Type-Aware Decomposed Framework for Few-Shot Named Entity Recognition

Anonymous ACL submission

Abstract

Despite the recent success achieved by several two-stage prototypical networks in few-shot named entity recognition (NER) task, *the over-detected false spans* at span detection stage and *the inaccurate and unstable prototypes* at type classification stage remain to be challenging problems. In this paper, we propose a novel **T**ype-**A**ware **D**ecomposed framework, namely TadNER, to solve these problems. We first present *a type-aware span filtering strategy* to filter out false spans by removing those semantically far away from type names. We then present *a type-aware contrastive learning strategy* to construct more accurate and stable prototypes by jointly exploiting support samples and type names as references. Extensive experiments on various benchmarks prove that our proposed TadNER framework yields a new state-of-the-art performance [1].

1 Introduction

Named entity recognition (NER) aims to detect entity spans and classify them into pre-defined categories (entity types). When there are sufficient labeled data, deep learning-based methods (Huang et al., 2015; Ma and Hovy, 2016; Lample et al., 2016; Chiu and Nichols, 2016) can get impressive performance. In real applications, it is desirable to recognize new categories which are unseen in training/source domain. However, collecting extra labeled data for these new types will be surely time-consuming and labour-expensive. Consequently, few-shot NER (Fritzler et al., 2019; Yang and Katiyar, 2020), which involves identifying unseen entity types based on a few labeled samples for each class (i.e., *support samples*) in test domain, has attracted great research interests in recent years.

End-to-end metric learning based methods are the mainstream in few-shot NER (Yang and Katiyar, 2020; Das et al., 2022). These methods need to simultaneously learn the complex structure consisting of entity boundary and entity type. When the domain gap is large, their performance will drop dramatically because it is extremely hard to capture such complicated structure information with only a few support examples for domain adaptation. This leads to the insufficient learning of boundary information, resulting that these methods often misclassify entity boundaries and cannot obtain very satisfying performance.

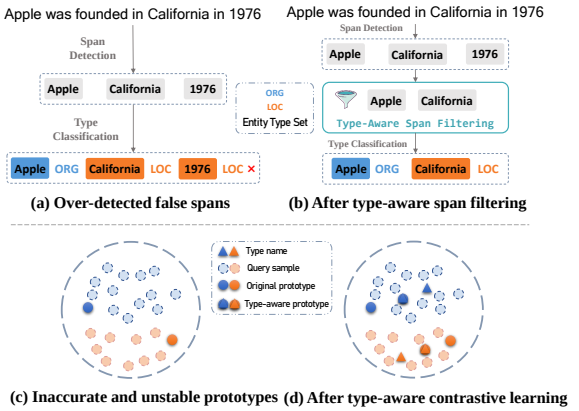


Figure 1: (a) shows over-detected false spans, (b) shows spans got by adopting our type-aware span filtering strategy. (c) shows inaccurate and unstable prototypes, (d) shows prototypes got by adopting our type-aware contrastive learning strategy.

Recently, there is an emerging trend in adopting two-stage prototypical networks (Wang et al., 2022; Ma et al., 2022c) for few-shot NER, which decompose NER into two separate *span extraction* and *type classification* tasks and perform one task at each stage. Since the decomposed methods only need to handle one single boundary detection task at the first stage, they can find more accurate boundaries and obtain better performance than end-to-end approaches.

While making good progress, these two-stage prototypical networks still face two challenging problems, i.e., *the over-detected false spans* and *the inaccurate and unstable prototypes* in correspond-

[1] Our code and data will be available at https://github.com/ANONYMOUS/TadNER.

ing stages. (1) The decomposed approaches usually recall at the span extraction stage in test phase many over-detected false spans whose types only exist in the source domain. For example, "1976" in Fig. 1 (a) belongs to a DATE type in the source domain since there are many samples like "Obama was born in 1961" in training, and thus it is easily recognized as a span by the span detector. However, there is no such label in the test domain and "1976" is thus assigned a false LOC type. (2) The prototypical networks in decomposed methods target at learning a type-agnostic metric similarity function to classify entities in test samples (*i.e., query samples*) via their distance to prototypes. Since the prototypes are constructed using very few support samples in the type-agnostic feature space, they might be inaccurate and unstable. For example, in Fig. 1 (c), a prototype is just the support sample in one-shot NER and thus deviates far away from the real class center.

Based on the above observations, we propose a **T**ype-**A**ware **D**ecomposed framework, namely TadNER, for few-shot **NER**. Our method follows the span detection and type classification learning scheme in the decomposed framework but moves two steps further to overcome the aforementioned issues.

Firstly, we present *a type-aware span filtering strategy* to filter out false spans by removing those semantically far away from type names [2]. By this means, the over-detected spans like "1976" whose types do not exist in test domain can be removed due to the long semantic distance to type names, as shown in Fig. 1 (b).

Secondly, we present *a type-aware contrastive learning strategy* to construct more accurate and stable prototypes by jointly leveraging type names and support samples as references. Through this way, the type names can serve as the guidance for prototypes and make them not deviate too far away from the class centers even in some extreme outlier cases, as shown in Fig. 1 (d).

Extensive experimental results on 5 benchmark datasets demonstrate the superiority of our TadNER over the state-of-the-art decomposed methods. In particular, in the hard intra Few-NERD and 1-shot Domain Transfer settings, TadNER achieves a 7% and 8% absolute F1 increase, respectively.

[2]Note that though the type assignments are unknown in few-shot NER, the type names (labels) in the test domain are provided.

2 Related Work

2.1 Few-Shot NER

Existing few-shot NER methods can be roughly categorized into two types: prompt-based and metric-based. The first type mainly focuses on exploring the general pre-trained language model knowledge for NER via prompt learning (Cui et al., 2021; Ma et al., 2022b; Huang et al., 2022; Lee et al., 2022). This type of methods rely heavily on the quantity of templates, prompts, or good examples. The second type expects to learn a feature space with good generalizability in the source domain and then classifies test samples via nearest class prototypes (Snell et al., 2017; Fritzler et al., 2019; Ji et al., 2022) or neighbor samples (Yang and Katiyar, 2020; Das et al., 2022).

There are also some efforts to improve few-shot NER performance by leveraging the type name (label) semantics (Hou et al., 2020; Wang et al., 2021; Ma et al., 2022a). These methods usually treat labels as class representatives and align tokens with them, yet neglecting the joint training of entity words and label representations. Hence they can only use either support sets or labels as class references. Instead, our method can exploit support samples and type names at the same time, which helps construct more accurate and stable prototypes in the target domain.

2.2 Task Decomposition and Contrastive Learning

Recently, several decomposed-based methods are proposed to solve NER problem (Shen et al., 2021; Wang et al., 2021; Zhang et al., 2022; Wang et al., 2022; Ma et al., 2022c). These methods can learn entity boundary information well in data-limited scenarios and often get better results. However, the widely used prototypical networks in these methods may encounter inaccurate and unstable prototypes given limited support samples at the type classification stage. Besides, they may face the problem of over-detected false spans produced at the span detection stage. Our method can address these two issues via the proposed type-aware contrastive learning and type-aware span filtering strategies.

Our method is also inspired by the idea of contrastive learning (Chen et al., 2020; Khosla et al., 2020). Due to its good generalization performance, two recent methods (Das et al., 2022; Huang et al., 2022) borrow this idea for few-shot NER, which construct contrastive loss between tokens or be-

tween the token and the prompt. However, they are both the end-to-end approach and thus have the inherent drawback that cannot learn good entity boundary information. In contrast, our method is a decomposed one and our contrastive loss is constructed between tokens with additional type name information, which can find accurate boundary and learn a type-aware feature space.

3 Method

In this section, we detail our TadNER framework consisting of *a span detection stage* and *a type classification stage*. We first train a pre-trained language model (PLM) with a classification layer to detect entity spans as existing decomposed methods (Ma et al., 2022c) do. We then address two challenging problems unsolved by previous methods. (1) For the problem of inaccurate and unstable prototypes, we train the pre-trained language model via *a type-aware contrastive learning strategy* which can help construct type-aware high-quality prototypes. (2) For the problem of over-detected false spans, we introduce *a type-aware span filtering strategy* to remove false spans whose types only exist in the source domain.

Note that for few-shot NER, both the training and inference in source and test domains need two stages, where the type-aware contrastive learning and type-aware span filtering strategies take effect at the type classification stage in the training and test domain, respectively. The overall structure of our TadNER is shown in Figure 2.

3.1 Task Formulation

Given a sequence $X = \{x_1, x_2, ..., x_N\}$ with N tokens, NER aims to assign each token x_i a corresponding label $y_i \in \mathcal{T} \cup \{\text{O}\}$, where $\mathcal{T}$ is the entity type set and O denotes the non-entity label.

For few-shot NER, a model is trained in a source domain dataset $\mathcal{D}_{source}$ with the entity type set $\mathcal{T}_{source} = \{t_1, t_2, ...t_m\}$. The model is then fine-tuned in a test/target domain datset $\mathcal{D}_{target}$ with the entity type set $\mathcal{T}_{target} = \{t_1, t_2, ...t_n\}$ using a given support set $\mathcal{S}_{target}$. The entity token set and corresponding label set in $\mathcal{S}_{target}$ are denoted as $E^s = \{e_1^s, e_2^s, ..., e_M^s\}$ and $Y^s = \{y_1^s, y_2^s, ..., y_M^s\}$, where $y_i^s \in \mathcal{T}_{target}$ is the label and M is the number of entity tokens. The model is supposed to recognize entities in the query set $\mathcal{Q}_{target}$ of the target domain. Since $\mathcal{T}_{source} \cap \mathcal{T}_{target} = \emptyset$, it is very challenging to learn a generalized NER model to recognize unseen class examples.

More specifically, in the n-way k-shot setting, there are n labels in $\mathcal{T}_{target}$ and k examples associated with each label in the support set $\mathcal{S}_{target}$.

3.2 Source Domain Training

The source domain training consists of span detection and type classification stages. The procedure is shown in Fig. 2 (a).

3.2.1 Span Detection

We adopt sequence labeling for the first stage in training, which is as same as an existing decomposed method (Ma et al., 2022c) for few-shot NER.

We use BERT (Devlin et al., 2019) as our PLM encoder f_{θ_1} with parameters θ_1 in our method. Given an input sentence $X = \{x_1, x_2, ..., x_N\}$, the encoder produces contextualized representations for each token as:

$$\mathbf{H} = [\mathbf{h_1}, ..., \mathbf{h_N}] = f_{\theta_1}([x_1, ..., x_N]), \quad (1)$$

where $\mathbf{H} \in \mathbb{R}^{N*r}$ [3]. $\mathbf{H}$ is then fed into a classification layer consisting of a dropout layer (Srivastava et al., 2014) and a linear layer to get the probability distribution $\mathbf{P} = [\mathbf{p}(\mathbf{x_1}), ..., \mathbf{p}(\mathbf{x_N})]$ ($\mathbf{p}(\mathbf{x_i}) \in \mathbb{R}^{|C|}$, $C = \{I, O\}$) using a softmax function:

$$\mathbf{p}(\mathbf{x_i}) = softmax(Dropout(\mathbf{W} \cdot \mathbf{h_i} + \mathbf{b})), \quad (2)$$

where $\mathbf{W} \in \mathbb{R}^{|C|*r}$ and $\mathbf{b} \in \mathbb{R}^{|C|}$ are the weight matrix and bias.

After that, the training loss is formulated by the averaged cross-entropy of the probability distribution and the ground-truth labels:

$$\mathcal{L}_{span} = \frac{1}{N} \sum_{i=1}^{N} CrossEntropy(y_i, \mathbf{p}(\mathbf{x_i})), \quad (3)$$

where y_i=0 when the i-th token is O-token, y_i=1 otherwise. Specifically, we denote the training loss of span detection stage as $\mathcal{L}_{span}$. During the training procedure, the parameters $\{\theta_1, \mathbf{W}, \mathbf{b}\}$ are updated to minimize $\mathcal{L}_{span}$.

3.2.2 Type Classification

Representation Given an input sentence X, we only select entity-tokens $E = \{e_1, e_2, ..., e_M\}$ ($E \subset X$) with ground-truth labels $Y = \{y_1, y_2, ..., y_M\}$ for the training of this stage. For

[3] In this paper, r denotes the hidden size of the pretrained language model encoder.

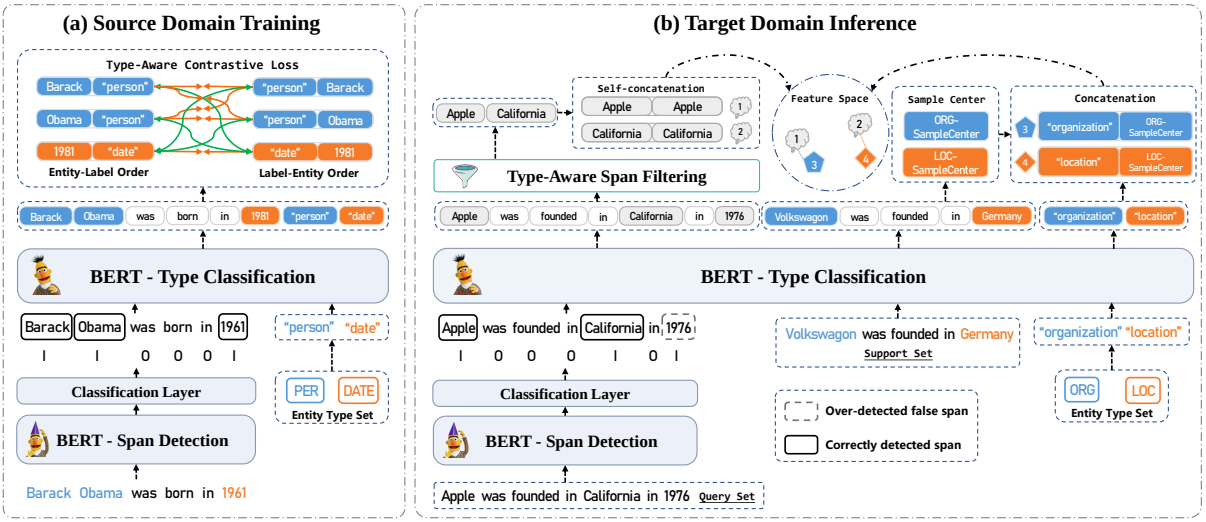


Figure 2: The overall structure of our proposed TadNER framework. (a) Training in the source domain. (b) Inference on the query set by utilizing the support samples in the target domain. Note that the source and target domains have different entity type sets.

the entity type set $\mathcal{T}_{source} = \{t_1, t_2, ..., t_m\}$ of the source domain D_{source}, we manually convert them into their corresponding type names $\mathcal{T}'_{source} = \texttt{Map}(\mathcal{T}_{source}) = \{t'_1, t'_2, ..., t'_m\}$[4].

After that, to obtain tokens with type name information, which are further used for calculating contrastive loss, we concatenate entity tokens with their corresponding labels in two orders, i.e., entity-label order and label-entity order. Here we use another encoder f_{θ_2} with parameters θ_2 to obtain contextual representations:

$$\mathbf{h_i^{el}} = f_{\theta_2}(e_i) \oplus f_{\theta_2}(\texttt{Map}(y_i)) \tag{4}$$

$$\mathbf{h_i^{le}} = f_{\theta_2}(\texttt{Map}(y_i)) \oplus f_{\theta_2}(e_i), \tag{5}$$

where $\oplus$ is the concatenation operator, and $\mathbf{h_i^{el}}$ and $\mathbf{h_i^{le}}$ denote two kinds of type-aware representations of the entity-token e_i, which are obtained in entity-label order and label-entity order, respectively.

Type-Aware Contrastive Learning To learn a generalized and type-aware feature space, which can further be used for constructing more accurate and stable prototypes, we borrow the idea of contrastive learning (Khosla et al., 2020) and use two kinds of type-aware token representations mentioned above to construct positive and negative pairs as shown in Fig. 2 (a), i.e., those with the same label in different orders as positive pairs and those with different labels as negative pairs. The type-aware contrastive loss is calculated as:

$$\mathcal{L}_{type} = -\sum_{i=1}^{M} \log \frac{\frac{1}{\|Z_i\|} \sum\limits_{z \in Z_i} \exp(sim(\mathbf{h_i^{el}}, \mathbf{h_z^{le}})/\tau)}{\sum\limits_{j=1}^{M} \exp(sim(\mathbf{h_i^{el}}, \mathbf{h_j^{le}})/\tau)}, \tag{6}$$

$$Z_i = \{z \mid 1 \le z \le M, y_z = y_i\}, \tag{7}$$

$$sim(\mathbf{h_i^{el}}, \mathbf{h_j^{le}}) = \frac{\mathbf{h_i^{el}} \cdot \mathbf{h_j^{le}}^{\top}}{\sum_{k=1}^{M} (\mathbf{h_k^{el}} \cdot \mathbf{h_j^{le}}^{\top})}, \tag{8}$$

where M is the number of entity tokens in a batch and Z_i is the set of positive samples with the same label type y_i. Here we adopt the dot product with a normalization factor as the similarity function $sim()$. We also add a temperature hyper-parameter τ for focusing more on difficult pairs (Chen et al., 2020). During the source domain training, the parameters θ_2 are updated to minimize $\mathcal{L}_{type}$.

3.3 Target Domain Inference

As illustrated in Fig. 2 (b), during the target domain inference, we first extract candidate spans in query sentences. We then remove the over-detected false spans via the proposed type-aware span filtering strategy. We finally classify remaining candidate spans into certain entity types to get the final results.

[4] Map() is the function used to convert a label to a type name, e.g. "PER" to "person". Please refer to Appendix A.6 for type names of all datasets.

Algorithm 1 Type classification with type-aware span filtering in TadNER

Require: Candidate span set C_{span}, Support set S_{target}, Label type set $\mathcal{T}_{target}$, PLM for the type classifier f_{θ_2}
$\mathcal{L}_{prev} \in \mathbb{R}_+$ (arbitrarily large value)
$\mathcal{L}_{label} = \mathcal{L}_{prev} - 1$
while $\mathcal{L}_{label} < \mathcal{L}_{prev}$ **do**
  $\mathcal{L}_{prev} = \mathcal{L}_{label}$
  Calculate $\mathcal{L}_{label}$ using Eq. (9).
  Update $f_{\theta_2} \rightarrow f_{\theta_2'}$ by backpropagation to reduce $\mathcal{L}_{label}$
end while
Calculate the threshold γ_t using Eq. (11).
Calculate prototypes of all types in $\mathcal{T}_{target}$ using Eq. (12)
$S_{result} = \{\}$
for each span s_i in C_{span} **do**
  $max\_sim = \max_{t_j \in \mathcal{T}_{target}} ((f_{\theta_2'}(s_i) \oplus f_{\theta_2'}(s_i)) \cdot \mathbf{p_j}^\mathsf{T})$
  if $max\_sim/2 > \gamma_t$ **then**
    Assign the label y_{pred} to s_i using Eq. (14)
    $S_{result} = S_{result} \cup \{s_i\}$
  end if
end for
return S_{result}

3.3.1 Span Detection

The span detector with its parameters $\{\theta_1, \mathbf{W}, \mathbf{b}\}$ trained in the source domain is further fine-tuned with samples in the support set $\mathcal{S}_{target}$ in the target domain to minimize $\mathcal{L}_{span}$ in Eq.(3). To alleviate the risk of over-fitting, we adopt a loss-based early stopping strategy, i.e., stopping the fine-tuning procedure once the loss rises β times continuously, where β is a hyper-parameter.

After fine-tuning the span detector, we use it to detect entity words of query sentences in $\mathcal{Q}_{target}$ and then consider continuous entity words as a candidate span, e.g., "Barack Obama". Finally, we obtain the candidate span set C_{span} containing all candidate spans, which will be assigned entity types at the type classification stage.

3.3.2 Type Classification

Domain Adaption Benefiting from the generalized and type-aware feature space trained in the source domain, we can further get a domain-specific encoder $f_{\theta_2'}$ via fine-tuning with the following loss:

$$\mathcal{L}_{label} = \frac{1}{M} \sum_{i=1}^{M} \frac{s(e_i^s, \texttt{Map}(y_i^s))}{\sum_{t_j \in \mathcal{T}_{target}} s(e_i^s, \texttt{Map}(t_j))}, \quad (9)$$

$$s(p, q) = f_{\theta_2}(p) \cdot f_{\theta_2}(q)^\mathsf{T}. \quad (10)$$

Type-Aware Span Filtering As we illustrate in the introduction, the span detector may generate some over-detected false spans whose type names only exist in the source domain. To solve this problem, we propose a type-aware span filtering strategy during the inference phase to remove these false spans. Intuitively, the over-detected false spans are extracted because they do not consider the semantics of entity type names in the target domain. Based on this assumption, we calculate a threshold γ_t with the fine-tuned encoder $f_{\theta_2'}$ using entity tokens and corresponding type names in the support set:

$$\gamma_t = \min_{1 \le i \le M} f_{\theta_2'}(e_i^s) \cdot f_{\theta_2'}(\texttt{Map}(y_i^s))^\mathsf{T}. \quad (11)$$

The threshold γ_t will be used to remove the over-detected false spans. The remaining candidate spans will be assigned the corresponding label types.

Type-Aware Prototype Construction We can construct a type-aware prototype for each entity type $t_j \in \mathcal{T}_{target}$, which is more accurate and stable owing to the generalized and type-aware feature space learned in the source domain:

$$\mathbf{p_j} = f_{\theta_2'}(\texttt{Map}(t_j)) \oplus \frac{1}{\|Z_j\|} \sum_{i \in Z_j} f_{\theta_2'}(e_i^s), \quad (12)$$

$$Z_j = \{i \mid 1 \le i \le M, y_i^s = t_j\}, \quad (13)$$

where $\oplus$ is the concatenation operator and Z_j denotes the set of entity words with the label type t_j in the support set.

Inference For each remaining candidate span s_i, we assign it a label type $t_j \in \mathcal{T}_{target}$ with the highest similarity:

$$y_{pred} = \arg\max_{t_j, t_j \in \mathcal{T}_{target}} (\mathbf{h_i} \cdot \mathbf{p_j}^\mathsf{T}), \quad (14)$$

$$\mathbf{h_i} = f_{\theta_2'}(s_i) \oplus f_{\theta_2'}(s_i), \quad (15)$$

where $\mathbf{p_j}$ is the type-aware prototype representation corresponding to the label type t_j, and y_{pred} is the predicted label type of the candidate span s_i. $\mathbf{h_i}$ is the self-concatenated representation of s_i for consistency with the dimension of the prototype $\mathbf{p_j}$.

The entire procedure of type classification in the target domain is presented in Algorithm 1.

4 Evaluation Protocal

4.1 Settings

Few-shot NER has two typical settings, including the Few-NERD setting and the Domain Transfer setting.

4.1.1 Few-NERD Setting

Datasets Ding et al. (2021) propose a large scale dataset Few-NERD (Wiki) for few-shot NER, which contains 66 fine-grained entity types across 8 coarse-grained entity types. It contains two intra and inter tasks where the train/dev/test sets are divided according to the coarse-grained and fine-grained types, respectively. Note the intra task is more challenging since the coarse-grained entity types in the source and target domains are non-overlapping.

Evaluation Following Ma et al. (2022c), we adopt the episode-level evaluation method. Each episode consists of a support set and a query set, both given in the n-way k-shot form. In each episode, the model trained in the source domain is tested on the query set by utilizing the support set. To make fair comparisons, we obtain the micro F1 score with the episode-data processed by Ding et al. (2021) [5]. We report the mean F1 score with standard deviation using 3 different seeds.

4.1.2 Domain Transfer Setting

Datasets Following Das et al. (2022), we conduct cross-domain experiments under the Domain Transfer setting. In this setting, data are from different text domains (e.g., Wiki, News), and thus it even harder to obtain desirable performance. We take OntoNotes (General) (Weischedel et al., 2013) as our source domain, and evaluate few-shot performances on I2B2 (Medical) (Stubbs and Uzuner, 2015), CoNLL (News) (Tjong Kim Sang and De Meulder, 2003), WNUT (Social) (Derczynski et al., 2017) and GUM (Zeldes, 2017) domains. Please refer to Appendix A.1 for more details about the datasets.

Evaluation Yang and Katiyar (2020) point that sampling test episodes may not reflect the real-world performance due to various data distributions, and they propose to sample support sets and then test the model in the original test set. Each support set consists of k examples corresponding to each label. The final micro F1 scores and standard deviations are obtained using different sampled support sets. We also adopt this evaluation schema for Domain Transfer settings, and we directly use the support sets sampled by Das et al. (2022)[6] for fair comparisons.

[5]We use the same data as those in https://ningding97.github.io/fewnerd/ for fair comparisons.

[6]https://github.com/psunlpgroup/CONTaiNER.

4.2 Baselines

We compare our proposed TadNER with many strong baselines which are divided into *one-stage* and *two-stage* types.

The *one-stage* baselines include ProtoBERT (Snell et al., 2017), NNShot (Yang and Katiyar, 2020), StructShot (Yang and Katiyar, 2020), and CONTaiNER (Das et al., 2022).

The *two-stage* baselines include ESD (Wang et al., 2022) and the state-of-the-art method DecomposedMetaNER (Ma et al., 2022c) on Few-NERD leaderboard [7]. We reproduce DecomposedMetaNER and conduct experiments under Domain Transfer settings with the dataset-level evaluation schema since the authors only report results using the episode-level evaluation method.

Please refer to Appendix A.2 and Appendix A.3 for more descriptions about the baselines as well as the implementation details.

5 Results and Analysis

5.1 Main Results

Table 1 and 2 report the comparison results between our method and baselines under Few-NERD and Domain Transfer, respectively. We have the following important observations.

Our model is the best in both intra and inter tasks under the Few-NERD setting. In particular, in the more challenging intra task, our TadNER reaches an average 7.7% F1 improvement.

Under the Domain Transfer setting, our model outperforms all baselines on average. It also achieves the best performance in almost all separate cases with only one exception.

If there are only very few samples provided (e.g., 1 shot or 1∼2 shot), the improvements by our model become more significant, which is a very attractive property.

We also find that DecomposedMetaNER performs extremely poorly in Domain Transfer. Note there are many sentences without entities in some datasets [8] in this setting, which bring about many over-detected false spans and seriously hurt

[7]Few-NERD leaderboard: https://paperswithcode.com/dataset/few-nerd. Results on the leaderboard are tested with the latest version of data, which is corresponding with https://github.com/microsoft/vert-papers/tree/master/papers/DecomposedMetaNER#few-nerd-arxiv-v6-version.

[8]e.g., I2B2, where nearly 80% of the sentences have no entities. In the episode-level evaluation setting, this does not happen because the query set is sampled by n-way k-shot.

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	
One-stage	ProtoBERT†	20.76±0.84	15.05±0.44	42.54±0.94	35.40±0.13	28.44	38.83±1.49	32.45±0.79	58.79±0.44	52.92±0.37	45.75
	NNShot†	25.78±0.91	18.27±0.41	36.18±0.79	27.38±0.53	26.90	47.24±1.00	38.87±0.21	55.64±0.63	49.57±2.73	47.83
	StructShot†	30.21±0.90	21.03±1.13	38.00±1.29	26.42±0.60	28.92	51.88±0.69	43.34±0.10	57.32±0.63	49.57±3.08	50.53
	CONTaiNER*	41.51±0.07	36.62±0.04	57.83±0.01	51.04±0.24	46.75	50.92±0.29	47.02±0.24	63.35±0.07	60.14±0.16	55.36
Two-stage	ESD†	36.08±1.60	30.00±0.70	52.14±1.50	42.15±2.60	40.09	59.29±1.25	52.16±0.79	69.06±0.80	64.00±0.43	61.13
	DecomposedMetaNER†	49.48±0.85	42.84±0.46	62.92±0.57	57.31±0.25	53.14	64.75±0.35	58.65±0.43	71.49±0.47	68.11±0.05	65.75
	TadNER	**60.29±0.14**	**54.82±0.21**	**67.53±0.11**	**60.55±0.17**	**60.80**	**64.79±0.09**	**63.17±0.18**	**71.87±0.05**	**69.30±0.14**	**67.28**

Table 1: F1 scores with standard deviations for Few-NERD. † denotes the results reported by Ma et al. (2022c). * denotes the results reported by our replication using data of the same version since the original ones do not report standard deviations. The best results are in **bold** and the second best ones are underlined.

Paradigms	Models	1-shot					5-shot				
		I2B2	CoNLL	WNUT	GUM	Avg.	I2B2	CoNLL	WNUT	GUM	Avg.
One-stage	ProtoBERT†	13.4 ± 3.0	49.9 ± 8.6	17.4 ± 4.9	17.8 ± 3.5	24.6	17.9 ± 1.8	61.3 ± 9.1	22.8 ± 4.5	19.5 ± 3.4	30.4
	NNShot†	15.3 ± 1.6	61.2 ± 10.4	22.7 ± 7.4	10.5 ± 2.9	27.4	22.0 ± 1.5	74.1 ± 2.3	27.3 ± 5.4	15.9 ± 1.8	34.8
	StructShot†	21.4 ± 3.8	62.4 ± 10.5	24.2 ± 8.0	7.8 ± 2.1	29.0	30.3 ± 2.1	74.8 ± 2.4	30.4 ± 6.5	13.3 ± 1.3	37.2
	CONTaiNER†	21.5 ± 1.7	61.2 ± 10.7	27.5 ± 1.9	18.5 ± 4.9	32.2	**36.7 ± 2.1**	75.8 ± 2.7	32.5 ± 3.8	25.2 ± 2.7	42.6
Two-stage	DecomposedMetaNER*	15.5 ± 3.0	61.2 ± 9.2	27.7 ± 5.3	20.3 ± 4.2	31.2	19.8 ± 2.6	75.2 ± 5.8	29.8 ± 3.9	33.5 ± 2.4	39.6
	TadNER	**29.9 ± 2.5**	**70.4 ± 7.8**	**33.3 ± 4.8**	**26.1 ± 3.7**	**39.9**	34.6 ± 6.8	**78.8 ± 4.7**	**32.7 ± 3.8**	**35.9 ± 1.7**	**45.5**

Table 2: F1 scores with standard deviations for Domain Transfer. † denotes the results reported by Das et al. (2022). Since no previous two-stage methods have conducted experiments under this setting, we choose the strong DecomposedMetaNER for reproduction experiments, and * denotes the results reported by our replication. The best results are in **bold** and the second best ones are underlined.

the performance of DecomposedMetaNER. In contrast, our TadNER can remove false spans and achieve promising results in various scenarios with the help of the type-aware span filtering strategy.

5.2 Ablation Study

To validate the effectiveness of the main components in TadNER, we introduce the following variant baselines for the ablation study. 1) TadNER *w/o* Type-Aware Span Filtering (TASF) removes the type-aware span filtering strategy and directly feeds all spans detected at span detection stage to type classification. 2) TadNER *w/o* Type Names (TN) replaces type names with random type-agnostic vectors when calculating type-aware contrastive loss and constructs class prototypes with support samples only. 3) TadNER *w/o* Type-Aware Contrastive Learning (TACL) further removes random vectors and only uses entity tokens for constructing positive or negative pairs.

It is clear from Table 3 that removing any components will bring about performance decreases. 1) If removing the type-aware span filtering strategy, the performance drops a lot in various cases, including entity-sparse ones like I2B2 and entity-dense ones like GUM. This proves the robustness and effectiveness of our model in various real-world applications. 2) Removing type names also results in a significant performance decrease, indicating that our model does learn a type-aware feature space which plays a critical role in few-shot scenarios. 3) When we adopt contrastive learning without guidance of type names, the performance shows the biggest drop. This is because the replaced token-level contrastive learning requires a projection layer to prevent the model from collapsing. However, this makes it extremely hard for the model to learn the basic clustering capabilities.

Model	Domain Transfer				
	I2B2	CoNLL	WNUT	GUM	Avg.
TadNER	**29.9**	**70.4**	**33.3**	**26.1**	**39.9**
TadNER *w/o* TASF	14.3	67.7	30.5	22.8	33.8
TadNER *w/o* TN	10.1	64.3	29.6	20.8	31.2
TadNER *w/o* TACL	7.5	41.6	14.3	12.6	19.0

Table 3: Results (F1 scores) for ablation study under 1-shot Domain Transfer setting. The best results are in **bold** and the second best ones are underlined.

5.3 Case Study

To examine how our model accurately constructs prototypes and filters out over-detected false spans with the help of type names, we randomly select one query sentence from Few-NERD intra and

C1: Query sentence: with the promotion of emrespor to the turkish tff third league at the end of the 2011 season
DecomposedMetaNER: **organization-sportsteam**: **emrespor** (√), **turkish tff third league** (×)
TadNER (ours): **organization-sportsteam**: **emrespor** (√) **organization-sportsleague**: **turkish tff third league** (√)

C2: Query sentence: Leicestershire beat Somerset by an innings and 39 runs in two days.
DecomposedMetaNER: **ORG**: **Leicestershire** (√) **LOC**: **Somerset** (×), **two** (×)
TadNER (ours): **ORG**: **Leicestershire** (√), **Somerset** (√)

Figure 3: Case study. C1 and C2 are from Few-NERD intra and CoNLL2003 in Cross datasets, respectively, and organization-sportsteam, organization-sportsleague, ORG and LOC are entity types.

CoNLL2003 for case study. We compare TadNER with DecomposedMetaNER (Ma et al., 2022c), which also belongs to the two-stage methods.

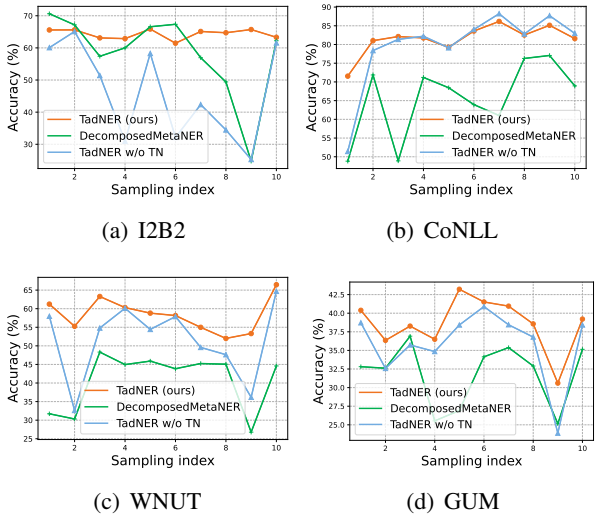
As shown in Fig. 3, in the first case, our model correctly predicts "turkish tff third league" as "organization-sportsleague" type, while DecomposedMetaNER identifies it as a wrong "organization-sportsteam" type. Since the type name and the entity span have a overlapping word "league", incorporating the type name into the construction of the prototype will make the identification much easier. Conversely, without the type name, it would be difficult to distinguish between two categories of entities because they both represent "sports-related organizations".

In the second case, DecomposedMetaNER incorrectly identifies "two" as an entity span and then assigns it a wrong entity type "LOC", since there are many samples like "The two sides had not met since Oct. 18" in the source domain Ontonotes, where "two" is an entity of "CARDINAL" type. In contrast, our TadNER removes this false span successfully by applying the type-aware span filtering strategy.

5.4 Impact of Type-Aware Prototypes

In order to investigate the effectiveness of our proposed strategy for solving the problem of inaccurate and unstable prototypes, we further perform an analysis of the impact of stability and quality of prototypes. We select two baselines as our compared methods. One is DecomposedMetaNER (Ma et al., 2022c). The other is TadNER w/o Type Names (TN) (the second variant baseline in the ablation study) as our compared methods. Here we adopt the same 10 samplings used in the 1-shot Domain Transfer experiments.

As shown in Fig. 4, our proposed TadNER achieves a significant improvement over DecomposedMetaNER on each dataset and is more stable under different samplings. Besides, removing type names will cause a sharp performance drop in some



(a) I2B2 (b) CoNLL (c) WNUT (d) GUM

Figure 4: Impacts of prototypes by different methods under 1-shot Domain Transfer setting. The horizontal and vertical coordinates indicate the n-th sampling and the accuracy of type classification, respectively.

cases for TadNER w/o TN, indicating that the incorporation of type names indeed helps construct more stable and accurate prototypes.

6 Conclusion

In this paper, we propose a novel TadNER framework for few-shot NER, which handles the span detection and type classification sub-tasks at two stages. For type classification, we present a type-aware contrastive learning strategy to learn a type-aware and generalized feature space, enabling the model to construct more accurate and stable prototypes with the help of type names. Based on it, we introduce a type-aware span filtering strategy for removing over-detected false spans produced at the span detection stage. Extensive experiments demonstrate that our method achieves superior performance over previous SOTA methods, especially in the challenging scenarios. In the future, we will extend TadNER to other NLP tasks such as POS tagging and explore its ability for zero-shot NER [9].

[9] A preliminary study on zero-shot NER has shown our improvements over SpanNER. Please refer to Appendix A.5.

Limitations

Our proposed TadNER mainly focuses on the type classification stage of few-shot NER and simply adopt binary classification for detecting entity spans. There might be better solutions, e.g., using global boundary matrix. However, due to its high GPU memory requirements, we do not include it in our current framework. This drives us to find more efficient and powerful span detector for better few-shot NER performance in the future.

Ethics Statement

Our work is entirely at the methodological level and therefore there will not be any negative social impacts. In addition, since the performance of the model is not yet at a practical level, it cannot be applied in certain high-risk scenarios (such as the I2B2 dataset used in our paper) yet, leaving room for further improvements in the future.

References

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR.

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 Passonneau, and Rui Zhang. 2022. CONTaiNER: Few-shot named entity recognition via contrastive learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6338–6353, Dublin, Ireland. 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.

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*, SAC '19, page 993–1000, New York, NY, USA. Association for Computing Machinery.

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.

Yucheng Huang, Kai He, Yige Wang, Xianli Zhang, Tieliang Gong, Rui Mao, and Chen Li. 2022. COPNER: Contrastive learning with prompt guiding for few-shot named entity recognition. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2515–2527, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

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

Bin Ji, Shasha Li, Shaoduo Gan, Jie Yu, Jun Ma, Huijun Liu, and Jing Yang. 2022. Few-shot named entity recognition with entity-level prototypical network enhanced by dispersedly distributed prototypes. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1842–1854, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673. Curran Associates, Inc.

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

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

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.

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 *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1956–1971, Dublin, Ireland. Association for Computational Linguistics.

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

Tingting Ma, Huiqiang Jiang, Qianhui Wu, Tiejun Zhao, and Chin-Yew Lin. 2022c. Decomposed meta-learning for few-shot named entity recognition. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1584–1596, Dublin, Ireland. Association for Computational Linguistics.

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

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 *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2782–2794, Online. Association for Computational Linguistics.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical Networks for Few-shot Learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Amber Stubbs and Özlem Uzuner. 2015. Annotating longitudinal clinical narratives for de-identification: The 2014 i2b2/uthealth corpus. *Journal of biomedical informatics*, 58:S20–S29.

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

Peiyi Wang, Runxin Xu, Tianyu Liu, Qingyu Zhou, Yunbo Cao, Baobao Chang, and Zhifang Sui. 2022. An enhanced span-based decomposition method for few-shot sequence labeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5012–5024, Seattle, United States. Association for Computational Linguistics.

Yaqing Wang, Haoda Chu, Chao Zhang, and Jing Gao. 2021. Learning from language description: Low-shot named entity recognition via decomposed framework. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1618–1630, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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*.

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.

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.

Amir Zeldes. 2017. The gum corpus: Creating multilayer resources in the classroom. *Lang. Resour. Eval.*, 51(3):581–612.

Xinghua Zhang, Bowen Yu, Yubin Wang, Tingwen Liu, Taoyu Su, and Hongbo Xu. 2022. Exploring modular task decomposition in cross-domain named entity recognition. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 301–311.

A Appendix

A.1 Datasets Details

Table 4 shows statistics of various datasets used in our experiments.

Dataset	Domain	# Classes	# Sentences	# Entities
Few-NERD	Wikipedia	66	188.2k	491.7k
I2B2'14	Medical	23	140.8k	29.2k
CoNLL'03	News	4	20.7k	35.1k
GUM	Wiki	11	3.5k	6.1k
WNUT'17	Social	6	5.7k	3.9k
OntoNotes	General	18	76.7k	104.2k

Table 4: Dataset statistics

A.2 Baselines

ProtoBERT (Fritzler et al., 2019) adopts a token-level prototypical network, where the prototype of each class is obtained by averaging token samples of the same label, and the label of each unlabeled token in the query set is determined by its nearest class prototype.

NNShot (Yang and Katiyar, 2020) pre-trains BERT by traditional classification methods in the source domain training phase, and decides the class of each unlabeled token by the nearest neighbor at the token level in the target domain inference phase.

StructShot (Yang and Katiyar, 2020) is based on NNshot and uses an abstract transition probability for Viterbi decoding during testing.

ESD (Wang et al., 2022) uses a span-level prototypical network, which designs multiple prototypes for O-tokens and uses inter- and cross-span attention for better span representation.

CONTaiNER (Das et al., 2022) first trains BERT in the source domain using token-level contrastive learning loss function, then fine-tunes the trained model on the support set, and finally use the nearest neighbor method proposed in NNShot (Yang and Katiyar, 2020) for target domain inference phase.

DecomposedMetaNER (Ma et al., 2022c) is a decomposed approach that incorporates model-agnostic meta-learning strategy into traditional prototypical network to learn a model-agnostic model and more fully exploits the support set.

A.3 Implementation Details

Following previous methods (Ding et al., 2021; Das et al., 2022; Ma et al., 2022c), we use `bert-base-uncased` model (Devlin et al., 2019) from HuggingFace (Wolf et al., 2020)[10] as our encoder f_{θ_1} and f_{θ_2}.

During the source domain training procedure, we use AdamW (Loshchilov and Hutter, 2019) as the optimizer with a learning rate of 3e-5 and 1% linear warmup steps, and the batch size is set to 64. We set the temperature hyper-parameter τ = 0.05 in Eq.(6) and keep dropout rate as 0.2 in the classification layer of the span detection.

As for the early stopping strategy in 3.3.1, we found that the fewer samples face a higher risk of over-fitting, and a lower β threshold is required. So we set $\beta = 2$ in all 1-shot settings and $\beta = 6$ in all other cases. Table 5 shows the searching space of each hyper-parameter. Besides, we implement our framework with Pytorch 1.12[11] and train it with a V100-16G GPU.

Using a V100-16G GPU, we trained the model on the source domain OntoNotes dataset for 60 minutes. The finetuning procedures of span detection and type classification stages require less than 20 seconds in total under 5-shot settings.

Learning rate	{1e-5, 3e-5, 1e-4}
Batch size	{ 32, 64, 128}
Dropout rate	{0.1, 0.2, 0.5}
temperature τ	{0.01, 0.05, 0.1}
Early stopping threshold β	{1, 2, 4, 6, 8}

Table 5: Hyper-parameters search space in our experiments.

A.4 Performance of the Span Detection

Table 6 and Table 7 show the performance of the span detection under Few-NERD and Domain Transfer settings, respectively. As we can see that the precision of the span detection phase is much lower than the recall, which also indicates the existence of the over-detection problem in the decomposed method, especially serious in entity-sparse dataset (I2B2). While this problem is alleviated in our proposed approach by introducing the type-aware span filtering strategy.

[10] https://huggingface.co/bert-base-uncased
[11] https://pytorch.org/

Metric	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
Precision	66.57	68.69	74.84	75.37	65.27	67.30	72.34	73.20
Recall	76.25	76.50	80.98	81.57	73.30	73.67	77.84	77.50
Micro F1	70.31	72.01	77.62	78.26	68.20	69.92	74.78	75.17

Table 6: Performance of the Span Detection under Few-NERD settings. Precision, recall and micro F1 scores are reported.

Metric	1-shot				5-shot			
	I2B2	CoNLL	WNUT	GUM	I2B2	CoNLL	WNUT	GUM
Precision	17.77	74.85	42.85	47.42	19.41	86.91	41.78	56.71
Recall	66.92	84.81	62.84	50.96	74.15	90.30	67.10	67.33
Micro F1	27.73	79.13	50.16	49.02	30.68	88.52	50.72	61.54

Table 7: Performance of the Span Detection under Domain Transfer settings. Precision, recall and micro F1 scores are reported.

A.5 Zero-Shot Performance

Since there is no domain-specific support set under zero-shot NER settings, it is extremely challenging and rarely explored. While we believe our proposed TadNER can obtain certain zero-shot ability after training in the source doamin for the following two reasons: 1) the model can extract entity spans in the span detection stage before fine-tuning with support samples, 2) since the feature space learnt in the type classification stage is well generalized and type-aware, we can directly adopt the representations of type names as prototypes of novel entity types. To demonstrate the promising performance of our model under zero-shot settings, we select SpanNER (Wang et al., 2021) as a strong baseline, which is a decomposed-based method and good at solving zero-shot NER problem.

Model	Domain Transfer				
	I2B2	CoNLL	WNUT	GUM	Avg.
SpanNER (0-shot)	8.02	23.63	24.82	6.57	15.76
TadNER (0-shot)	**17.13**	**43.14**	**25.06**	**7.62**	**23.24**

Table 8: F1 scores under Domain Transfer zero-shot settings.

As shown in Table 8, our proposed TadNER performs better than SpanNER (Wang et al., 2021) under every case. The reason for this may be that the type classification of SpanNER is based on a traditional supervised classification model, which performs worse generalization in cross-domain scenarios. Besides, compared with previous metric-based methods (Das et al., 2022; Ma et al., 2022c) for few-shot NER, which heavily rely on support sets and had **no** zero-shot capability, our method is more inspirational for future zero-shot NER works.

A.6 Type Names of Labels

Dataset	Original Labels	Corresponding Type names
Few-NERD	art-broadcastprogram	broadcast program
	art-film	film
	art-music	music
	art-other	other art
	art-painting	painting
	art-writtenart	written art
	person-actor	actor
	person-artist/author	artist author
	person-athlete	athlete
	person-director	director
	person-other	other person
	person-politician	politician
	person-scholar	scholar
	person-soldier	soldier
	product-airplane	airplane
	product-car	car
	product-food	food
	product-game	game
	product-other	other product
	product-ship	ship
	product-software	software
	product-train	train
	product-weapon	weapon
	other-astronomything	astronomy thing
	other-award	award
	other-biologything	biology thing
	other-chemicalthing	chemical thing
	other-currency	currency
	other-disease	disease
	other-educationaldegree	educational degree
	other-god	god
	other-language	language
	other-law	law
	other-livingthing	living thing
	other-medical	medical
	building-airport	airport
	building-hospital	hospital
	building-hotel	hotel
	building-library	library
	building-other	other building
	building-restaurant	restaurant
	building-sportsfacility	sports facility
	building-theater	theater
	event-attack/battle /war/militaryconflict	attack battle war military conflict
	event-disaster	disaster
	event-election	election
	event-other	other event
	event-protest	protest
	event-sportsevent	sports event
	location-bodiesofwater	bodies of water
	location-GPE	geographical social political entity
	location-island	island
	location-mountain	mountain
	location-other	other location
	location-park	park
	location-road/railway /highway/transit	road railway highway transit
	organization-company	company
	organization-education	education
	organization-government /governmentagency	government agency
	organization-media/newspaper	media newspaper
	organization-other	other organization
	organization-politicalparty	political party
	organization-religion	religion
	organization-showorganization	show organization
	organization-sportsleague	sports league
	organization-sportsteam	sports team

Table 9: Original labels and their corresponding natural-language-form type names of Few-NERD.

Dataset	Original Labels	Corresponding Type names
I2B2'14	AGE	age
	BIOID	biometric ID
	CITY	city
	COUNTRY	country
	DATE	date
	DEVICE	device
	DOCTOR	doctor
	EMAIL	email
	FAX	fax
	HEALTHPLAN	health plan number
	HOSPITAL	hospital
	IDNUM	ID number
	LOCATION_OTHER	location
	MEDICALRECORD	medical record
	ORGANIZATION	organization
	PATIENT	patient
	PHONE	phone number
	PROFESSION	profession
	STATE	state
	STREET	street
	URL	url
	USERNAME	username
	ZIP	zip code
CoNLL'03	PER	person
	LOC	location
	ORG	organization
	MISC	miscellaneous
GUM	abstract	abstract
	animal	animal
	event	event
	object	object
	organization	organization
	person	person
	place	place
	plant	plant
	quantity	quantity
	substance	substance
	time	time
WNUT'17	corporation	corporation
	creative-work	creative work
	group	group
	location	location
	person	person
	product	product
Ontonotes	CARDINAL	cardinal
	DATE	date
	EVENT	event
	FAC	fac
	GPE	geographical social political entity
	LANGUAGE	language
	LAW	law
	LOC	location
	MONEY	money
	NORP	nationality religion
	ORDINAL	ordinal
	ORG	organization
	PERCENT	percent
	PERSON	person
	PRODUCT	product
	QUANTITY	quantity
	TIME	time
	WORK_OF_ART	work of art

Table 10: Original labels and their corresponding natural-language-form type names of datasets under Domain Transfer settings.