Decomposed Meta-Learning for Few-Shot Named Entity Recognition

Tingting Ma[1†*], **Huiqiang Jiang**[2*], **Qianhui Wu**[2*], **Tiejun Zhao**[1], **Chin-Yew Lin**[2]
[1]Harbin Institute of Technology, Harbin, China
[2]Microsoft Research Asia
hittingtingma@gmail.com
{hjiang, qianhuiwu, cyl}@microsoft.com, tjzhao@hit.edu.cn

Abstract

Few-shot named entity recognition (NER) systems aim at recognizing novel-class named entities based on only a few labeled examples. In this paper, we present a decomposed meta-learning approach which addresses the problem of few-shot NER by sequentially tackling few-shot span detection and few-shot entity typing using meta-learning. In particular, we take the few-shot span detection as a sequence labeling problem and train the span detector by introducing the model-agnostic meta-learning (MAML) algorithm to find a good model parameter initialization that could fast adapt to new entity classes. For few-shot entity typing, we propose MAML-ProtoNet, *i.e.*, MAML-enhanced prototypical networks to find a good embedding space that can better distinguish text span representations from different entity classes. Extensive experiments on various benchmarks show that our approach achieves superior performance over prior methods.[1]

1 Introduction

Named entity recognition (NER) aims at locating and classifying text spans into pre-defined entity classes such as locations, organizations, *etc*. Deep neural architectures have shown great success in fully supervised NER (Lample et al., 2016; Ma and Hovy, 2016; Chiu and Nichols, 2016; Peters et al., 2017) with a fair amount of labeled data available for training. However, in practical applications, NER systems are usually expected to rapidly adapt to some new entity types unseen during training. It is costly while not flexible to collect a number of additional labeled data for these types. As a result, the problem of few-shot NER, which involves learning unseen entity types from only a few labeled examples for each class (also known as *support examples*), has attracted considerable attention from the research community in recent years.

Previous studies on few-shot NER are typically based on token-level metric learning, in which a model compares each query token to the prototype (Snell et al., 2017) of each entity class or each token of support examples and assign the label according to their distances (Fritzler et al., 2019; Hou et al., 2020; Yang and Katiyar, 2020). Alternatively, some more recent attempts have switched to span-level metric-learning (Yu et al., 2021; Wang et al., 2021a) to bypass the issue of token-wise label dependency while explicitly utilizing phrasal representations.

However, these methods based on metric learning might be less effective when encountering large domain gap, since they just directly use the learned metric without any further adaptation to the target domain. In other words, they do not fully explore the information brought by the support examples. There also exist additional limitations in the current methods based on span-level metric learning. First, the decoding process requires careful handling of overlapping spans due to the nature of span enumeration. Second, the class prototype corresponding to non-entities (*i.e.*, prototype of the "O" class) is usually noisy because non-entity common words in the large vocabulary rarely share anything together in common. Moreover, when targeting at a different domain, the only available information useful for domain transfer is the limited number of support examples. Unfortunately, these key examples are only used for inference-phase similarity calculation in previous methods.

To tackle these limitations, this paper presents a decomposed meta-learning framework that addresses the problem of few-shot NER by sequentially conducting *few-shot entity span detection* and *few-shot entity typing* respectively via meta-

[*]Equal contributions.
[†]Work during internship at Microsoft Research Asia.
[1]Our implementation is publicly available at https://github.com/microsoft/vert-papers/tree/master/papers/DecomposedMetaNER

Findings of the Association for Computational Linguistics: ACL 2022, pages 1584 - 1596

learning. Specifically, for *few-shot span detection*, we model it as a sequence labeling problem to avoid handling overlapping spans. Note that the detection model aims at *locating* named entities and is class-agnostic. We only feed the detected entity spans to the typing model for entity class inference, and hence the problem of noisy "O" prototype could also be eliminated. When training the span detector, we specifically use the model-agnostic meta-learning (MAML) (Finn et al., 2017) algorithm to find a good model parameter initialization that could fast adapt to new entity classes with learned class-agnostic meta-knowledge of span boundaries after updating with the target-domain support examples. The boundary information of domain-specific entities from the support examples is supposed to be effectively leveraged via these update steps such that the model could better transfer to the target domain. For *few-shot entity typing*, we implement the typing model with standard prototypical networks (Snell et al., 2017, ProtoNet), and propose MAML-ProtoNet to narrow the gap between source domains and the target domain. Compared with ProtoNet which only uses support examples for inference-phase similarity calculation, the proposed MAML-Proto additionally utilizes these examples to modify the shared embedding space of spans and prototypes by clustering spans representations from the same entity class while dispersing those from different entity classes for more accurate predictions.

We evaluate our proposed framework on several benchmark datasets with different few-shot settings. Experimental results show that our framework achieves superior performance over previous state-of-the-art methods. We also conduct qualitative and quantitative analyses over how the different strategies to conduct meta-learning might affect the performance.

2 Task Definition

Given an input sequence $\boldsymbol{x} = \{x_i\}_{i=1}^{L}$ with L tokens, an NER system is supposed to output a label sequence $\boldsymbol{y} = \{y_i\}_{i=1}^{L}$, where x_i is the i-th token, $y_i \in \mathcal{Y} \cup \{\mathrm{O}\}$ is the label of x_i, $\mathcal{Y}$ is the pre-defined entity class set, and O denotes non-entities.

In this paper, we focus on the standard N-way K-shot setting as in Ding et al. (2021). An example of 2-way 1-shot episode is shown in Table 1. In the training phase, we consider training episodes $\mathcal{E}_{train} = \{(\mathcal{S}_{train}, \mathcal{Q}_{train}, \mathcal{Y}_{train})\}$ built from source-domain labeled data, where $\mathcal{S}_{train} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N\times K}$ denotes the support set, $\mathcal{Q}_{train} = \{\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)}\}_{j=1}^{N\times K'}$ denotes the query set, $\mathcal{Y}_{train}$ denotes the set of entity classes, and $|\mathcal{Y}_{train}| = N$. In the testing phase, we consider novel episodes $\mathcal{E}_{new} = \{(\mathcal{S}_{new}, \mathcal{Q}_{new}, \mathcal{Y}_{new})\}$ constructed with data from target domains in a similar way. In the few-shot NER task, a model learned with training episodes $\mathcal{E}_{train}$ is expected to leverage the support set $\mathcal{S}_{new} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N\times K}$ of a novel episode $(\mathcal{S}_{new}, \mathcal{Q}_{new}, \mathcal{Y}_{new}) \in \mathcal{E}_{new}$ to make predictions on the query set $\mathcal{Q}_{new} = \{\boldsymbol{x}^{(j)}\}_{j=1}^{N\times K'}$. Here, $\mathcal{Y}_{new}$ denotes the set of entity classes with a cardinality of N. Note that, $\forall\, \mathcal{Y}_{train}, \mathcal{Y}_{new}, \mathcal{Y}_{train} \cap \mathcal{Y}_{new} = \emptyset$.

Target Types $\mathcal{Y}$	[person-actor], [art-film]
Support set $\mathcal{S}$	(1) *Jack Gordon*[person-actor] (born 27 June 1985) is an English actor . (2) This location had also been used to shoot the film " *Saving Private Ryan*[art-film] " .
Query Set $\mathcal{Q}$	Kurland starred in " Taps " , which won first prize at the Rhode Island International Film Festival in 2006 .
Expected output	*Kurland*[person-actor] starred in " *Taps*[art-film] " , which won first prize at the Rhode Island International Film Festival in 2006 .

Table 1: An example of the simplest 2-way 1-shot setting, which contains two entity classes and each class has one example (shot) in the support set $\mathcal{S}$. Different colors indicate different entity classes.

3 Methodology

Figure 1 illustrates the overall framework of our decomposed meta-learning approach for few-shot named entity recognition. It is composed of two steps: *entity span detection* and *entity typing*.

3.1 Entity Span Detection

The span detection model aims at locating all the named entities in an input sequence. The model should be type-agnostic, *i.e.*, we do not differentiate the specific entity classes. As a result, the parameters of the model can be shared across different domains and classes. With this in mind, we train the span detection model by exploiting model-agnostic meta-learning (Finn et al., 2017) to promote the learning of the domain-invariant internal representations rather than domain-specific features. In this way, the meta-learned model is

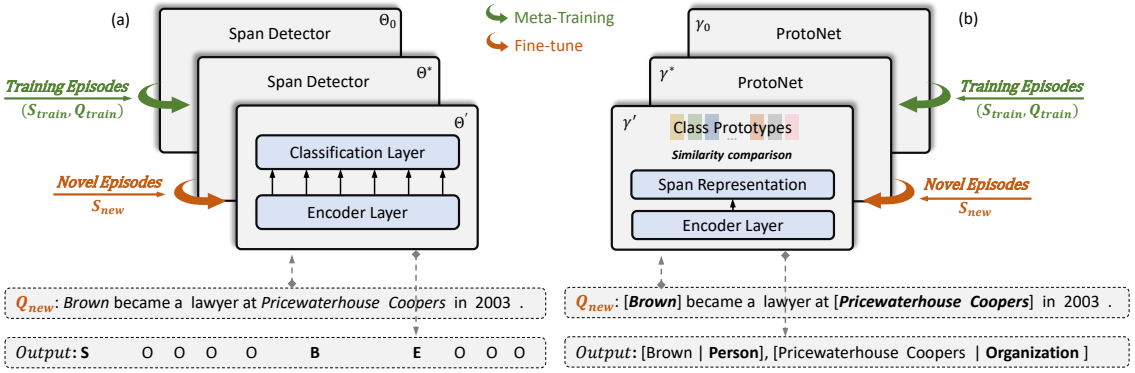


Figure 1: The framework of our proposed approach is decomposed into two modules: (a) entity span detection with parameters Θ and (b) entity typing with parameters γ. Two modules are trained independently using (S_{train}, Q_{train}). At meta-test time, these two modules firstly are finetuned on the support set S_{new}, then given a query sentence in Q_{new}, the spans detected by (a) are sent to (b) for entity typing.

expected to be more sensitive to target-domain support examples, and hence only a few fine-tune steps on these examples can make rapid progress without overfitting.

3.1.1 Basic Detector

Model In this work, we implement a strong span detector via sequence labeling. We apply the BIOES tagging scheme instead of the standard BIO2 to provide more specific and fine-grained boundary information of entity spans.[2] Given an input sequence $\boldsymbol{x} = \{x_i\}_{i=1}^{L}$ with L tokens, we first leverage an encoder f_θ to obtain contextualized representations $\boldsymbol{h} = \{h_i\}_{i=1}^{L}$ for all tokens:

$$\boldsymbol{h} = f_\theta(\boldsymbol{x}). \tag{1}$$

With each h_i derived, we then use a linear classification layer to compute the probability distribution of labels that indicate whether the token x_i is inside an entity or not, using a *softmax* function:

$$p(x_i) = \text{softmax}(Wh_i + b), \tag{2}$$

where $p(x_i) \in \mathbb{R}^{|C|}$ with $C = \{B, I, O, E, S\}$ being the label set. $\Theta = \{\theta, W, b\}$ are trainable parameters.

Training Generally, the learning loss *w.r.t.* $\boldsymbol{x}$ is modeled as the averaged cross-entropy of the predicted label distribution and the ground-truth one over all tokens. Following Wu et al. (2020), we add a maximum term here to mitigate the problem of insufficient learning for tokens with relatively higher losses, which can be formulated as:

$$\begin{aligned}\mathcal{L}(\Theta) = &\frac{1}{L}\sum_{i=1}^{L}\text{CrossEntropy}\left(y_i, p(x_i)\right) \\ &+ \lambda \max_{i\in\{1,2,\ldots,L\}}\text{CrossEntropy}\left(y_i, p(x_i)\right),\end{aligned} \tag{3}$$

where $\lambda \geq 0$ is a weighting factor.

Inference For inference, we use the learned model to predict the label distribution for each token in a given test case. We apply the Viterbi algorithm (Forney, 1973) for decoding. It is worthy to note that we do not train a transition matrix here, but simply add constraints to ensure that the predicted label sequence would not violate the BIOES tagging scheme.

3.1.2 Meta-Learning Procedure

Here we elaborate on the proposed meta-learning procedure which consists of two phases: meta-training on $\mathcal{E}_{train}$ and meta-testing on $\mathcal{E}_{new}$. The Appendix A.1 describes the general framework of meta-learning for reference.

Meta-Training In this phase, we train a mention detection model $\mathcal{M}_\Theta$ by repeatedly simulating the *Meta-Testing* phase, where the meta-trained model is fine-tuned with the support set of a novel episode and then tested on the corresponding query set.

Specifically, we first randomly sample an episode $(\mathcal{S}_{train}^{(i)}, \mathcal{Q}_{train}^{(i)}, \mathcal{Y}_{train}^{(i)})$ from $\mathcal{E}_{train}$ and perform *inner-update*:

$$\Theta_i' = U^n(\Theta; \alpha, \mathcal{S}_{train}^{(i)}), \tag{4}$$

[2] We found BIOES to be stronger than BIO for *type-agnostic span detection* as it explicitly encourages the model to learn more specific and fine-grained boundary information. Besides, our entity typing model aims to assign an entity type for each *detected* span, which does not involve any tagging scheme.

where U^n denotes n-step gradient updates with the learning rate α to minimize $\mathcal{L}(\Theta; \mathcal{S}^{(i)}_{train})$, *i.e.*, the loss in Eq. (3) derived from the support set $\mathcal{S}^{(i)}_{train}$.

We then evaluate Θ' on the query set $\mathcal{Q}^{(i)}_{train}$ and perform *meta-update* by aggregating multiple episodes:

$$\min_{\Theta} \sum_{i} \mathcal{L}(\Theta'_i; \mathcal{Q}^{(i)}_{train}). \tag{5}$$

Since Eq. (5) involves the second order derivative, we employ its first-order approximation for computational efficiency:

$$\Theta \leftarrow \Theta - \beta \sum_{i} \nabla_{\Theta'_i} \mathcal{L}(\Theta'_i; \mathcal{Q}^{(i)}_{train}), \tag{6}$$

where β denotes the learning rate used in meta-update.

Meta-Testing In this phase, we first fine-tune the meta-trained span detection model $\mathcal{M}_{\Theta^*}$ with the loss function defined in Eq. (3) on the support set $\mathcal{S}_{new}$ from a novel episode, and then make predictions for corresponding query examples $\mathcal{Q}_{new}$ with the fine-tuned model $\mathcal{M}_{\Theta'}$.

3.2 Entity Typing

For entity typing, we aim to assign a specific entity class for each span output by the mention detection model. In the few-shot learning scenario, we take the prototypical networks (ProtoNet) (Snell et al., 2017) as the backbone for entity typing. To explore the knowledge brought by support examples from a novel episode, we propose to enhance the ProtoNet with the model-agnostic meta-learning (MAML) algorithm (Finn et al., 2017) for a more representative embedding space, where text spans from different entity classes are more distinguishable to each other.

3.2.1 Basic Model: ProtoNet

Span Representation Given an input sequence with L tokens $\boldsymbol{x} = \{x_i\}_{i=1}^{L}$, we use an encoder g_γ to compute contextual token representations $\boldsymbol{h} = \{h_i\}_{i=1}^{L}$ in the same way as Eq. (1):

$$\boldsymbol{h} = g_\gamma(\boldsymbol{x}). \tag{7}$$

Assume $x_{[i,j]}$ being the output of the span detection model which starts at x_i and ends at x_j, we compute the span representation of $x_{[i,j]}$ by averaging representations of all tokens inside $x_{[i,j]}$:

$$s_{[i,j]} = \frac{1}{j-i+1} \sum_{k=i}^{j} h_k. \tag{8}$$

Class Prototypes Let $S_k = \{x_{[i,j]}\}$ denotes the set of entity spans contained in a given support set $\mathcal{S}$ that belongs to the entity class $y_k \in \mathcal{Y}$, we compute the prototype c_k for each entity class y_k by averaging span representations of all $x_{[i,j]} \in S_k$:

$$c_k(\mathcal{S}) = \frac{1}{|S_k|} \sum_{x_{[i,j]} \in S_k} s_{[i,j]}. \tag{9}$$

Training Given a training episode denoted as $(\mathcal{S}_{train}, \mathcal{Q}_{train}, \mathcal{Y}_{train})$, we first utilize the support set $\mathcal{S}_{train}$ to compute prototypes for all entity classes in $\mathcal{Y}_{train}$ via Eq. (9). Then, for each span $x_{[i,j]}$ from the query set $\mathcal{Q}_{train}$, we calculate the probability that $x_{[i,j]}$ belongs to an entity class $y_k \in \mathcal{Y}$ based on the distance between its span representation $s_{[i,j]}$ and the prototype of y_k:

$$p(y_k; x_{[i,j]}) = \frac{\exp\left\{-d\left(c_k(\mathcal{S}_{train}), s_{[i,j]}\right)\right\}}{\sum_{y_i \in \mathcal{Y}} \exp\left\{-d\left(c_i(\mathcal{S}_{train}), s_{[i,j]}\right)\right\}}, \tag{10}$$

where $d(\cdot,\cdot)$ denotes the distance function. Let $y_{[i,j]} \in \mathcal{Y}$ denote the ground-truth entity class *w.r.t.* $x_{[i,j]}$, the parameters of the ProtoNet, *i.e.*, γ, are trained to minimize the cross-entropy loss:

$$\mathcal{L}(\gamma) = \sum_{x_{[i,j]} \in \mathcal{Q}_{train}} -\log p(y_{[i,j]}; x_{[i,j]}). \tag{11}$$

Inference During inference time, given a novel episode $(\mathcal{S}_{new}, \mathcal{Q}_{new}, \mathcal{Y}_{new})$ for inference, we first leverage the learned model to compute prototypes for all $y_k \in \mathcal{Y}_{new}$ on $\mathcal{S}_{new}$. Then, upon the mention detection model, we inference the entity class for each detected entity span $x_{[i,j]}$ in $\mathcal{Q}_{new}$ by taking the label $y_k \in \mathcal{Y}_{new}$ with the highest probability in Eq. (10):

$$\hat{y}_{[i,j]} = \arg\max_{y_k} p(y_k; x_{[i,j]}). \tag{12}$$

3.2.2 MAML Enhanced ProtoNet

Here, we elaborate on the procedure to integrate the ProtoNet and the model-agnostic meta-learning.

Meta-Training Given a randomly sampled episode $(\mathcal{S}^{(i)}_{train}, \mathcal{Q}^{(i)}_{train}, \mathcal{Y}^{(i)}_{train})$ from $\mathcal{E}_{train}$, for *inner-update*, we first compute prototypes for each entity class in $\mathcal{Y}_{train}$ using $\mathcal{S}^{(i)}_{train}$ via Eq. (9), and then take each span $x_{[i,j]} \in \mathcal{S}^{(i)}_{train}$ as the query item in conventional ProtoNet for gradient update:

$$\gamma'_i = U^n(\gamma; \alpha, \mathcal{S}^{(i)}_{train}), \tag{13}$$

where U^n denotes n-step gradient updates with the learning rate α to minimize the cross-entropy loss $\mathcal{L}(\gamma; \mathcal{S}^{(i)}_{train})$ as in Eq. (11).

As for *meta-update*, we first re-compute prototypes for each entity class in $\mathcal{Y}^{(i)}_{train}$ with γ', *i.e.*, the model parameters obtained from *inner-update*. After that, we perform *meta-update* by evaluating γ' on the query set $\mathcal{Q}^{(i)}_{train}$. We employ the first-order approximation again for computational efficiency. When aggregating gradients from multiple episodes, it could be formulated as:

$$\gamma \leftarrow \gamma - \beta \sum_i \nabla_{\gamma'_i} \mathcal{L}(\gamma'_i; \mathcal{Q}^{(i)}_{train}), \qquad (14)$$

Meta-Testing Given $(\mathcal{S}_{new}, \mathcal{Q}_{new}, \mathcal{Y}_{new})$, a novel episode unseen during training, conventional ProtoNet directly adopts the meta-trained model to compute prototypes with $\mathcal{S}_{new}$, and then inference on $\mathcal{Q}_{new}$. Here, we first take the support examples from $\mathcal{S}_{new}$ to fine-tune the meta-learned model γ^* for a few steps in a way the same as Eq. (13), however, the loss is computed on $\mathcal{S}_{new}$. Then, we leverage $\mathcal{S}_{new}$ again to compute prototypes with the fine-tuned model, and further inference the entity class for each detected span in $\mathcal{Q}_{new}$ as in Eq. (12).

4 Experiments

4.1 Settings

4.1.1 Datasets

We conduct experiments to evaluate the proposed approach on two groups of datasets.

Few-NERD (Ding et al., 2021). It is annotated with a hierarchy of 8 coarse-grained and 66 fine-grained entity types. Two tasks are considered on this dataset: i) **Intra**, where all entities in train/dev/test splits belong to different coarse-grained types. ii) **Inter**, where train/dev/test splits may share coarse-grained types while keeping the fine-grained entity types mutually disjoint. [3]

Cross-Dataset (Hou et al., 2020). Four datasets focusing on four domains are used here: CoNLL-2003 (Tjong Kim Sang, 2002) (news), GUM (Zeldes, 2017) (Wiki) , WNUT-2017 (Derczynski et al., 2017) (social), and Ontonotes (Pradhan et al., 2013) (mixed). We take two domains for training, one for validation, and the remaining for test. For fair comparison, we directly use sampled episodes by Hou et al. (2020). For more details of these datasets, please refer to the Appendix A.2.

4.1.2 Evaluation

For evaluation on **Few-NERD**, we employ episode evaluation as in Ding et al. (2021) and calculate the precision (P), recall (R), and micro F1-score (F1) over all test episodes. For evaluation on **Cross-Dataset**, we calculate P, R, F1 within each episode and then average over all episodes as in Hou et al. (2020). For all results, we report the mean and standard deviation based on 5 runs with different seeds.

4.1.3 Implementation Details

We implement our approach with PyTorch 1.9.0[4]. We leverage two separate BERT models for f_θ in Eq. (1) and g_γ in Eq. (7), respectively. Following previous methods (Hou et al., 2020; Ding et al., 2021), we use the BERT-base-uncased model (Devlin et al., 2019). The parameters of the embedding layer are frozen during optimization. We train all models for 1,000 steps and choose the best model with the validation set. We use a batch size of 32, maximum sequence length of 128, and a dropout probability of 0.2. For the optimizers, we use AdamW (Loshchilov and Hutter, 2019) with a 1% linearly scheduled warmup. We perform grid search for other hyper-parameters and select the best settings with the validation set. For more details, please refer to the Appendix A.3.

4.2 Main Results

Baselines For FewNERD, we compare the proposed approach to ESD (Wang et al., 2021a), CONTAINER (Das et al., 2021), and methods from Ding et al. (2021), *e.g.*, ProtoBERT, StructShot, *etc*. For Corss-Dataset, we compare our method to L-TapNet+CDT (Hou et al., 2020) and other baselines from Hou et al. (2020), *e.g.*, TransferBERT, Matching Network, *etc*. Please refer to the Appendix A.4 for more details about baselines.

Results Table 2 and Table 3 report the results of our approach alongside those reported by previous

[3] https://github.com/thunlp/Few-NERD

[4] https://pytorch.org/

[5] To make fair comparison with CONTAINER (Das et al., 2021) and ESD (Wang et al., 2021a), we use the data from https://cloud.tsinghua.edu.cn/f/8483dc1a34da4a34ab58/?dl=1, which corresponds to the results reported in https://arxiv.org/pdf/2105.07464v5.pdf. For results of our approach on data from https://cloud.tsinghua.edu.cn/f/0e38bd108d7b49808cc4/?dl=1, please refer to our Github.

Models	Intra				Inter			
	1∼2-shot		5∼10-shot		1∼2-shot		5∼10-shot	
	5 way	10 way	5 way	10 way	5 way	10 way	5 way	10 way
ProtoBERT†	23.45±0.92	19.76±0.59	41.93±0.55	34.61±0.59	44.44±0.11	39.09±0.87	58.80±1.42	53.97±0.38
NNShot†	31.01±1.21	21.88±0.23	35.74±2.36	27.67±1.06	54.29±0.40	46.98±1.96	50.56±3.33	50.00±0.36
StructShot†	35.92±0.69	25.38±0.84	38.83±1.72	26.39±2.59	57.33±0.53	49.46±0.53	57.16±2.09	49.39±1.77
CONTAINER (Das et al., 2021)	40.43	33.84	53.70	47.49	55.95	48.35	61.83	57.12
ESD (Wang et al., 2021a)	41.44±1.16	32.29±1.10	50.68±0.94	42.92±0.75	66.46±0.49	59.95±0.69	**74.14±0.80**	67.91±1.41
Ours	**52.04±0.44**	**43.50±0.59**	**63.23±0.45**	**56.84±0.14**	**68.77±0.24**	**63.26±0.40**	71.62±0.16	**68.32±0.10**

Table 2: F1 scores with standard deviations on Few-NERD for both inter and intra settings. † denotes the results reported in Ding et al. (2021).[5] The best results are in **bold**.

Models	1-shot				5-shot			
	News	Wiki	Social	Mixed	News	Wiki	Social	Mixed
TransferBERT‡	4.75±1.42	0.57±0.32	2.71±0.72	3.46±0.54	15.36±2.81	3.62±0.57	11.08±0.57	35.49±7.60
SimBERT‡	19.22±0.00	6.91±0.00	5.18±0.00	13.99±0.00	32.01±0.00	10.63±0.00	8.20±0.00	21.14±0.00
Matching Network‡	19.50±0.35	4.73±0.16	17.23±2.75	15.06±1.61	19.85±0.74	5.58±0.23	6.61±1.75	8.08±0.47
ProtoBERT‡	32.49±2.01	3.89±0.24	10.68±1.40	6.67±0.46	50.06±1.57	9.54±0.44	17.26±2.65	13.59±1.61
L-TapNet+CDT (Hou et al., 2020)	44.30±3.15	12.04±0.65	20.80±1.06	15.17±1.25	45.35±2.67	11.65±2.34	23.30±2.80	20.95±2.81
Ours	**46.09±0.44**	**17.54±0.98**	**25.14±0.24**	**34.13±0.92**	**58.18±0.87**	**31.36±0.91**	**31.02±1.28**	**45.55±0.90**

Table 3: F1 scores with standard deviations on Cross-Dataset. ‡ denotes the results reported in Hou et al. (2020). The best results are in **bold**.

state-of-the-art methods.[6] It can be seen that our proposed method outperforms the prior methods with a large margin, achieving an performance improvement up to 10.60 F1 scores on Few-NERD (*Intra*, 5way 1∼2 shot) and 19.71 F1 scores on Cross-Dataset (*Wiki*, 5-shot), which well demonstrates the effectiveness of the proposed approach. Table 2 and Table 3 also depict that compared with the results of Few-NERD *Inter*, where the training episodes and test episodes may be constructed with the data from the same domain while still focusing on different fine-grained entity classes, our approach attains more impressive performance in other settings where exists larger transfer gap, *e.g.*, transferring across different coarse entity classes even different datasets built from different domains. This suggests that our approach is good at dealing with difficult cases, highlighting the necessity of exploring information contained in target-domain support examples and the strong adaptation ability of our approach.

4.3 Ablation Study

To validate the contributions of different components in the proposed approach, we introduce the following variants and baselines for ablation study: *1) Ours w/o MAML*, where we train both the mention detection model and the ProtoNet in a conventional supervised manner and then fine-tune with few-shot examples. *2) Ours w/o Span Detector*, where we remove the mention detection step and integrate MAML with token-level prototypical networks. *3) Ours w/o Span Detector w/o MAML*, where we further eliminate the meta-learning procedure from *Ours w/o Span Detector*, and thus becomes the conventional token-level prototypical networks. *4) Ours w/o ProtoNet*, where we directly apply the original MAML algorithm to train a BERT-based tagger for few-shot NER.

	Intra	Inter
Ours	**52.04**	**68.77**
1) Ours w/o MAML	48.76	64.44
2) Ours w/o Span Detector	36.06	53.56
3) Ours w/o Span Detector w/o MAML	23.45	44.44
4) Ours w/o ProtoNet	21.20	45.71

Table 4: Ablation study: F1 scores on Few-NERD 5-way 1∼2-shot are reported.

Table 4 highlights the contributions of each component in our proposed approach. Generally speaking, removing any of them will generally lead to a performance drop. Moreover, we can draw some in-depth observations as follows. 1) *Ours* outperforms *Ours w/o MAML* and *Ours w/o Span Detector* outperforms *Ours w/o Span Detector w/o MAML* in-

[6] We also provide the intermediate results, *i.e.*, F1-scores of entity span detection in the Appendix A.5.

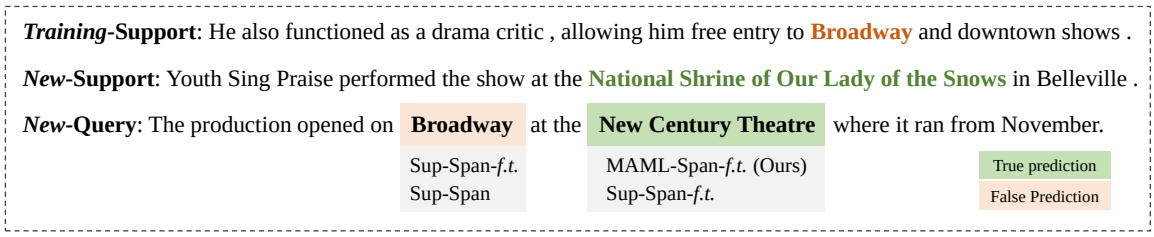
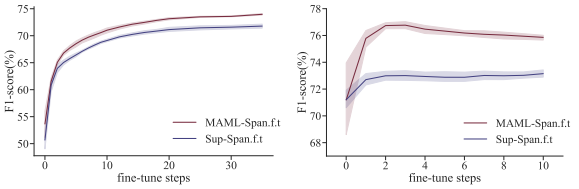


Figure 2: Case study of span detection. **Sup-Span**: train a span detector in the fully supervised manner on available data from all training episodes, and then directly use it for span detection. **Sup-Span-f.t.**: further fine-tune the model learned by *Sup-Span* as in the proposed approach.

dicate that exploring information contained in support examples with the proposed meta-learning procedure does bring performance gain for few-shot transfer. 2) *Ours* outperforms *Ours w/o Span Detector* and *Ours w/o MAML* outperforms *Ours w/o Span Detector w/o MAML* demonstrate the essentiality of the decomposed framework (*i.e.*, mention detection and entity typing) to mitigate the problem of noisy prototype for non-entities. 3) Though MAML plays an important role in learning from few-shot support examples, *Ours w/o ProtoNet*, which requires the model to adapt the up-most classification layer without sharing knowledge with training episodes leads to unsatisfactory results, verifying the reasonableness and the effectiveness of our decomposed meta-learning procedure.

How does MAML promote the span detector? To bring up insights on how MAML promotes the span detector, here we introduce two baselines and compare them to our approach by case study. As shown in Figure 2, given a query sentence from a novel episode, *Sup-Span* only predicts a false positive span "Broadway" while missing the golden span "New Century Theatre". Note that "Broadway" appears in training corpus as an entity span, indicating that the span detector trained in a fully supervised manner performs well on seen entity spans, but struggles to detect un-seen entity spans. Figure 2 also shows that both our method and *Sup-Span-f.t.* can successfully detect "New Century Theatre". However, *Sup-Span-f.t.* still outputs "Broadway" while our method can produce more accurate predictions. This shows that though fine-tuning can benefit full supervised model on new entity classes to some extend, it may bias too much to the training data.

We further investigate how performances of aforementioned span detectors vary with different fine-tune steps. As shown in Figure 3, our



(a) Intra (b) Inter

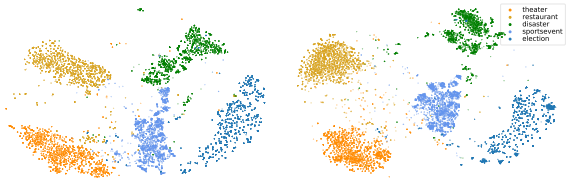
Figure 3: F1 scores of differently trained span detectors *w.r.t.* fine-tune steps on Few-NERD 5-way 1∼2-shot test set. The light-colored area indicates the range of results obtained from multiple random seeds.

	Intra	Inter
Ours (w/ MAML-ProtoNet)	**52.04**	**68.77**
Ours w/ ProtoNet	50.53	67.79

Table 5: Analysis on entity typing under Few-NERD 5-way 1∼2-shot setting. F1 scores are reported. **Ours w/ ProtoNet**: built upon the same span detection model as *Ours*, directly leverage ProtoNet for inference.

model (*MAML-Span-f.t.*) consistently outperforms *Sup-Span-f.t.*, suggesting that the proposed meta-learning procedure could better leverage support examples from novel episodes and meanwhile, help the model adapt to new episodes more effectively.

How does MAML enhance the ProtoNet? We first compare the proposed MAML-Proto to the conventional ProtoNet based on the same span detector proposed in this paper. Table 5 shows that our MAML-ProtoNet achieves superior performance than the conventional ProtoNet, which verifies the effectiveness of leveraging the support examples to refine the learned embedding space at test time. To further analyze how MAML adjusts the representation space of entity spans and prototypes, we utilize *t-SNE* (van der Maaten and Hinton, 2008) to reduce the dimension of span representations obtained from ProtoNet and MAML-ProtoNet



(a) Proto (b) MAML-Proto

Figure 4: t-SNE visualization of span representations for entity typing on Few-NERD *Intra*, 5-way 5∼10-shot dev set. The representations are obtained from BERT trained with ProtoNet, and our MAML enhanced ProtoNet respectively.

for entity typing, the visualization is shown in Figure 4. We can see that MAML enhanced Proto can cluster span representations of the same entity class while dispersing span representations of different entity classes . Therefore, compared with ProtoNet, it is easier for the proposed MAML-ProtoNet to assign an entity class for a query span by measuring similarities between its representation and the prototype of each entity class.

5 Related Work

Neural NER Modern NER systems usually formulate the NER task as a sequence labeling problem and tackle it by implementing deep neural networks and a token-level classification layer with a conditional random field (Lafferty et al., 2001, CRF) layer on top (Ma and Hovy, 2016; Chiu and Nichols, 2016; Liu et al., 2019; Devlin et al., 2019). Alternative approaches for NER are also proposed to handle the problem based on span classification (Ouchi et al., 2020; Fu et al., 2021), machine reading comprehension (Li et al., 2020b), and sequence generation (Yan et al., 2021).

Few-Shot Learning and Meta-Learning Recently, few-shot learning has received increasing attention in the NLP community (Han et al., 2018; Geng et al., 2019; Chen et al., 2019; Brown et al., 2020; Schick and Schütze, 2021; Gao et al., 2021). and meta-learning has become a popular paradigm for few-shot settings. Typical meta-learning approaches can be divided into three categories: black-box adaption based methods (Santoro et al., 2016), optimization based methods (Finn et al., 2017), and metric learning based methods(Vinyals et al., 2016; Snell et al., 2017). Our work takes advantages of two popular meta-learning approaches, *i.e.*, prototypical network (Snell et al., 2017) and MAML (Finn et al., 2017). The most related work of this paper is Triantafillou et al. (2020), which similarly implements MAML updates over prototypical networks for few-shot image classification.

Few-Shot NER Studies on few-shot NER typically adopt metric learning based approaches at either token-level (Fritzler et al., 2019; Hou et al., 2020; Yang and Katiyar, 2020; Tong et al., 2021) or span-level (Yu et al., 2021; Wang et al., 2021a). Athiwaratkun et al. (2020) and Cui et al. (2021) also propose to address the problem via sequence generation and adapt the model to a new domain within the conventional transfer learning paradigm (training plus finetuning). Differently, Wang et al. (2021b) propose to decompose the problem into span detection and entity type classification to better leverage type description. They exploit a traditional span-based classifier to detect entity spans and leverage class descriptions to learn representations for each entity class. When adapting the model to new domains in the few-shot setting, they directly fine-tune the model with the support examples. In this paper, we propose a decomposed meta-learning based method to handle few-shot span detection and few-shot entity typing sequentially for few-shot NER. The contribution and novelty of our work lie in that: i) Previous work transfers the metric-learning based model learned in source domains to a novel target domain either without any parameter updates (Hou et al., 2020; Wang et al., 2021a) or by simply applying conventional fine-tuning (Cui et al., 2021; Das et al., 2021; Wang et al., 2021b), while we introduce the model-agnostic meta-learning and integrate it with the prevalent prototypical networks to leverage the information contained in support examples more effectively. ii) Existing studies depend on one (Hou et al., 2020) or multiple prototypes (Tong et al., 2021; Wang et al., 2021a) to represent text spans of non-entities ("O") for class inference, while we avoid this problem by only locating named entities during span detection. Moreover, meta-learning has also been exploited in a few recent studies (Li et al., 2020a; de Lichy et al., 2021) for few-shot NER. However, our work substantially differs from them in that we proposed a decomposed meta-learning procedure to separately optimize the span detection model and the entity typing model.

6 Conclusion

This paper presents a decomposed meta-learning method for few-shot NER problem, *i.e.*, sequentially tackle few-shot span-detection and few-shot entity typing using meta-learning. We formulate the few-shot span detection as a sequence labeling problem and employ MAML to learn a good parameter initialization, which enables the model to fast adapt to novel entity classes by fully exploring information contained in support examples. For few-shot entity typing, we propose MAML-ProtoNet, which can find a better embedding space than conventional ProtoNet to represent entity spans from different classes more distinguishably, thus making more accurate predictions. Extensive experiments on various benchmarks show that our approach achieves superior performance over prior methods.

References

Ben Athiwaratkun, Cicero Nogueira dos Santos, Jason Krone, and Bing Xiang. 2020. Augmented natural language for generative sequence labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 375–385, Online. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Mingyang Chen, Wen Zhang, Wei Zhang, Qiang Chen, and Huajun Chen. 2019. Meta relational learning for few-shot link prediction in knowledge graphs. 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 4217–4226, Hong Kong, China. Association for Computational Linguistics.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

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

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

Cyprien de Lichy, Hadrien Glaude, and William Campbell. 2021. Meta-learning for few-shot named entity recognition. In *Proceedings of the 1st Workshop on Meta Learning and Its Applications to Natural Language Processing*, pages 44–58, Online. Association for Computational Linguistics.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147, Copenhagen, Denmark. Association for Computational Linguistics.

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

Ning Ding, 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 *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3198–3213, Online. Association for Computational Linguistics.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

G David Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Alexander Fritzler, Varvara Logacheva, and Maksim Kretov. 2019. Few-shot classification in named entity recognition task. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 993–1000.

Jinlan Fu, Xuanjing Huang, and Pengfei Liu. 2021. SpanNER: Named entity re-/recognition as span prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7183–7195, Online. Association for Computational Linguistics.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. 2019. Induction networks for few-shot text classification. 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 3904–3913, Hong Kong, China. Association for Computational Linguistics.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4803–4809, Brussels, Belgium. Association for Computational Linguistics.

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1381–1393, Online. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 282–289. Morgan Kaufmann.

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

J. Li, B. Chiu, S. Feng, and H. Wang. 2020a. Few-shot named entity recognition via meta-learning. *IEEE Transactions on Knowledge & Data Engineering*.

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

Tianyu Liu, Jin-Ge Yao, and Chin-Yew Lin. 2019. Towards improving neural named entity recognition with gazetteers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5301–5307, Florence, Italy. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

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

Hiroki Ouchi, Jun Suzuki, Sosuke Kobayashi, Sho Yokoi, Tatsuki Kuribayashi, Ryuto Konno, and Kentaro Inui. 2020. Instance-based learning of span representations: A case study through named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6452–6459, Online. Association for Computational Linguistics.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1765, Vancouver, Canada. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1842–1850. JMLR.org.

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, pages 2339–2352, Online. Association for Computational Linguistics.

Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4077–4087.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Meihan Tong, Shuai Wang, Bin Xu, Yixin Cao, Minghui Liu, Lei Hou, and Juanzi Li. 2021. Learning from miscellaneous other-class words for few-shot named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6236–6247, Online. Association for Computational Linguistics.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. 2020. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3630–3638.

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

Yaqing Wang, Haoda Chu, Chao Zhang, and Jing Gao. 2021b. Learning from language description: Low-shot named entity recognition via decomposed framework. In *Findings of the Association for Computational Linguistics: EMNLP 2021*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Qianhui Wu, Zijia Lin, Guoxin Wang, Hui Chen, Börje F. Karlsson, Biqing Huang, and Chin-Yew Lin. 2020. Enhanced meta-learning for cross-lingual named entity recognition with minimal resources. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9274–9281. AAAI Press.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various NER subtasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5808–5822, Online. Association for Computational Linguistics.

Yi Yang and Arzoo Katiyar. 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6365–6375, Online. Association for Computational Linguistics.

Sung Whan Yoon, Jun Seo, and Jaekyun Moon. 2019. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7115–7123. PMLR.

Dian Yu, Luheng He, Yuan Zhang, Xinya Du, Panupong Pasupat, and Qi Li. 2021. Few-shot intent classification and slot filling with retrieved examples. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 734–749, Online. Association for Computational Linguistics.

Amir Zeldes. 2017. The gum corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

A Appendix

A.1 Meta learning

The goal of meta-learning is to learn to fast adapt to a new few-shot task that is never-seen-before. To train a meta-learning model, a large number of episodes $\mathcal{T}_{train}$ (few-shot tasks) are constructed from training data D_{train}, which usually follows the N-way K-shot task formulation and are used to train the meta-learning model. One episode contains a small training set $\mathcal{S}_{train}$, called support set, and a test set $\mathcal{Q}_{train}$, called query set. The meta-learner generates a task-specific model for a new task $\mathcal{T}_i$ via updating on support set $\mathcal{S}_{train}$, then the task-specific model is tested on $\mathcal{Q}_{train}$ to get a test error. The meta-learner then learns to learn new tasks by considering how to reduce the test error on $\mathcal{Q}_{train}$ by updating on $\mathcal{S}_{train}$. To evaluate the task learning ability of a meta-learner, a bunch of episodes $\mathcal{T}_{test}$ are constructed from the normal test data D_{test}, and the expectation of performance on $\mathcal{Q}_{test}$ from all test episodes is severed as evaluation protocol. To distinguish the training phase of meta-learner on episodes $\mathcal{T}_{train}$ and training of a task-specific model on support set $\mathcal{S}$, the former is called meta-training and the latter is called training. Similarly, the testing of a meta-learner on $\mathcal{T}_{test}$ is called meta-testing, and the evaluating of a task-specific model on query set $\mathcal{Q}$ is called testing.

A.2 Datasets

Table A.1 shows the dataset statistics of original data for constructing few-shot episodes.

Dataset	Domain	# Sentences	# Classes
Few-NERD	Wikipedia	188.2k	66
CoNLL03	News	20.7k	4
GUM	Wiki	3.5k	11
WNUT	Social	5.6k	6
OntoNotes	Mixed	159.6k	18

Table A.1: Evaluation dataset statistics

For Few-NERD, we use episodes released by Ding et al. (2021)[7] which contain 20,000 episodes for training, 1,000 episodes for validation, and 5,000 episodes for testing. Each episode is an N-way K∼2K-shot few-shot task. As for Cross-Dataset, two datasets are used for constructing training episodes, one dataset is used for validation, and episodes from the remained dataset are used for evaluation. We use public episodes[8] constructed by Hou et al. (2020). For 5shot, 200 episodes are used for training, 100 episodes for validation, and 100 for testing. For the 1shot experiment, 400/100/200 episodes are used for training/validation/testing, except for experiments on OntoNotes(Mixed), where 400/200/100 episodes are constructed for train/dev/test.

A.3 Additional Implementation Details

Parameter Setting We use `BERT-base-uncased` from Huggingface Library (Wolf et al., 2020) as our base encoder following Ding et al. (2021). We use AdamW (Loshchilov and Hutter, 2019) as our optimizer with a learning rate of 3e-5 and 1% linear warmup steps at both the meta-training and finetuning in meta-testing time for all experiments. The batch size is set to 32, the max sequence length is set to 128 and we keep dropout rate as 0.1. At meta-training phase, the inner update step is set to 2 for all experiments. When finetuning the span detector at meta-testing phase, the finetune step is set to 3 for all inter settings on Few-NERD dataset and 30 for other experiments. For entity typing, the finetune step at meta-testing phase is set to 3 for all experiments on Few-NERD dataset, 20 for all Cross-Dataset experiments. To further boost the performance, we only keep entities that have a similarity score with its nearest prototype greater than a threshold of 2.5. We set max-loss coefficient λ as 2 at meta-training query set evaluation phase, 5 at other phases. We validate our model on dev set every 100 steps and select the checkpoint with best f1 score performance on dev set within the max train steps 1,000. We use grid search for hyperparameter setting, the search space is shown in Table A.2. The total model has 196M parameters and trains in ≈60min on a Tesla V100 GPU.

Learning rate	{1e-5, 3e-5, 1e-4}
Meta-test fine-tune steps	{3, 5, 10, 20, 30}
Max-loss coefficient λ	{0, 1, 2, 5, 10}
Type similarity threshold	{1, 2.5, 5}
Mini-batch size	{16, 32}

Table A.2: Hyper-parameters search space used in our experiments.

[7] https://ningding97.github.io/fewnerd/

[8] https://github.com/AtmaHou/FewShotTagging

A.4 Baselines

We consider the following metric-learning based baselines:

SimBERT (Hou et al., 2020) applies BERT without any finetuning as the embedding function, then assign each token's label by retrieving the most similar token in the support set .

ProtoBERT (Fritzler et al., 2019) uses a token-level prototypical network (Snell et al., 2017) which represents each class by averaging token representation with the same label, then the label of each token in the query set is decided by its nearest class prototype.

MatchingBERT (Vinyals et al., 2016) is similar to ProtoBERT except that it calculates the similarity between query instances and support instances instead of class prototypes.

L-TapNet+CDT (Hou et al., 2020) enhances TapNet (Yoon et al., 2019) with pair-wise embedding, label semantic, and CDT transition mechanism.

NNShot (Yang and Katiyar, 2020) pretrains BERT for token embedding by conventional classification for training, a token-level nearest neighbor method is used at testing.

StructShot (Yang and Katiyar, 2020) improves NNshot by using an abstract transition probability for Viterbi decoding at testing.

ESD (Wang et al., 2021a) is a span-level metric learning based method. It enhances prototypical network by using inter- and cross-span attention for better span representation and designs multiple prototypes for O label.

Besides, we also compare with the finetune-based methods:

TransferBERT (Hou et al., 2020) trains a token-level BERT classifier, then finetune task-specific linear classifier on support set at test time.

CONTAINER (Das et al., 2021) uses token-level contrastive learning for training BERT as token embedding function, then finetune the BERT on support set and apply a nearest neighbor method at inference time.

A.5 Results of Span Detection

Table A.3 and Table A.4 show the performance of our span detection module on Few-NERD and Cross-Dataset.

Models	1∼2-shot		5∼10-shot	
	5 way	10 way	5 way	10 way
Intra	73.69±0.14	74.32±1.84	77.76±0.24	78.66±0.15
Inter	76.71±0.30	76.63±0.24	75.97±0.14	76.62±0.11

Table A.3: F1 scores of our entity span detection module on **Few-NERD** for both inter and intra settings.

Models	News	Wiki	Social	Mixed
1-shot	65.06±0.91	35.63±2.17	38.89±0.55	46.52±1.24
5-shot	74.20±0.33	46.26±1.28	43.16±1.23	54.70±0.88

Table A.4: F1 scores of our entity span detection module on **Cross-Dataset**.