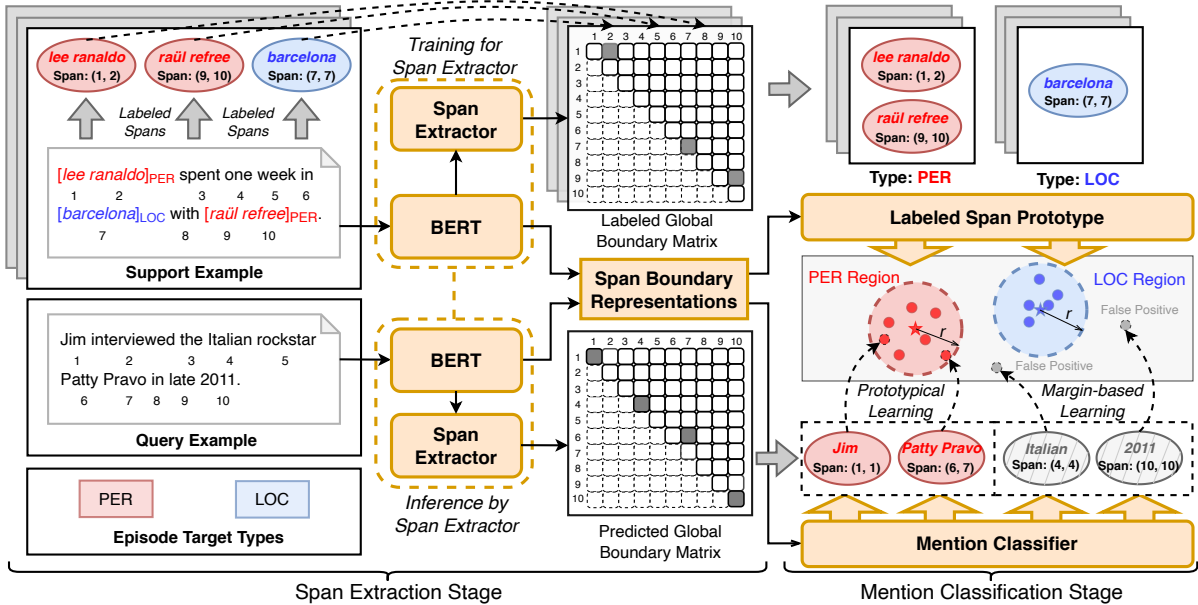


Figure 2: The model architecture of SpanProto. Given one episode data, we first transform the sequential tags into labeled spans and obtain the corresponding global boundary matrix. We train the span extractor over support examples and predict the spans for query examples. In the classification stage, we introduce prototypical learning and margin-based learning to teach the model to learn better semantic representations. (Best viewed in color.)

learning. In contrast, two-stage mainly focuses on two training stages consist of entity span extraction and mention type classification. (Ma et al., 2022b) is a related work of our paper, which utilizes model-agnostic meta-learning (MAML) (Finn et al., 2017) to improve the adaptation ability of the two-stage model. Different from them, we aim to 1) further improve the performance on detecting and extracting the candidate entity spans, and 2) alleviate the bottleneck of false positives in the two-stage few-shot NER system.

### 3 Our Proposal: SpanProto

We formally present the notations and the techniques of our proposed SpanProto. The model architecture is shown in Figure 2.

#### 3.1 Notations

Different from token-wise classification (Ding et al., 2021; Hou et al., 2020), we define the span-based  $N$ -way  $K$ -shot setting for few-shot NER. Suppose that we have a training set  $\mathcal{D}_{train}$  and an evaluation set  $\mathcal{D}_{eval}$ . Given one training episode data  $\mathcal{E}_{train} = (\mathcal{S}_{train}, \mathcal{Q}_{train}, \mathcal{T}_{train}) \in \mathcal{D}_{train}$ , where  $\mathcal{S}_{train}$  and  $\mathcal{Q}_{train}$  denote the support set and the query set, respectively.  $\mathcal{T}_{train}$  denotes the entity type set, and  $|\mathcal{T}_{train}| = N$ . For each example  $(X, \mathcal{M}, \mathcal{Y}) \in \mathcal{S}_{train} \cup \mathcal{Q}_{train}$ ,  $X = \{x_i\}_{i=1}^L$  denotes the input sentence with  $L$  language tokens.

$\mathcal{M} = \{(s_j, e_j)\}_{j=1}^{L'}$  represents the mention span set for the sentence  $X$ , where  $s_j, e_j$  denote the start and end position in the sentence for the  $j$ -th span, and  $0 \leq s_j \leq e_j \leq L$ .  $L'$  is the number of spans.  $\mathcal{Y} = \{y_j\}_{j=1}^{L'}$  is the entity type set and  $y_j \in \mathcal{T}_{train}$  is the label for the  $j$ -th span  $(s_j, e_j)$ .

#### 3.2 Span Extractor

The span extractor aims to generate all candidate entity spans. Specifically, given one training episode data  $\mathcal{E}_{train} = (\mathcal{S}_{train}, \mathcal{Q}_{train}, \mathcal{T}_{train})$  from  $\mathcal{D}_{train}$ , we use the support example  $(X_s, \mathcal{M}_s, \mathcal{Y}_s) \in \mathcal{S}_{train}$  to train the span extractor, where  $X_s = \{x_i\}_{i=1}^L$  denotes the input sentence with  $L$  tokens,  $\mathcal{M}_s$  and  $\mathcal{Y}_s$  are the labeled spans and labeled entity types for  $X_s$ , respectively.

We first obtain the contextual embeddings by  $\mathbf{H} = \mathcal{F}(X_s)$ , where  $\mathcal{F}(\cdot)$  denotes the encoder (e.g. BERT (Devlin et al., 2019)),  $\mathbf{H} \in \mathbb{R}^{L \times h}$  is the output representation at the last layer of the encoder,  $h$  denotes the embedding size. For each token  $x_i \in X_s$ , the token embedding is denoted as  $\mathbf{h}_i \in \mathbb{R}^h$ .

Rather than detecting the entity mention based on token-wise sequence labeling (Ma et al., 2022b), we expect that the span extractor captures the explicit boundary information regardless of the token-wise label dependency. We follow the idea of self-attention (Vaswani et al., 2017) which can be used to calculate the attentive correlation for a token pair

$(x_i, x_j)$  without any dependencies. Specifically, we first compute the *query* and *key* item for each token  $x_i$ . Formally, we have:

$$\mathbf{q}_i = W_q \mathbf{h}_i + b_q, \mathbf{k}_i = W_k \mathbf{h}_i + b_k \quad (1)$$

where  $\mathbf{q}_i, \mathbf{k}_i \in \mathbb{R}^h$  denote the *query* and *key* embeddings, respectively.  $W_q, W_k \in \mathbb{R}^{h \times h}$  are the trainable weights and  $b_q, b_k \in \mathbb{R}^h$  denote the bias. Similar to the biaffine decoder (Yu et al., 2020), we design a score function  $f(i, j)$  to evaluate the probability of the token pair  $(x_i, x_j)$  being an entity span boundary:

$$f(i, j) = \mathbf{q}_i^T \mathbf{k}_j + W_v (\mathbf{h}_i + \mathbf{h}_j) \quad (2)$$

where  $W_v \in \mathbb{R}^{h \times h}$  denotes the trainable weight. Based on the label spans, we then generate a labeled global boundary matrix for each support sentence:

$$\Omega_{i,j} = \begin{cases} 1 & i \leq j \wedge (i, j) \in \mathcal{M}_s; \\ 0 & i \leq j \wedge (i, j) \notin \mathcal{M}_s; \\ -\text{inf} & i > j; \end{cases} \quad (3)$$

where  $\Omega_{i,j}$  is the score of the span  $(i, j)$ . To explicitly learn the span boundary, inspired by Su (2021)<sup>3</sup>, we design a span-based cross-entropy loss to urge the model to learn the boundary information on each training support example:

$$\mathcal{L}_{span} = \log \left( 1 + \sum_{1 \leq i \leq j \leq L} \exp \left( (-1)^{\Omega_{i,j}} f(i, j) \right) \right) \quad (4)$$

For each query example  $(X_q, \mathcal{M}_q, \mathcal{Y}_q)$  in  $\mathcal{Q}_{train}$ , we can obtain the predicted global boundary matrix  $\Omega'$  generated by the span extractor. We follow (Yu et al., 2020) to recall the candidate spans which have a higher score than the pre-defined confidence threshold  $\theta$ . We denote  $\hat{\mathcal{M}}_q = \{(s_j, e_j) | \Omega'_{s_j, e_j} \geq \theta\}$  as the predict result.

### 3.3 Mention Classifier

In the mention classification stage, we introduce a mention classifier to assign the pre-defined entity type for each generated span in the query example. We leverage prototypical learning to teach the model to learn the semantic representations for each labeled span and enable it to adapt to a new-class domain. To remedy the problem of false positives, we further propose a margin-based learning objective to improve the model performance.

<sup>3</sup><https://spaces.ac.cn/archives/8373>.

#### 3.3.1 Prototypical Learning

Specifically, given one episode data  $\mathcal{E}_{train} = (\mathcal{S}_{train}, \mathcal{Q}_{train}, \mathcal{T}_{train})$ , we can compute the prototype  $\mathbf{c}_t \in \mathbb{R}^h$  for each class by averaging the representations of all spans in  $\mathcal{S}_{train}$  which share the same entity type  $t \in \mathcal{T}_{train}$ . Formally, we have:

$$\mathbf{c}_t = \frac{1}{K_t} \sum_{(X_s, \mathcal{M}_s, \mathcal{Y}_s) \in \mathcal{S}_{train}} \sum_{\substack{(s_j, e_j) \in \mathcal{M}_s \\ y_j \in \mathcal{Y}_s}} \mathbb{I}(y_j = t) \mathbf{u}_j \quad (5)$$

where

$$K_t = \sum_{(X_s, \mathcal{M}_s, \mathcal{Y}_s) \in \mathcal{S}_{train}} \sum_{\substack{(s_j, e_j) \in \mathcal{M}_s \\ y_j \in \mathcal{Y}_s}} \mathbb{I}(y_j = t) \quad (6)$$

represents the span number of the entity type  $t$ .  $\mathbf{u}_j \in \mathbb{R}^h$  denotes the span boundary representations, and  $\mathbf{u}_j = \mathbf{h}_{s_j} + \mathbf{h}_{e_j}$ .  $\mathbb{I}(\cdot)$  is the indicator function.

Note that, the labeled span set  $\mathcal{M}_q$  and corresponding type set  $\mathcal{Y}_q$  for each query sentence  $X_q$  in  $\mathcal{Q}_{train}$  is visible during the training stage. Hence, the prototypical loss for each sentence can be calculated as:

$$\mathcal{L}_{proto} = \frac{1}{|\mathcal{M}_q|} \sum_{\substack{(s_j, e_j) \in \mathcal{M}_q \\ y_j \in \mathcal{Y}_q}} -\log p(y_j | s_j, e_j) \quad (7)$$

where

$$p(y_j | s_j, e_j) = \text{SoftMax}(-d(\mathbf{u}_j, \mathbf{c}_{y_j})) \quad (8)$$

is the probability distribution with a distance function  $d(\cdot, \cdot)$ .

#### 3.3.2 Margin-based Learning

Recall that the span set  $\hat{\mathcal{M}}_q$  is extracted by the span extractor for the query sentence  $X_q \in \mathcal{Q}_{train}$ . We find that some mention spans in  $\hat{\mathcal{M}}_q$  do not exist in the ground truth  $\mathcal{M}_q$  which can be viewed as false positives. Formally, We denote  $\mathcal{M}_q^- = \{(s_j, e_j) | (s_j, e_j) \in \hat{\mathcal{M}}_q \wedge (s_j, e_j) \notin \mathcal{M}_q\}$  as the set of these false positives. Take Figure 2 as an example, the span extractor generates four candidate mentions for the query example, i.e., ‘‘Jim’’, ‘‘Patty Pravo’’, ‘‘Italian’’ and ‘‘2011’’, where ‘‘Italian’’ and ‘‘2011’’ are the false positives in this case.

Intuitively, the false positive can be viewed as a special entity mention, which has no type to be assigned in  $\mathcal{T}_{train}$ , but could be an entity in other episode data. In other words, the real type of this

---

**Algorithm 1** Training Procedure of SpanProto

---

**Require:** Training data  $\mathcal{D}_{train}$ , Training total step  $T$ , Pre-training step of the span extractor  $T' < T$ .

- 1: **for** each step  $st \in \{1, 2, \dots, T\}$  **do**
- 2:    $\lambda = 0$  if  $st < T'$  else  $\lambda = 1$ ;
- 3:   Sample  $(\mathcal{S}_{train}, \mathcal{Q}_{train}, \mathcal{T}_{train})$  from  $\mathcal{D}_{train}$ ;
- 4:    $\mathcal{L}_{span\_eps} = \mathcal{L}_{proto\_eps} = \mathcal{L}_{mrg\_eps} = 0$ .
- 5:   **for**  $(X_s, \mathcal{M}_s, \mathcal{Y}_s) \in \mathcal{S}_{train}$  **do**
- 6:     Compute  $\mathcal{L}_{span}$  for the span extractor in Eq. 4;
- 7:      $\mathcal{L}_{span\_eps} = \mathcal{L}_{span\_eps} + \mathcal{L}_{span}$ ;
- 8:   **end for**
- 9:   Get the prototype for each type  $t \in \mathcal{T}_{train}$  in Eq. 5;
- 10:   **for**  $(X_q, \mathcal{M}_q, \mathcal{Y}_q) \in \mathcal{Q}_{train}$  **do**
- 11:     Based on  $\mathcal{M}_q$  and  $\mathcal{Y}_q$ , calculate prototypical learning loss  $\mathcal{L}_{proto}$  in Eq. 7;
- 12:     Obtain the false positives set  $\mathcal{M}_q^-$ , and calculate margin-based learning loss  $\mathcal{L}_{mrg}$  in Eq. 9;
- 13:      $\mathcal{L}_{proto\_eps} = \mathcal{L}_{proto\_eps} + \mathcal{L}_{proto}$ ;
- 14:      $\mathcal{L}_{mrg\_eps} = \mathcal{L}_{mrg\_eps} + \mathcal{L}_{mrg}$ ;
- 15:   **end for**
- 16:   Obtain the overall loss by  $\mathcal{L} = \frac{1}{|\mathcal{S}_{train}|} \mathcal{L}_{span\_eps} + \frac{\lambda}{|\mathcal{Q}_{train}|} (\mathcal{L}_{proto\_eps} + \mathcal{L}_{mrg\_eps})$ ;
- 17:   Update the model by backpropagation to reduce  $\mathcal{L}$ ;
- 18: **end for**
- 19: **return** The trained SpanProto model.

---

false positive is unknown. Thus, a natural idea is that we can keep it away from all current prototypes in the semantic space. Specifically, we have:

$$\mathcal{L}_{mrg} = \frac{1}{|\mathcal{T}_{train}| |\mathcal{M}_q^-|} \sum_{t \in \mathcal{T}_{train}} \sum_{(s_j, e_j) \in \mathcal{M}_q^-} \max(0, r - d(\mathbf{u}_j^-, \mathbf{c}_t)) \quad (9)$$

where  $\mathbf{u}_j^- \in \mathbb{R}^h$  denotes the span boundary representations of the false positive span  $(s_j, e_j) \in \mathcal{M}_q^-$ . Specially, we let  $\mathcal{L}_{mrg}$  be 0 if  $\mathcal{M}_q^- = \emptyset$ .  $r > 0$  is the pre-defined margin value. Under the margin-based learning, we can obtain a noise-aware model by pulling away the false positive spans from all prototype regions which can be viewed as the hypersphere with a radius  $r$ .

### 3.4 Training and Evaluation of SpanProto

We provide a brief description of the training and evaluate the algorithm for our framework. During the training stage, the procedure can be found in Algorithm 1. For each training step, we randomly sample one episode data from  $\mathcal{D}_{train}$ , and then enumerate each support example to obtain the span-based loss (Algorithm 1, Line 3-8). For the support set, we obtain the span boundary representations, and then calculate the prototype for each target type (Algorithm 1, Line 9). Further, we leverage the span extractor to perform model inference on each query example to recall multiple candidate spans,

Datasets	Domain	#Types	#Sentences	#Entities
Few-NERD	General	66	188.2k	491.7k
OntoNotes	General	18	103.8k	161.8k
CoNLL-03	News	4	22.1k	35.1k
GUM	Wiki	11	3.5k	6.1k
WNUT-17	Social	6	4.7k	3.1k

Table 1: The statistics of each source dataset.

and then compute the prototypical learning loss and margin-based learning loss (Algorithm 1, Line 10-15). For the model training, the total training steps denote as  $T$ . We first pre-train the span extractor for  $T'$  ( $T' < T$ ) steps, and then both of the span extractor and mention classifier are jointly trained with three objective losses (Algorithm 1, Line 16-17).

During the model evaluation, given one episode data  $\mathcal{E}_{eval} = (\mathcal{S}_{eval}, \mathcal{Q}_{eval}, \mathcal{T}_{eval}) \in \mathcal{D}_{eval}$ . We first calculate the prototype representations based on the support set  $\mathcal{S}_{eval}$ . Then, for each query sentence in  $\mathcal{Q}_{eval}$ , we can utilize the the span extractor to extract all candidate spans  $\hat{\mathcal{M}}_q$ . In the mention classification stage, we calculate the distance between the extracted span and each prototype, and select the type of the nearest prototype as the result. Note that the ground truth in  $\mathcal{D}_{eval}$  is invisible, to recognize the false positives, we remove all spans whose distances between them and all prototypes are larger than  $r$ .

## 4 Experiments

### 4.1 Datasets and Baselines

We choose two widely used  $N$ -way  $K$ -shot based benchmarks to evaluate our SpanProto, including Few-NERD (Ding et al., 2021)<sup>4</sup> and CrossNER (Hou et al., 2020). Specifically, **Few-NERD** is annotated with 8 coarse-grained and 66 fine-grained entity types, which consists of two few-shot settings, i.e. Intra, and Inter. In the **Intra** setting, all entities in the training set, development set, and testing set belong to different coarse-grained types. In contrast, in the **Inter** setting, only the fine-grained entity types are mutually disjoint in different datasets. To make a fair comparison, we utilize the processed episode data released by Ding et al. (2021). **CrossNER** consists of four different NER domains, such as OntoNotes 5.0 (Weischedel et al., 2013), CoNLL-03 (Sang and Meulder, 2003), GUM (Zeldes, 2017) and WNUT-17 (Derczynski et al., 2017). During training, we randomly select

<sup>4</sup><https://github.com/thunlp/Few-NERD>.

Paradigms	Models	Intra					Inter				
		1~2-shot		5~10-shot		Avg.	1~2-shot		5~10-shot		Avg.
		5 way	10 way	5 way	10 way		5 way	10 way	5 way	10 way	
<i>One-stage</i>	ProtoBERT <sup>†</sup>	23.45±0.92	19.76±0.59	41.93±0.55	34.61±0.59	29.94	44.44±0.11	39.09±0.87	58.80±1.42	53.97±0.38	49.08
	NNShot <sup>†</sup>	31.01±1.21	21.88±0.23	35.74±2.36	27.67±1.06	29.08	54.29±0.40	46.98±1.96	50.56±3.33	50.00±0.36	50.46
	StructShot <sup>†</sup>	35.92±0.69	25.38±0.84	38.83±1.72	26.39±2.59	31.63	57.33±0.53	49.46±0.53	57.16±2.09	49.39±1.77	53.34
	CONTaiNER <sup>‡</sup>	40.43	33.84	53.70	47.49	43.87	55.95	48.35	61.83	57.12	55.81
<i>Two-stage</i>	ESD	41.44±1.16	32.29±1.10	50.68±0.94	42.92±0.75	41.83	66.46±0.49	59.95±0.69	74.14±0.80	67.91±1.41	67.12
	DecomMeta	52.04±0.44	43.50±0.59	63.23±0.45	56.84±0.14	53.90	68.77±0.24	63.26±0.40	71.62±0.16	68.32±0.10	67.99
	<b>SpanProto</b>	<b>54.49±0.39</b>	<b>45.39±0.72</b>	<b>73.10±1.15</b>	<b>64.63±0.22</b>	<b>59.40</b>	<b>73.36±0.18</b>	<b>66.26±0.33</b>	<b>82.68±0.42</b>	<b>78.69±0.50</b>	<b>75.25</b>

Table 2: F1 scores with standard deviations on Few-NERD for both inter and intra settings. <sup>†</sup> denotes the results reported in Das et al. (2022). <sup>‡</sup> is taken from Das et al. (2022) with no standard deviations reported.

Paradigms	Models	1-shot					5-shot				
		CONLL-03	GUM	WNUT-17	OntoNotes	Avg.	CONLL-03	GUM	WNUT-17	OntoNotes	Avg.
<i>One-stage</i>	Matching Network <sup>‡</sup>	19.50±0.35	4.73±0.16	17.23±2.75	15.06±1.61	14.13	19.85±0.74	5.58±0.23	6.61±1.75	8.08±0.47	10.03
	ProtoBERT <sup>‡</sup>	32.49±2.01	3.89±0.24	10.68±1.40	6.67±0.46	13.43	50.06±1.57	9.54±0.44	17.26±2.65	13.59±1.61	22.61
	L-TapNet+CDT	44.30±3.15	12.04±0.65	20.80±1.06	15.17±1.25	23.08	45.35±2.67	11.65±2.34	23.30±2.80	20.95±2.81	25.31
<i>Two-stage</i>	DecomMeta	46.09±0.44	17.54±0.98	25.14±0.24	34.13±0.92	30.73	58.18±0.87	31.36±0.91	31.02±1.28	45.55±0.90	41.53
	<b>SpanProto</b>	<b>47.70±0.51</b>	<b>20.16±0.80</b>	<b>30.19±0.94</b>	<b>37.91±0.79</b>	<b>33.99</b>	<b>61.61±1.03</b>	<b>43.75±0.50</b>	<b>31.37±0.94</b>	<b>49.04±0.93</b>	<b>46.44</b>

Table 3: F1 scores with standard deviations on CrossNER. <sup>‡</sup> denotes the results reported in Hou et al. (2020).

two of them as the training set, and the remaining two others are used for the development set and the testing set. The final results are derived from the testing set. We follow (Ma et al., 2022b) to utilize the generated episode data provided by Hou et al. (2020)<sup>5</sup>.

For the baselines, we choose multiple strong approaches from the paradigms of *one-stage* and *two-stage*. Concretely, the one-stage paradigm consists of ProtoBERT (Snell et al., 2017), Matching Network (Vinyals et al., 2016), StructShot (Yang and Katiyar, 2020), NNShot (Yang and Katiyar, 2020), CONTaiNER (Das et al., 2022) and L-TapNet+CDT (Hou et al., 2020). The two-stage paradigm includes ESD (Wang et al., 2021) and DecomMeta (Ma et al., 2022b).

## 4.2 Implementation Details

We choose BERT-base-uncased (Devlin et al., 2019) from HuggingFace<sup>6</sup> as the default pre-trained encoder  $\mathcal{F}$ . The max sequence length we set is 128. We choose AdamW as the optimizer with a warm up rate of 0.1. The training steps  $T$  and  $T'$  are set as 2000 and 200, respectively. We use the grid search to find the best hyper-parameters for each benchmark<sup>7</sup>. As a result, the threshold  $\theta$

<sup>5</sup><https://atmahou.github.io/attachments/ACL2020data.zip>.

<sup>6</sup><https://huggingface.co/transformers>.

<sup>7</sup>The details are shown in Appendix B

and the margin  $r$  are set as 0.8 and 3.0, respectively. We choose five random seeds from {12, 21, 42, 87, 100} and report the averaged results with standard deviations. We implement our model by Pytorch 1.8, and train the model with 8 V100-32G GPUs.

## 4.3 Main Results

Table 2 and 3 illustrate the main results of our proposal compared with other baselines. We thus make the following observations: 1) Our proposed SpanProto achieves the best performance and outperforms the state-of-the-art baselines with a large margin. Compared with DecomMeta (Ma et al., 2022b), the overall averaged results over Few-NERD Intra and Few-NERD Inter are improved by 5.5% and 7.26%, respectively. Likewise, we also have more than 3.0% advantages on CrossNER. 2) All the methods in the two-stage paradigm perform better than those one-stage approaches, which indicates the merit of the span-based approach for the task of few-shot NER. 3) In Few-NERD, the overall performance of the Intra scenario is lower than Inter. This phenomenon reflects that Intra is more challenging than Inter where the coarse-grained types are different in training/development/testing set. Despite this, we still obtain satisfying effectiveness. Results suggest that our method can adapt to a new domain in which the coarse-grained and fine-grained entity types are both unseen.

Methods	Few-NERD		CrossNER	
	Intra	Inter	1-shot	5-shot
<b>SpanProto</b>	<b>59.40</b>	<b>75.25</b>	<b>33.99</b>	<b>46.44</b>
w/o. Span Extractor	29.24	49.08	13.43	22.61
w/o. Mention Classifier	18.41	26.36	10.53	6.08
w/o. Margin-based Learning	54.29	71.37	30.88	43.37

Table 4: The ablation study results (averaged F1 score %) for Few-NERD and CrossNER. Detail results are listed in Appendix C.

Methods	Few-NERD		CrossNER	
	Intra	Inter	1-shot	5-shot
ESD	70.56	70.99	-	-
DecomMeta	76.11	76.48	46.53	54.58
<b>SpanProto</b>	<b>84.02</b>	<b>84.55</b>	<b>63.75</b>	<b>63.51</b>

Table 5: The averaged performance (F1 score %) in the span extraction stage over Few-NERD and CrossNER.

#### 4.4 Ablation Study

We conduct an ablation study to investigate the characteristics of the main components in SpanProto. We implement the following list of variants of SpanProto for the experiments. 1) w/o. Span Extractor: we remove the span extractor, and train the model with a conventional token-wise prototypical network<sup>8</sup>. 2) w/o. Mention Classifier: we remove all techniques in the mention classifier. To classify the span, we directly leverage the K-Means algorithm. 3) w/o. Margin-based Learning: we only remove the margin-based learning objective. More details of these variants are shown in Appendix A.

As shown in Table 4, the results demonstrate that 1) the performance of SpanProto drops when removing each component, which shows the significance of all components. 2) When removing the span extractor, the averaged F1 scores are decreased by more than 20%, indicating that the span extractor which bypasses the issues of multiple non-entity classes and token-level label dependency does make a contribution to the model performance. 3) SpanProto outperforms w/o. Margin-based Learning, which demonstrates that a model jointly trained by prototypical learning and margin-based learning objectives can mitigate the problem of false positives.

#### 4.5 Performance of the Span Extractor

To further analyze how the span extractor contributes to the few-shot NER. Specifically, we aim

<sup>8</sup>When removing the span extractor, the margin-based learning will not work because no spans can be recalled.

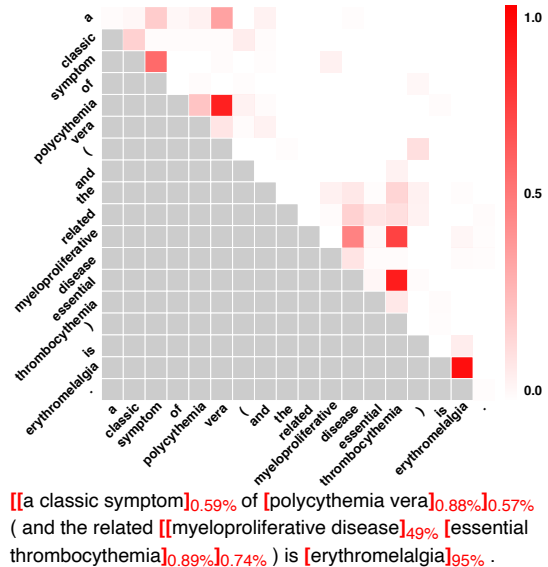


Figure 3: The case visualization of the global boundary matrix. The span could be an entity if the corresponding color is dark. (Best viewed in color.)

to solve the following research questions about the span extractor: 1) **RQ1**: Whether the proposed span extractor is better than previous methods? 2) **RQ2**: How does the model learn explicit span boundary information? 3) **RQ3**: How do the hyperparameters  $T'$  and  $\theta$  affect the model performance? **Comparison with Baselines.** We first perform a comparison between our proposal and previous methods. Specifically, we choose two strong baselines: 1) ESD (Wang et al., 2021) which leverages a sliding window to recall all candidate spans, and 2) DecomMeta (Ma et al., 2022b) which leverages the sequence labeling method with “BIOE” rules to detect the spans. From Table 5, we see that our span extractor outperforms other baselines by a large margin, which indicates that our method can produce more accurate predictions for span extraction.

**Case Study for the Span Extractor.** To explore how the model learns the span boundary, we randomly select one query sentence from Few-NERD and obtain the predicted global boundary matrix. As shown in Figure 3, our span extractor successfully recognizes the explicit boundary for the input sentence. In addition, we find that there existing nested entity mentions in few-shot NER, which share the same overlap sub-sequence. For example, “myeloproliferative disease”, “essential thrombocythemia”, and “myeloproliferative disease essential thrombocythemia” are the nested entities. Based on the global boundary matrix, our approach

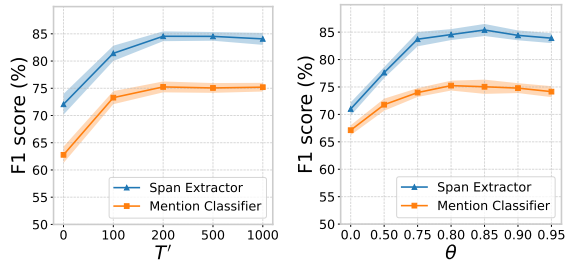


Figure 4: Hyper-parameter effectiveness (averaged F1 score %) of  $T'$  and  $\theta$  over Few-NERD Inter. We report the F1 curve for both span extractor and mention classifier.  $\theta = 0.0$  denotes to recall all mention spans, which is equal to the method of ESD (Wang et al., 2021).

Margin $r$	Few-NERD		CrossNER	
	Intra	Inter	1-shot	5-shot
$r = 1$	52.10	64.98	27.42	40.85
$r = 2$	58.08	74.60	<b>34.13</b>	46.19
$r = 3$	<b>59.40</b>	<b>75.25</b>	33.99	<b>46.44</b>
$r = 4$	58.15	74.43	33.04	45.82
$r = 5$	56.40	73.61	31.99	44.22
$r = 6$	55.10	72.08	31.17	43.50

Table 6: The parameter analysis of the margin  $r$ .

can be extended to few-shot nested NER without difficulty, which is ignored by previous works.

**Effectiveness of Hyper-parameters.** We aim to investigate the influence of two hyper-parameters  $T'$  and  $\theta$ , where  $T'$  is the pre-training step number of the span extractor, and  $\theta$  denotes the confidence threshold for predicting candidate spans. We conduct experiments over Few-NERD Inter. As shown in Figure 4. We can draw the following suggestions: 1) For the parameter  $T'$ , we find that pre-training the span extractor for  $T' = 200$  steps does improve the performance for both the span extractor and the mention classifier. When  $T' > 200$ , the F1 scores will decrease due to the over-fitting problem. 2) For the parameter  $\theta$ , we observe that the overall F1 score increases when increasing the threshold  $\theta$ , and we achieve the best performance when  $\theta \in [0.8, 0.85]$ .

#### 4.6 Effectiveness of the Mention Classifier

In this section, we specifically explore the effectiveness of the mention classifier by answering two questions. 1) **RQ4**: How does the hyper-parameter  $r$  affects the model performance? and 2) **RQ5**: How does the mention classifier adjust the semantic representations of each entity span?

**Effectiveness of the Hyper-parameter.** We further perform a hyper-parameter study on the margin

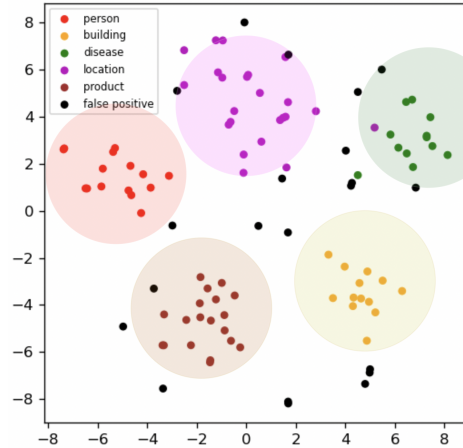


Figure 5: The t-SNE visualization of the span representations with 5-way 5~10-shot episode data from Few-NERD Inter. The points with different colors denote the entity span with different types. The circle represents the prototype region. (Best viewed in color.)

value  $r$ , which aims to separate the false positives via the margin-based learning objective. We conduct the experiments over Few-NERD and CrossNER, and the report the averaged F1 scores in Table 6. Through the experimental results, we find the best value is in  $2 \leq r \leq 3$ . Specifically, when the margin value is larger than 3, the performance is similar to the variants SpanProto (w/o. Margin-based Learning). In other words, it is hard to detect false positives. In addition, we find that the performance declines a lot when  $r < 2$ . We think that the general distance between the prototype and the true positive span is larger than 2. If the margin value is very small, more and more true positives could be viewed as the noises and be discarded.

**Visualizations.** We end this section with an investigation of how the SpanProto learns the representations in the semantic space. We randomly choose a 5-way 5~10-shot episode data from Few-NERD Inter, and then obtain the visualization by t-SNE (Van der Maaten and Hinton, 2008) toolkit. As shown in Figure 5, our method can cluster the span representations into the corresponding type prototype region, which can be viewed as a circle with a radius of 2. In addition, owing to the proposed margin-based learning, we observe that most of the false positives can be separated successfully.

#### 4.7 Error Analysis

Finally, we follow Wang et al. (2021) to conduct an error analysis into two types, i.e. “FP-Span” and “FP-Type”. As shown in Table 7, our SpanProto



Methods	F1	FP-Span	FP-Type
ProtoBERT	44.44	86.70	13.30
NNShot	54.29	84.70	15.30
StructShot	57.33	80.00	20.00
ESD	66.46	72.80	27.20
DecomMeta	76.11	76.48	46.53
<b>SpanProto</b>	<b>73.36</b>	<b>61.40</b>	<b>10.90</b>

Table 7: Error analysis (%) of 5-way 1 2-shot on FewNERD-INTER. “FP-Span” denotes extracted entities with the wrong span boundary, and “FP-Type” represents extracted entities with the right span boundary but the wrong entity type.

outperforms other strong baselines and has much less false positive prediction errors. Specifically, we achieve 61.40% of “FP-Span” and the result reduces by more than 10%, which demonstrates the effectiveness of the span extractor. Meanwhile, we also obtain the lowest error rate of “FP-Type”, owing to the introduction of the margin-based learning objective in the mention classifier.

## 5 Conclusion

We propose a span-based prototypical network (SpanProto) for few-shot NER, which is a *two-stage* approach includes span extraction and mention classification. To improve the performance of the span extraction, we introduce a global boundary matrix to urge the model to learn explicit boundary information regardless of token-wise label dependency. We further utilize prototypical learning to adjust the span representations and solve the issue of false positives by the margin-based learning objective. Extensive experiments demonstrate that our framework consistently outperforms strong baselines. In the future, we will extend our framework to other NLP tasks, such as slot filling, Part-Of-Speech (POS) tagging, etc.

## Limitations

Our proposed span extractor is based on the global boundary matrix, which could consume more memory than the conventional sequence labeling methods. This drives us to further improve the overall space efficiency of the framework. In addition, we only focus on the few-shot NER in a  $N$ -way  $K$ -shot settings in this paper. But, we think it is possible to extend our work to other NER scenarios, such as transfer learning, semi-supervised learning or full-data supervised learning. We also leave them as our future research.

## Ethical Considerations

Our contribution in this work is fully methodological, namely a Span-based Prototypical Network (SpanProto) to boost the performance of the few-shot NER. Hence, there are no direct negative social impacts of this contribution.

## Acknowledgments

This work has been supported by the National Natural Science Foundation of China under Grant No. U1911203, Alibaba Group through the Alibaba Innovation Research Program, the National Natural Science Foundation of China under Grant No. 61877018, the Research Project of Shanghai Science and Technology Commission (20dz2260300) and The Fundamental Research Funds for the Central Universities.

## References

- Sarkar Snigdha Sarathi Das, Arzoo Katiyar, Rebecca J. Passonneau, and Rui Zhang. 2022. Container: Few-shot named entity recognition via contrastive learning. In *ACL*, pages 6338–6353.
- Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *EMNLP*, pages 140–147.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.
- Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. 2021. Few-nerd: A few-shot named entity recognition dataset. In *ACL*, pages 3198–3213.
- Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135.
- Alexander Fritzier, Varvara Logacheva, and Maksim Kretov. 2019. Few-shot classification in named entity recognition task. In *SAC*, pages 993–1000.
- Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *EMNLP*, pages 6894–6910.
- Ruiying Geng, Binhua Li, Yongbin Li, Jian Sun, and Xiaodan Zhu. 2020. Dynamic memory induction networks for few-shot text classification. In *ACL*, pages 1087–1094.

- Yutai Hou, Wanxiang Che, Yongkui Lai, Zhihan Zhou, Yijia Liu, Han Liu, and Ting Liu. 2020. Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network. In *ACL*, pages 1381–1393.
- Jiixin 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 *EMNLP*, pages 10408–10423.
- Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.
- Mike Huisman, Jan N. van Rijn, and Aske Plaat. 2021. A survey of deep meta-learning. *Artif. Intell. Rev.*, 54(6):4483–4541.
- Vivek Kulkarni, Yashar Mehdad, and Troy Chevalier. 2016. Domain adaptation for named entity recognition in online media with word embeddings. *CoRR*, abs/1612.00148.
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL*, pages 260–270.
- Elad Levi, Tete Xiao, Xiaolong Wang, and Trevor Darrell. 2021. Rethinking preventing class-collapsing in metric learning with margin-based losses. In *ICCV*, pages 10296–10305.
- Jie Ma, Miguel Ballesteros, Srikanth Doss, Rishita Anubhai, Sunil Mallya, Yaser Al-Onaizan, and Dan Roth. 2022a. Label semantics for few shot named entity recognition. In *ACL*, pages 1956–1971.
- Tingting Ma, Huiqiang Jiang, Qianhui Wu, Tiejun Zhao, and Chin-Yew Lin. 2022b. Decomposed meta-learning for few-shot named entity recognition. In *ACL*, pages 1584–1596.
- Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL*.
- Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*, pages 1756–1765.
- Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *NAACL*, pages 142–147.
- Adam Santoro, Sergey Bartunov, Matthew M. Botvinick, Daan Wierstra, and Timothy P. Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *ICML*, volume 48, pages 1842–1850.
- Yongliang Shen, Xinyin Ma, Zeqi Tan, Shuai Zhang, Wen Wang, and Weiming Lu. 2021. Locate and label: A two-stage identifier for nested named entity recognition. In *ACL*, pages 2782–2794.
- Jiawei Sheng, Shu Guo, Zhenyu Chen, Juwei Yue, Lihong Wang, Tingwen Liu, and Hongbo Xu. 2020. Adaptive attentional network for few-shot knowledge graph completion. In *EMNLP*, pages 1681–1691.
- Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. In *NIPS*, pages 4077–4087.
- Jianling Su. 2021. Globalpointer: A unified framework for nested and flat named entity recognition.
- Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.
- Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *NIPS*, pages 3630–3638.
- Chengyu Wang, Minghui Qiu, Taolin Zhang, Tingting Liu, Lei Li, Jianing Wang, Ming Wang, Jun Huang, and Wei Lin. 2022a. EasyNLP: A comprehensive and easy-to-use toolkit for natural language processing. *CoRR*, abs/2205.00258.
- Jianing Wang, Chengyu Wang, Minghui Qiu, Qihui Shi, Hongbin Wang, Jun Huang, and Ming Gao. 2022b. KECP: knowledge enhanced contrastive prompting for few-shot extractive question answering. *CoRR*, abs/2205.03071.
- Peiyi Wang, Runxin Xu, Tianyu Liu, Qingyu Zhou, Yunbo Cao, Baobao Chang, and Zhifang Sui. 2021. An enhanced span-based decomposition method for few-shot sequence labeling. In *NAACL*, pages 5012–5024.
- Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv.*, 53(3):1–63:34.
- Jason W. Wei and Kai Zou. 2019. EDA: easy data augmentation techniques for boosting performance on text classification tasks. In *EMNLP*, pages 6381–6387.
- Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*, 23.

Models	Intra					Inter				
	1~2-shot		5~10-shot		Avg.	1~2-shot		5~10-shot		Avg.
	5 way	10 way	5 way	10 way		5 way	10 way	5 way	10 way	
<b>SpanProto</b>	<b>54.49±0.39</b>	<b>45.39±0.72</b>	<b>73.10±1.15</b>	<b>64.63±0.22</b>	<b>59.40</b>	<b>73.36±0.18</b>	<b>66.26±0.33</b>	<b>82.68±0.42</b>	<b>78.69±0.50</b>	<b>75.25</b>
w/o. Span Extractor	23.10±0.37	21.63±0.29	37.91±0.44	34.32±0.44	29.24	45.17±0.25	36.18±0.35	59.52±1.0	55.45±0.90	49.08
w/o. Mention Classifier	14.02±0.25	11.33±0.33	31.20±0.75	17.09±0.20	18.41	25.40±0.22	19.77±0.36	26.88±0.41	33.39±0.50	26.36
w/o. Margin-based Learning	51.92±0.40	40.32±0.52	68.10±0.88	56.82±0.19	54.29	68.07±0.22	62.52±0.30	79.10±0.35	75.79±0.33	71.37

Table 8: F1 scores with standard deviations on Few-NERD for both inter and intra settings. <sup>†</sup> denotes the results reported in Das et al. (2022). <sup>‡</sup> is taken from Das et al. (2022) with no standard deviations reported.

Models	1-shot					5-shot				
	CONLL-03	GUM	WNUT-17	OntoNotes	Avg.	CONLL-03	GUM	WNUT-17	OntoNotes	Avg.
<b>SpanProto</b>	<b>47.70±0.51</b>	<b>20.16±0.80</b>	<b>30.19±0.94</b>	<b>37.91±0.79</b>	<b>33.99</b>	<b>61.61±1.03</b>	<b>43.75±0.50</b>	<b>31.37±0.94</b>	<b>49.04±0.93</b>	<b>46.44</b>
w/o. Span Extractor	9.10±0.37	11.13±0.21	17.28±0.44	16.21±0.40	13.43	25.63±0.23	21.76±0.39	33.31±0.33	9.74±0.50	22.61
w/o. Mention Classifier	13.68±0.35	9.13±0.30	11.10±0.35	8.21±0.23	10.53	6.17±0.29	5.02±0.31	5.81±0.31	7.32±0.20	6.08
w/o. Margin-based Learning	32.11±0.16	29.10±0.20	30.32±0.31	31.99±0.13	30.88	40.07±0.20	43.50±0.29	44.80±0.31	45.11±0.27	43.37

Table 9: F1 scores with standard deviations on CrossNER. <sup>†</sup> denotes the results reported in Hou et al. (2020).

Hyper-parameter	Value
Batch Size	{1, 2, 4, 8}
Learning Rate	{1e-5, 2e-5, 5e-5, 1e-4, 2e-4}
Dropout Rate	{0.1, 0.3, 0.5}
$T'$	{0, 100, 200, 500, 1,000}
$\theta$	{0.0, 0.50, 0.75, 0.80, 0.85, 0.90, 0.95}
$r$	{1, 2, 3, 4, 5, 6}

Table 10: The searching scope for each hyper-parameter.

Shuhui Wu, Yongliang Shen, Zeqi Tan, and Weiming Lu. 2022. Propose-and-refine: A two-stage set prediction network for nested named entity recognition. *CoRR*, abs/2204.12732.

Yi Yang and Arzoo Katiyar. 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. In *EMNLP*, pages 6365–6375.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *ACL*, pages 6470–6476.

Amir Zeldes. 2017. The GUM corpus: creating multilayer resources in the classroom. *Lang. Resour. Evaluation*, 51(3):581–612.

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

## A Details of Our Variants

**SpanProto w/o. Span Extractor.** We remove the span extractor. We leverage the standard prototypical network to perform the token-level type classification, which is the same as ProtoBERT.

**SpanProto w/o. Mention Classifier.** We remove the mention classifier. To support the classification, we directly use the K-Means algorithm. Specifically, we first train the span extractor on the support set and extract all candidates for each query example. We then obtain the span embedding based on the contextual representations. Thus, for each episode data, we can obtain all span representations, we leverage the K-Means to obtain  $N$  cluster on all support spans, and then predict each type of the query span based on the nearest cluster.

**SpanProto w/o. Margin-based Learning.** We remove the margin-based learning objective to validate its contribution. Therefore, we do not obtain the false positives from the query set in the training episode. Specifically, we first obtain all candidates for each query example, and then directly classify each span based on the prototypical network.

## B Details of the Grid Search

The searching scope of each hyper-parameter is shown in Table 10. Note that, the batch size in the  $N$ -way  $K$ -shot setting means the number of episode data in one batch.

## C Details of the Ablation Study

The detail results of ablation study are listed in Table 8 and Table 9.