# Meta-Learning Triplet Network with Adaptive Margins for Few-Shot Named Entity Recognition

Anonymous ACL submission


## Abstract

Meta-learning methods have been widely used in few-shot named entity recognition (NER), especially prototype-based methods. However, the `Other(O)` class is difficult to be represented by a prototype vector because there are generally a large number of samples in the class that have miscellaneous semantics. To solve the problem, we propose MeTNet, which generates prototype vectors for entity types only but not `O`-class. We design an improved triplet network to map samples and prototype vectors into a low-dimensional space that is easier to be classified and propose an adaptive margin for each entity type. The margin plays as a radius and controls a region with adaptive size in the low-dimensional space. Based on the regions, we propose a new inference procedure to predict the label of a query instance. We conduct extensive experiments in both in-domain and cross-domain settings to show the superiority of MeTNet over other state-of-the-art methods. In particular, we release a Chinese few-shot NER dataset FEW-COMM extracted from a well-known e-commerce platform. To the best of our knowledge, this is the first Chinese few-shot NER dataset. For reproducibility, all the datasets and codes are provided in the supplementary materials.


## 1 Introduction

Named entity recognition (NER), as a fundamental task in information extraction (Ritter et al., 2012), aims to locate and classify words or expressions into *pre-defined entity types*, such as `persons`, `organizations`, `locations`, `dates` and `quantities`. While a considerable number of approaches based on deep neural networks have shown remarkable success in NER, they generally require massive labeled data as training set. Unfortunately, in some specific domains, named entities that need professional knowledge to understand are difficult to be manually annotated in a large scale.

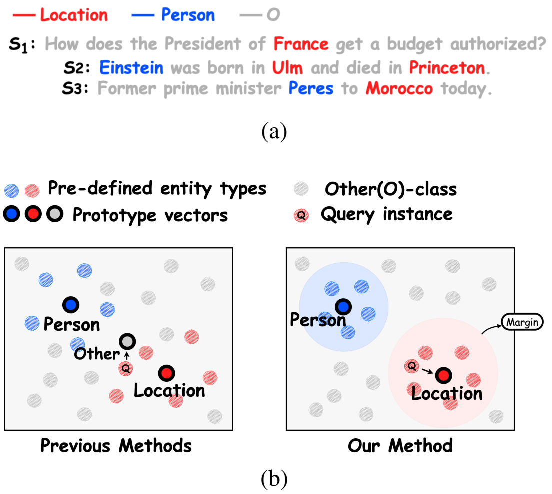


Figure 1: (a): Samples in `O`-class are semantically different. (b): The comparison between previous methods and ours to handle `O`-class. **Left:** Since the query instance whose true label is `Location` is closest to the prototype vector of `O`-class, previous methods misclassify it to `O`-class. **Right:** We compute prototype vectors for entity types only and learn an adaptive margin for each entity type to determine a region. Samples in the region of a class are labeled with the class, while samples outside of all the regions are predicted to be in `O`-class.

To address the problem, few-shot NER has been studied, which aims to recognize unseen entity types with few annotations. In particular, some models (Fritzler et al., 2019; Hou et al., 2020; Wang et al., 2021) are proposed based on the prototypical network (PROTO) (Snell et al., 2017), which is a popular meta-learning method. The general procedure of these prototype-based NER models is summarized as follows. First, they generate a prototype vector for each class, including both entity types and `Other(O)` class, to represent the class. Then they compute the distance between a query sample (instance) [1] and all these prototype vectors, and predict the query instance to the class with the smallest distance. However, for NER, the `O`-class

[1] We interchangeably use sample and instance in this paper.

covers all the miscellaneous words that are not classified as entity types. These words could span a wide range of semantics. For example, in Figure 1a, the words "was", "president", "budget" and "today" are semantically different even if they all belong to O-class. A single prototype vector would thus be insufficient to model the miscellaneous semantics of O-class, which could further lead to the incorrect prediction of query instances (see Figure 1b).

In this paper, to solve the issue, we propose to generate prototype vectors only for entity types but not O-class. In particular, we design a **Me**ta-Learning **T**riplet **Net**work with adaptive margins, namely, MeTNet, to map samples and prototype vectors into a low-dimensional space, where the inter-class distance between samples is enlarged and the intra-class distance between samples and their corresponding prototype vectors is shortened. We further design an improved triplet loss function with adaptive margins, which assigns different weights to samples, minimizes the absolute distance between an anchor and a positive sample, and maximizes the absolute distance between an anchor and a negative sample. The adaptive margin plays as a radius and controls a region for each entity type in the low-dimensional space (see Figure 1b). Based on these regions, we further propose a novel inference procedure. Specifically, given a query instance, we predict it to be in O-class, if it is located outside all the regions; otherwise, we label it with the entity type of its located region. Further, if it is contained in multiple regions, we label it with the entity type that has the smallest distance between the query instance and the region center. Finally, we summarize our main contributions in this paper as follows.

- We propose an improved triplet network with adaptive margins (MeTNet) and a new inference procedure for few-shot NER.
- We release the first Chinese few-shot NER dataset FEW-COMM, to our best knowledge.
- We perform extensive experiments to show the superiority of MeTNet over other competitors.

## 2 Related Work

### 2.1 Meta-Learning

Meta-learning, also known as "learning to learn", aims to train models to adapt to new tasks rapidly with few training samples. Some existing methods (Snell et al., 2017; Vinyals et al., 2016) are based on metric learning. For example, Matching Network (Vinyals et al., 2016) computes similarities between support sets and query instances, while the prototypical network (Snell et al., 2017) learns a prototype vector for each class and classifies query instances based on the nearest prototype vector. Other representative metric-based methods include Siamese Network (Koch et al., 2015) and Relation Network (Sung et al., 2018). Further, some approaches, such as MAML (Finn et al., 2017) and Reptile (Nichol et al., 2018), are optimization-based, which aim to train a meta-learner as an optimizer or adjust the optimization process. There also exist model-based methods, which learn a hidden feature space and predict the label of a query instance in an end-to-end manner. Compared with the optimization-based methods, model-based methods could be easier to optimize but less generalizable to out-of-distribution tasks (Hospedales et al., 2020). The representative model-based methods include MANNs (Santoro et al., 2016), Meta networks (Munkhdalai and Yu, 2017), SNAIL (Mishra et al., 2017) and CPN (Garnelo et al., 2018).

### 2.2 Few-shot NER

Few-shot NER has recently received great attention (Huang et al., 2021; Das et al., 2021; Ma et al., 2022) and meta-learning-based methods have been applied to solve the problem. For example, Fritzler et al. (2019) combine PROTO (Snell et al., 2017) with conditional random field for few-shot NER. Inspired by the nearest neighbor inference (Wiseman and Stratos, 2019), StructShot (Yang and Katiyar, 2020) employs structured nearest neighbor learning and Viterbi algorithm to further improve PROTO. MUCO (Tong et al., 2021) trains a binary classifier to learn multiple prototype vectors for representing miscellaneous semantics of O-class. ESD (Wang et al., 2021) uses various types of attention based on PROTO to improve the model performance. However, most of these methods use one or multiple prototype vectors to represent O-class, while we compute prototype vectors for entity types only and further design a new inference procedure.

Very recently, prompt-based techniques have also been applied in few-shot NER (Cui et al., 2021; Ma et al., 2021; Chen et al., 2021; Cui et al., 2022). However, the performance of these methods is very unstable, which heavily depend on the designed prompts (Cui et al., 2021). Thus, without a large validation set, their applicability is limited in few-shot learning.

## 3 Background

### 3.1 Problem Definition

A training set $\mathcal{D}_{train}$ consists of word sequences and their label sequences. Given a word sequence $X = \{x_1, ..., x_n\}$, we denote $L = \{l_1, ..., l_n\}$ as its corresponding label sequence. We use $\mathcal{Y}_{train}$ to denote the label set of the training data and $l_i \in \mathcal{Y}_{train}$. In addition, given a test set $\mathcal{D}_{test}$, let $\mathcal{Y}_{test}$ denote the label set of the test set, which satisfies $\mathcal{Y}_{train} \cap \mathcal{Y}_{test} = \emptyset$. Our goal is to develop a model that learns from $\mathcal{D}_{train}$ and then makes predictions for unseen classes in $\mathcal{Y}_{test}$, for which we only have few annotations.

### 3.2 Meta-training

Meta-learning methods include two stages: meta-training and meta-testing. In meta-training, the model is trained on meta-tasks sampled from $\mathcal{D}_{train}$. Each meta-task contains a support set and a query set. To create a training meta-task, we first sample $N$ classes from $\mathcal{Y}_{train}$. After that, for each of these $N$ classes, we sample $K$ instances as the support set $\mathcal{S}$ and $L$ instances as the query set $\mathcal{Q}$. The support set is similar as the training set in the traditional supervised learning but it only contains a few samples; the query set acts as the test set but it can be used to compute gradients for updating model parameters in meta-training stage. Given the support set, we refer to the task of making predictions over the query set as *$N$-way $K$-shot classification*.

### 3.3 Meta-testing

In the testing stage, we also use meta-tasks to test whether our model can adapt quickly to new classes. To create a testing meta-task, we first sample $N$ new classes from $\mathcal{Y}_{test}$. Similar as in meta-training, we then sample the support set and the query set from the $N$ classes, respectively. The support set is used for fine-tuning while the query set is for testing. Finally, we evaluate the average performance on the query sets across all testing meta-tasks.

## 4 Method

In this section, we describe our MeTNet algorithm. We first give an overview of MeTNet, which is illustrated in Figure 2. MeTNet first represents samples with BERT text encoder, based on which the embeddings of words and prototype vectors are initialized. Then it generates triples based on the support sets and prototype vectors, and employs an improved triplet network with adaptive margins to map words and prototype vectors into a space that is much easier to classify. For each entity type, an adaptive margin plays as a radius and controls a region centered at the corresponding prototype vector. These regions are further used in the inference stage. Next, we describe each component of MeTNet in detail.

### 4.1 Text Encoder

We first represent each word in a low-dimensional embedding vector. Following (Yang and Katiyar, 2020; Ding et al., 2021), we use BERT (Devlin et al., 2018) as our text encoder. Specifically, given a sequence of $n$ words $[x_1, x_2, ..., x_n]$, we take the output of the final hidden layer in BERT as the initial representations $\mathbf{h}_i$ for $x_i$:

$$[\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n] = \texttt{BERT}_\phi([x_1, x_2, ..., x_n]), \quad (1)$$

where $\phi$ represents parameters of BERT. Then for each pre-defined entity type $c_j$, we construct its initial prototype vector $\mathbf{h}_{c_j}$ by averaging the representations of words labeled as $c_j$.

### 4.2 Triplet Network

A triplet network (Hoffer and Ailon, 2015) is composed of three sub-networks, which have the same network architecture with shared parameters to be learned. For the triplet network, triples are taken as its inputs. Each triple consists of an anchor, a positive sample and a negative sample, and we feed each sample into a sub-network.

**Construct Triples** We first construct triples for different entity types. Specifically, for each entity type, we take its prototype vector as the anchor, instances in the entity type as positive samples, and other instances as negative ones. Since the number of negative samples is generally larger than that of positive samples, we select $k$ negative samples with the nearest distance to the prototype vector. After that, for each positive sample and each negative sample, we construct triples, respectively.

**Improved Triplet Loss** Given the distance $d_p$ between the anchor and the positive sample, and the distance $d_n$ between the anchor and the negative sample, the original triplet loss aims to optimize the *relative distance* among the anchor, the positive

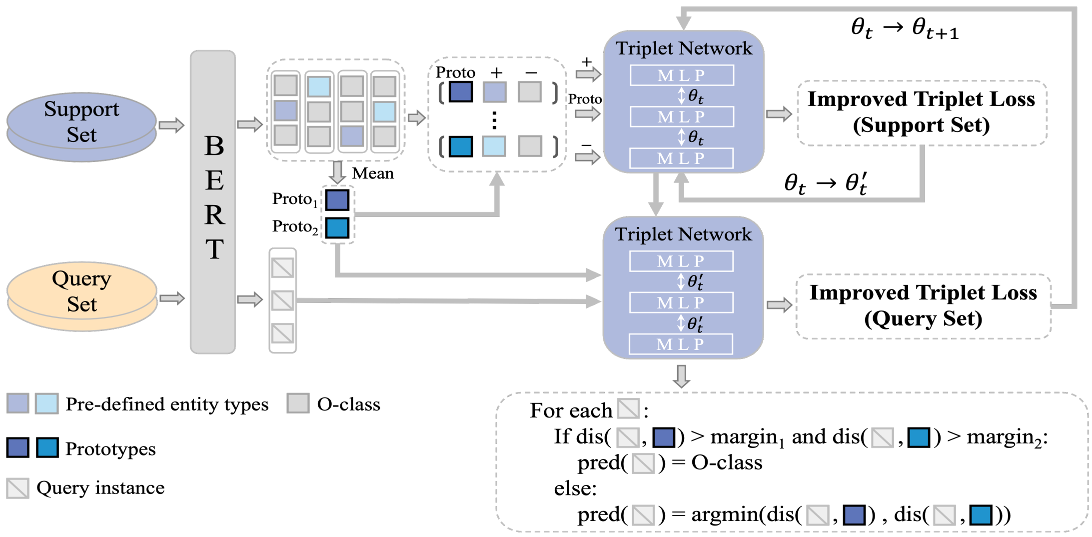


Figure 2: The overall architecture of MeTNet for a 2-way 2-shot problem.

sample and the negative sample, which is formulated as:

$$\mathcal{L}_T = max(0, m + d_p - d_n), \tag{2}$$

$$d_p = d(f_\theta(\mathbf{h}_a), f_\theta(\mathbf{h}_p)), \tag{3}$$

$$d_n = d(f_\theta(\mathbf{h}_a), f_\theta(\mathbf{h}_n)), \tag{4}$$

where $m$ is a margin, $d(\cdot, \cdot)$ denotes the Euclidean distance function and $f_\theta(\cdot)$ is the embedding vector generated from the triplet network. However, there exist three main problems in the original triplet loss function. First, the relative distance could lead to a very small $d_p$ or a very large $d_n$ only, while our goal is to derive both of them. Second, the loss function considers all the samples are equally important, but their importance is empirically relevant to their distance to the anchor. Third, the margin is fixed and unique. However, different entity types generally correspond to regions with various sizes. To address these problems, we design an improved triplet loss as follows:

$$\mathcal{L}_{IT} = \frac{\alpha}{1 + e^{-(d_p - m_i)}} \cdot d_p + \frac{1 - \alpha}{1 + e^{-(m_i - d_n)}} \cdot max(m_i - d_n, 0), \tag{5}$$

where $\alpha$ is a balancing weight and $m_i$ denotes a learnable margin of entity type $c_i$. In Equation 5, we separately optimize the *absolute distances* $d_p$ and $d_n$. On the one hand, we directly minimize $d_p$. On the other hand, considering that each entity type uses a region to include positive samples, we thus maximize $d_n$ by pushing the negative sample away from the region. Further, we assign different weights to samples based on their distances to anchors. Intuitively, the farther the positive samples or the closer the negative samples are to the anchors, the larger the weights should be given to amplify the loss. Finally, we set adaptive margins for different entity types, which play as region radiuses and control region sizes.

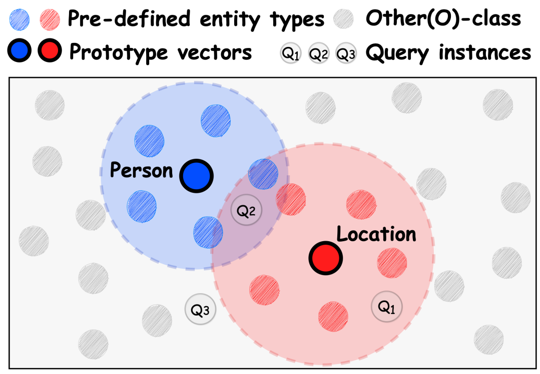


Figure 3: An example to illustrate the inference procedure in MeTNet. The dashed circles represent the regions of pre-defined entity types determined by adaptive margins. The labels of $Q_1$, $Q_2$ and $Q_3$ are predicted to be `Location`, `Person` and `O`-class, respectively.

### 4.3 Inference

In the inference stage, most existing methods calculate the distances between a query instance and all the prototype vectors for both entity types and `O`-class, and predict the query instance to be in the class with the smallest distance. Different from these methods, our model avoids handling `O`-class directly. Instead, we make predictions based on the regions of entity types. As shown in Figure 3, the entity types `Person` and `Location` have their own regions controlled by different margins. When a query instance (e.g., $Q_1$) is only located in one region, we label it with the entity type corresponding to the located region; when a query instance (e.g., $Q_2$) is contained in multiple regions, we calculate its distances to different region centers and predict its entity type to be that with the smallest distance; when a query instance (e.g., $Q_3$) is outside all the regions, it is labeled with `O`-class.

### 4.4 Training Procedure

Inspired by MAML (Finn et al., 2017), we first update the model parameters $\theta$ with samples in the support set:

$$\theta' = \theta - \alpha \nabla_\theta \mathcal{L}_{IT}(\theta; \mathcal{S}), \tag{6}$$

where $\alpha$ is the learning rate and $\mathcal{S}$ represents the support set. With few-step updates, $\theta$ becomes $\theta'$. Then based on $\theta'$, the triplet network can map query instances and prototype vectors into a low-dimensional space that is much easier to classify. After that, we update the model parameters $\theta$ with samples in the query set:

$$\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}_{IT}(\theta'; \mathcal{Q}), \tag{7}$$

where $\beta$ is the meta learning rate and $\mathcal{Q}$ represents the query set. This optimization simulates the testing process in the training stage and boosts the generalizability of the model to unseen classes with only few-step updates. The overall procedure of MeTNet is summarized in Algorithm 1.

**Algorithm 1** MeTNet Training procedure

**Input:** Training data $\{\mathcal{D}_{train}, \mathcal{Y}_{train}\}$; $ep$ epochs and the number $T$ of iterations of the model updated by the support set in a task; $N$ classes in the support set or the query set; $K$ samples in each class in the support set and $L$ samples in each class in the query set; the pre-trained BERT parameter $\phi$; the model parameter $\theta$; the set $\mathcal{M}$ of adaptive margins;

**Output:** $\phi$, $\theta$ and $\mathcal{M}$ after training;

```
Randomly initialize θ and M;
for each i ∈ [1, ep] do
    Y ← Sample(Y_train, N);
    S, Q ← ∅, ∅;
    for y ∈ Y do
        S ← S ∪ Sample(D_train{y}, K);
        Q ← Q ∪ Sample(D_train{y}\S, L);
    end for
    H_S, H_Q ← BERT_φ(S), BERT_φ(Q);
    H_P ← ∅;
    for y ∈ Y do
        H_P ← H_P ∪ mean(H_S{y});
    end for
    for t ∈ T do
        Construct triples by H_S, H_P;
        Input triples to the triplet network;
        Calculate L_IT by Equation 5;
        Update θ to θ' by Equation 6;
    end for
    Construct triples by H_Q, H_P;
    Input triples to the triplet network;
    Calculate L_IT by Equation 5;
    Update φ and θ based on θ' by Equation 7;
end for
return φ, θ and M
```

## 5 Experiments

In this section, we comprehensively evaluate the performance of MeTNet in both in-domain and cross-domain settings. The in-domain setting indicates that both the training set and the test set come from the same domain, while the cross-domain setting indicates that they are from different domains.

### 5.1 Datesets

We use four public English datasets and one new Chinese dataset. Statistics of these datasets are given in Table 1. For the English datasets, they are FEW-NERD (Ding et al., 2021), WNUT17 (Derczynski et al., 2017), Restaurant (Liu et al., 2013) and Multiwoz (Budzianowski et al., 2018). Specifically, FEW-NERD designs an annotation schema of 8 coarse-grained (e.g., "Person") entity types and 66 fine-grained (e.g., "Person-Artist") entity types, and constructs two tasks. One is FEW-NERD-INTRA, where all the entities in the training set (source domain), validation set and test set (target domain) belong to different coarse-grained types.

| Datasets | # Sentences | # Entities | # Classes | Domain |
|---|---|---|---|---|
| FEW-COMM | 66.2k | 140.9k | 92 | Commodity |
| FEW-NERD | 188.2k | 491.7k | 66 | General |
| WNUT | 4.7k | 3.1k | 6 | Social Media |
| Restaurant | 9.2k | 15.3k | 8 | Review |
| Multiwoz | 23.0k | 20.7k | 14 | Dialogue |

Table 1: Statistics of datasets. # Classes corresponds to the number of pre-defined entity types in a dataset.

The other is FEW-NERD-INTER, where only the fine-grained entity types are mutually disjoint in different sets. We conduct in-domain experiments on both tasks. To further validate the model's generalizability on cross-domain tasks, we also use three NER datasets from different domains, namely WNUT17 (Social), Restaurant (Review) and Multiwoz (Dialogue).

We also construct and conduct experiments on a Chinese few-shot NER dataset, namely, FEW-COMM. The dataset consists of 66,165 product description texts that merchants display on a large e-commerce platform, including 140,936 entities and 92 pre-defined entity types. These entity types are various commodity attributes that are manually defined by domain experts, such as "material", "color" and "origin". Specifically, we first hire five well-trained annotators to label the texts in one month and then ask four domain experts to review and rectify the results. To the best of our knowledge, it is the first Chinese dataset specially constructed for few-shot NER. Due to the space limitation, please see Appendix A for more details on the dataset.

### 5.2 Baselines

We compare MeTNet with seven other few-shot NER models, which can be grouped into three categories: (1) *optimization-based methods*: MAML (Finn et al., 2017) which adapts to new classes by using support instances and optimizes the loss of the adapted model based on the query instances. (2) *nearest-neighbor-based methods*: NNShot (Yang and Katiyar, 2020) and StructShot (Yang and Katiyar, 2020). NNShot determines the tag of a query instance based on the word-level distance and StructShot further improves NNShot by an additional Viterbi decoder. (3) *prototype-based methods*: PROTO (Snell et al., 2017), CONTaiNER (Das et al., 2021), ESD (Wang et al., 2021) and DecomMETA (Ma et al., 2022). Specifically, PROTO computes the prototype vector by averaging all the sample embeddings in the support set for each class. CONTaiNER proposes a contrastive learning method that optimizes the inter-token distribution distance for few-shot NER. ESD uses various types of attention based on PROTO to improve the model performance. DecomMETA addresses few-shot NER by sequentially tackling few-shot span detection and few-shot entity typing using meta-learning.

### 5.3 Experiment Setup

We implemented MeTNet by PyTorch. The model is initialized by He initialization (He et al., 2015) and trained by AdamW (Loshchilov and Hutter, 2017). We run the model for 6,000 epochs with the learning rates 0.2 and the meta learning rate 0.0001 for the improved triplet loss on all the datasets. For the text encoder, we use the pre-trained `bert-base-Chinese` model for the FEW-COMM dataset and `bert-base-uncased` model for other datasets. In the triplet network, we use three same fully connected layer with shared parameters and we set the dimensionality of the fully connected layer to 1024. We also fine-tune the number $T$ of iterations for updating parameters on the support set in each meta-task by grid search over $\{1, 3, 5, 7, 9\}$ and set it to 3 on all the datasets. For a fair comparison, we substitute the text encoder as that of MeTNet for all the baselines, use the original codes released by their authors and fine-tune the parameters of the models. We run all the experiments on a single NVIDIA v100 GPU. Following Ding et al. (2021), we evaluate the model performance based on 500 meta-tasks in meta-testing and report the average micro F1-score over 5 runs. We utilize the `IO` schema in our experiments, using `I-type` to denote all the words of a named entity and `O` to denote other words.

### 5.4 Results

**In-domain Experiments** The results of in-domain experiments in 1-shot and 5-shot settings on FEW-NERD dataset are shown in Table 2. From the table, MeTNet consistently outperforms all the baselines on the average F1 score. For example, compared with DecomMETA, MeTNet achieves $1.84\%$ improvements on the average F1 score; when compared against the PROTO model, MeTNet leads by $31.04\%$ on the average F1 score, which clearly demonstrates that our model is very effective in improving PROTO. On the FEW-COMM dataset (as shown in Table 3), our model also achieves the best performance across all the settings. All these results show that MeTNet, which learns adaptive margins for inference by an improved triplet network, can perform reasonably well.

**Cross-domain Experiments** We train models on FEW-NERD-INTER (General) as the source domain and test our models on WNUT (Social Media), Restaurant (Review) and Multiwoz (Dia-

| Method | FEW-NERD-INTER | | | | FEW-NERD-INTRA | | | | Average |
|---|---|---|---|---|---|---|---|---|---|
| | 5-1 | 5-5 | 10-1 | 10-5 | 5-1 | 5-5 | 10-1 | 10-5 | |
| MAML | 38.52±0.67 | 49.86±0.33 | 30.20±0.78 | 33.39±0.49 | 30.14±0.53 | 38.38±0.41 | 23.05±0.45 | 28.52±0.59 | 34.01 |
| NNShot | 55.24±0.40 | 54.49±0.91 | 40.21±1.63 | 49.23±1.15 | 26.30±1.21 | 38.91±0.53 | 24.69±0.23 | 32.63±2.59 | 40.21 |
| StructShot | 53.65±0.54 | 56.50±1.17 | 46.86±0.53 | 53.25±0.97 | 30.88±0.96 | 42.80±0.51 | 27.25±0.84 | 33.56±1.06 | 43.10 |
| PROTO | 35.78±0.71 | 47.01±1.31 | 30.12±0.77 | 47.13±0.57 | 15.68±0.92 | 36.58±0.87 | 12.68±0.59 | 28.99±1.06 | 31.75 |
| CONTaiNER† | 55.95 | 61.83 | 48.35 | 57.12 | 40.43 | 53.70 | 33.84 | 47.49 | 49.84 |
| ESD† | 66.46±0.49 | **74.14±0.80** | 59.95±0.69 | 67.91±1.41 | 41.44±1.16 | 50.68±0.94 | 32.29±1.10 | 42.92±0.75 | 54.47 |
| DecomMETA† | 68.77±0.24 | 71.62±0.16 | 63.26±0.40 | 68.32±0.10 | 52.04±0.44 | **63.23±0.45** | 43.50±0.59 | 56.84±0.14 | 60.95 |
| **MeTNet** | **70.12±0.63** | 73.30±0.54 | **65.97±0.69** | **71.47±0.61** | **54.59±0.83** | 62.53±0.53 | **46.80±0.91** | **57.51±0.87** | **62.79** |

Table 2: F1 scores (%) of 5-way 1-shot, 5-way 5-shot, 10-way 1-shot and 10-way 5-shot problems over FEW-NERD dataset. † denotes the results reported in Ma et al. (2022). We highlight the best results in bold.

| Method | FEW-COMM | | | |
|---|---|---|---|---|
| | 5-1 | 5-5 | 10-1 | 10-5 |
| MAML | 28.16±0.57 | 54.38±0.37 | 26.23±0.61 | 44.66±0.44 |
| NNShot | 48.40±1.27 | 71.55±1.37 | 41.75±0.93 | 67.91±1.51 |
| StructShot | 48.61±0.76 | 70.62±0.83 | 47.77±0.83 | 65.09±0.97 |
| PROTO | 22.73±0.86 | 53.95±0.98 | 22.17±0.90 | 45.81±0.99 |
| CONTaiNER | 57.13±0.47 | 63.38±0.68 | 51.87±0.58 | 60.98±0.71 |
| ESD | 65.37±0.79 | 73.29±0.95 | 58.32±0.89 | 70.93±1.01 |
| DecomMETA | 68.01±0.39 | 72.89±0.45 | 62.13±0.28 | 72.14±0.11 |
| **MeTNet** | **70.10±0.58** | **76.74±0.58** | **64.05±0.74** | **76.28±0.91** |

Table 3: F1 scores (%) of 5-way 1-shot, 5-way 5-shot, 10-way 1-shot and 10-way 5-shot problems over FEW-COMM dataset. We highlight the best results in bold.

logue), respectively. All the three datasets are in different domains from that of FEW-NERD-INTER. Since there is a large generalization gap between the training and test distributions, cross-domain experiments are generally more challenging than in-domain ones. Table 4 shows the results. From the table, we see that our model performs very well in both the 1-shot and 5-shot settings. This clearly shows the generalizability of our model.

## 5.5 Ablation Study

We conduct an ablation study to understand the characteristics of the main components of MeTNet. To show the importance of the proposed margin-based inference method, one variant generates prototype vectors for both entity types and O-class. In the inference stage, it computes the distance between a query instance and all these prototype vectors, and predict the query instance to be in the class with the smallest distance, which is similar as previous methods. We call this variant **MeTNet-piw** (use **p**revious **i**nference **w**ay). To study the importance of the triplet network in mapping prototype vectors and samples into a low-dimensional space that is easier to classify, we further remove the triplet network and replace it with a fully-connected layer. Due to the removal of the triplet network, adaptive margins cannot be learned, so we adopt the same inference procedure as in MeTNet-piw. We call this variant **MeTNet-piw-rtn** (use **p**revious **i**nference **w**ay and **r**emove **t**riplet **n**etwork ). To show the importance of the improved triplet loss, we replace it with the original triplet loss and call this variant **MeTNet-otl** (**o**riginal **t**riplet **l**oss). Finally, we remove the MAML training procedure to explore the impact of MAML on the model and call this variant **MeTNet-w/o-MAML**.

The results of ablation study are shown in Table 5. From the table, we observe: (1) MeTNet beats MeTNet-piw clearly. For example, in 5-way 1-shot problem on the FEW-COMM dataset, the F1 score of MeTNet is 66.10% while that of MeTNet-piw is only 54.66%. This shows that the margin-based inference can effectively enhance the model performance. (2) The advantage of MeTNet-piw over MeTNet-piw-rtn across all the datasets further shows that the triplet network can learn better embeddings for samples with different classes in the low-dimensional space. (3) MeTNet leads MeTNet-otl in all the classification tasks. This demonstrates that our improved triplet loss is highly effective. (4) Compared against MeTNet-w/o-MAML, MeTNet leads by 3.4% on the average F1 score, which shows the importance of MAML to the model.

## 5.6 Visualization

Figure 4 visualizes the word-level representations of a query set generated by PROTO and MeTNet in the 5-way 1-shot and 5-way 5-shot settings on the FEW-NERD-INTER dataset. Note that PROTO generates prototype vectors for both entity types and O-class, while MeTNet only generates that for entity types. From the figure, we see that words in O-class are widely distributed, so using a prototype vector to represent O-class is insufficient. For those

| Method | WNUT | | Restaurant | | Multiwoz | | Average | |
|---|---|---|---|---|---|---|---|---|
| | 5-1 | 5-5 | 5-1 | 5-5 | 5-1 | 5-5 | 5-1 | 5-5 |
| MAML | 17.77±0.67 | 23.69±0.71 | 17.53±0.83 | 22.81±0.77 | 20.82±1.01 | 23.61±0.87 | 18.71 | 23.37 |
| NNShot | 15.93±0.61 | 23.78±0.67 | 19.37±0.73 | 32.83±0.89 | 27.77±0.91 | 42.19±1.03 | 21.02 | 32.93 |
| StructShot | 17.29±1.01 | 25.18±0.96 | 20.75±1.07 | 34.18±1.18 | 30.79±1.21 | 44.01±1.31 | 22.46 | 34.08 |
| PROTO | 13.04±0.71 | 23.20±0.93 | 15.68±1.01 | 32.71±1.07 | 22.09±0.81 | 41.78±0.79 | 16.94 | 32.56 |
| CONTaiNER | 18.15±1.17 | 19.54±1.09 | 27.74±0.89 | 33.41±0.97 | 34.88±2.03 | 41.92±1.93 | 26.92 | 31.62 |
| ESD | 19.24±0.87 | 26.00±0.96 | 24.53±1.03 | 37.85±0.97 | 35.81±1.87 | 42.88±1.05 | 26.53 | 35.58 |
| DecomMETA | 20.98±0.11 | 31.17±0.16 | 29.75±0.27 | 41.13±0.19 | 33.79±0.22 | 47.01±0.36 | 28.17 | 39.77 |
| **MeTNet** | **21.84**±0.88 | **33.27**±0.75 | **31.80**±0.67 | **45.53**±0.74 | **39.88**±0.93 | **52.13**±0.91 | **31.17** | **43.64** |

Table 4: F1 scores (%) of 5-way 1-shot, 5-way 5-shot problems over three datasets for cross-domain experiments. We highlight the best results in bold.

| Method | FEW-NERD-INTER | | | | FEW-NERD-INTRA | | | | FEW-COMM | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 5-1 | 5-5 | 10-1 | 10-5 | 5-1 | 5-5 | 10-1 | 10-5 | 5-1 | 5-5 | 10-1 | 10-5 |
| MeTNet-piw | 60.35 | 62.75 | 53.89 | 55.59 | 42.92 | 50.84 | 33.31 | 40.40 | 58.66 | 66.33 | 49.75 | 64.11 |
| MeTNet-piw-rtn | 50.77 | 62.14 | 41.29 | 54.87 | 34.19 | 46.56 | 28.65 | 38.42 | 51.34 | 61.58 | 45.61 | 61.65 |
| MeTNet-otl | 65.37 | 70.61 | 55.60 | 63.81 | 44.78 | 57.32 | 36.27 | 44.96 | 62.66 | 73.17 | 54.76 | 72.70 |
| MeTNet-w/o-MAML | 68.14 | 70.27 | 63.73 | 69.32 | 52.37 | 58.13 | 43.28 | 51.31 | 68.19 | 71.73 | 59.85 | 72.81 |
| **MeTNet** | **70.12** | **73.30** | **65.97** | **71.47** | **54.59** | **62.53** | **46.80** | **57.51** | **70.10** | **76.74** | **64.05** | **76.28** |

Table 5: Ablation study: F1 scores (%) of 5-way 1-shot, 5-way 5-shot, 10-way 1-shot and 10-way 5-shot classification over FEW-NERD and FEW-COMM datasets. 'rtn' means removing triplet network, 'piw' means using previous inference way and 'otl' means using original triplet loss. We highlight the best results in bold.

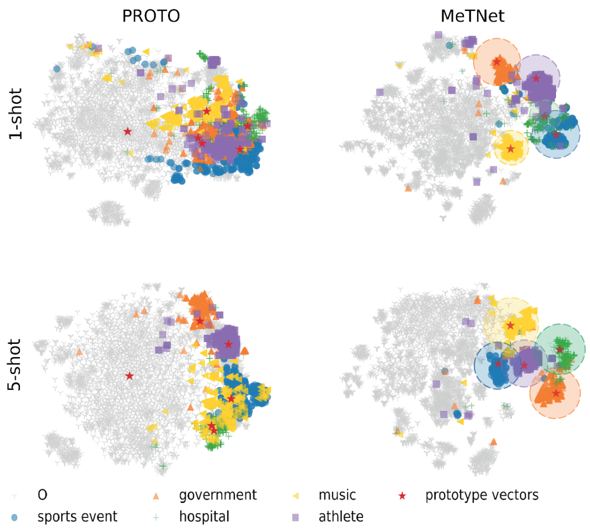


Figure 4: t-SNE visualizations on the FEW-NERD-INTER test sets. The representations are obtained from PROTO and MeTNet. The dashed circles represent the regions determined by adaptive margins.

samples closer to other prototype vectors, they are easily misclassified. Instead of representing O-class with a prototype vector, MeTNet addresses the problem by learning adaptive margins for entity types only and using a margin-controlled region to make prediction. Samples outside these regions are labeled with O-class. Further, our method MeTNet can generate word embeddings that are clearly separated, which further explains the effectiveness of MeTNet.

## 6 Conclusion

In this paper, we studied the few-shot NER problem and proposed MeTNet, which is a meta-learning triplet network with adaptive margins. As a prototype-based method, MeTNet uses a triplet network to map samples and prototype vectors into a low-dimensional space that is easier to be classified. Further, to solve the problem that O-class is semantically complex and thus hard to be represented by a prototype vector, we designed an improved triplet loss function with adaptive margins and presented a margin-based inference procedure to predict the label of a query instance. We performed extensive experiments in both in-domain and cross-domain settings. Experimental results show that MeTNet can achieve significant performance gains over other state-of-the-art methods. In particular, we released the first Chinese few-shot NER dataset from a large-scale e-commerce platform, which aims to provide more insight for future study on few-shot NER.

## Ethics Statement

The proposed method has no obvious potential risks. All the scientific artifacts used/created are properly cited/licensed, and the usage is consistent with their intended use. The paper collects a new dataset FEW-COMM, which does not contain any sensitive information. The dataset is keeping with the rules and reviewed by experts to ensure that it does not create additional risks. Also, we open up our codes and hyperparameters to facilitate future reproduction without repeated energy cost.

## References


Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz–a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.

Xiang Chen, Ningyu Zhang, Lei Li, Xin Xie, Shumin Deng, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2021. Lightner: A lightweight generative framework with prompt-guided attention for low-resource ner. *arXiv preprint arXiv:2109.00720*.

Ganqu Cui, Shengding Hu, Ning Ding, Longtao Huang, and Zhiyuan Liu. 2022. Prototypical verbalizer for prompt-based few-shot tuning. *arXiv preprint arXiv:2203.09770*.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using bart. In *Findings of ACL*, pages 1835–1845.

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

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the wnut2017 shared task on novel and emerging entity recognition. In *W-NUT*, pages 140–147.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. 2021. Few-nerd: A few-shot named entity recognition dataset. In *ACL*, pages 3198–3213.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pages 1126–1135.

Alexander Fritzler, Varvara Logacheva, and Maksim Kretov. 2019. Few-shot classification in named entity recognition task. In *SAC*, pages 993–1000.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. 2018. Conditional neural processes. In *ICML*, pages 1704–1713.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034.

Elad Hoffer and Nir Ailon. 2015. Deep metric learning using triplet network. In *SIMBAD*, pages 84–92.

Timothy M. Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. 2020. Meta-learning in neural networks: A survey. *CoRR*, abs/2004.05439.

Yutai Hou, Wanxiang Che, Yongkui Lai, Zhihan Zhou, Yijia Liu, Han Liu, and Ting Liu. 2020. Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network. In *ACL*, pages 1381–1393.

Jiaxin Huang, Chunyuan Li, Krishan Subudhi, Damien Jose, Shobana Balakrishnan, Weizhu Chen, Baolin Peng, Jianfeng Gao, and Jiawei Han. 2021. Few-shot named entity recognition: An empirical baseline study. In *EMNLP*, pages 10408–10423.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2.

Jingjing Liu, Panupong Pasupat, Scott Cyphers, and Jim Glass. 2013. Asgard: A portable architecture for multilingual dialogue systems. In *ICASSP*, pages 8386–8390.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Ruotian Ma, Xin Zhou, Tao Gui, Yiding Tan, Qi Zhang, and Xuanjing Huang. 2021. Template-free prompt tuning for few-shot ner. *arXiv preprint arXiv:2109.13532*.

Tingting Ma, Huiqiang Jiang, Qianhui Wu, Tiejun Zhao, and Chin-Yew Lin. 2022. Decomposed meta-learning for few-shot named entity recognition. *arXiv preprint arXiv:2204.05751*.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2017. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. In *ICML*, pages 2554–2563.

Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*.

Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. 2012. Open domain event extraction from twitter. In *KDD*, pages 1104–1112.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *ICML*, pages 1842–1850.

Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. In *NIPS*, pages 4077–4087.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *CVPR*, pages 1199–1208.

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. *arXiv preprint arXiv:2106.15167*.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*.

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

Sam Wiseman and Karl Stratos. 2019. Label-agnostic sequence labeling by copying nearest neighbors. In *ACL*, pages 5363–5369.

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

# A FEW-COMM

## A.1 Entity types

As introduced in Section 5.1 of the main text, FEW-COMM is manually annotated with 92 pre-defined entity types, and we list all the types and the number of samples belonging to each type in Table 6. We find that since FEW-COMM is collected from real application scenarios, there is a long-tailed distribution problem, which is a common problem in real scenarios. How to overcome the influence of long-tailed distribution on the model is a crucial research direction.

## A.2 Splits

We divided the training set, validation set and test set in a ratio of 6:2:2. Among them, the training set includes 55 entity types, the validation set includes 18 entity types, and the test set includes 19 entity types. The entity types contained in the three sets are disjoint.

## A.3 Examples

We provide some examples on FEW-COMM dataset for further understanding, which is shown in Table 7.

Table 6: All the pre-defined entity types and the number of samples belonging to each type in FEW-COMM dataset.

| Entity types | # Samples | Entity types | # Samples | Entity types | # Samples | Entity types | # Samples |
|---|---|---|---|---|---|---|---|
| 其他属性 | 44259 | 功能功效 | 13412 | 材质 | 11126 | 适用人群 | 9483 |
| 颜色 | 6955 | 产地 | 4959 | 适用对象 | 2520 | 成分 | 2356 |
| 适用季节 | 1791 | 品质等级 | 1671 | 接口 | 1379 | 适用时间 | 1292 |
| 运输服务 | 1245 | 型号 | 1210 | 商品特色 | 1135 | 国产/进口 | 920 |
| 分类 | 897 | 形状形态 | 874 | 香型 | 860 | 组合形式 | 808 |
| 适用性别 | 801 | 连接方式 | 786 | 控制方式 | 706 | 领型 | 697 |
| 甜度 | 674 | 适用品牌 | 636 | 送礼对象 | 614 | 供电方式 | 585 |
| 面料材质 | 569 | 风味 | 564 | 大小 | 550 | 口感 | 546 |
| 系列 | 530 | 筒高 | 510 | 造型 | 503 | 厚度 | 486 |
| 是否有机 | 483 | 技术类型 | 478 | 厚薄 | 472 | 填充材质 | 469 |
| 适用运营商 | 466 | 袖长 | 465 | 适用车型 | 462 | 糖含量 | 460 |
| 光度 | 457 | 脂肪含量 | 456 | 是否带盖 | 451 | 加热方式 | 447 |
| 长短 | 444 | 版型 | 441 | 适用衣物 | 440 | 资质认证 | 439 |
| 外观 | 436 | 消毒方式 | 430 | 是否清真 | 430 | 部位 | 428 |
| 是否净洗 | 426 | 长度 | 426 | 适用生肖 | 426 | 配件类型 | 424 |
| 袖型 | 422 | 果肉颜色 | 419 | 适用空间 | 419 | 适用燃料 | 416 |
| 适用星座 | 415 | 酸碱度 | 413 | 剂型 | 413 | 锅底类型 | 412 |
| 销售方式 | 412 | 鞋垫材质 | 410 | 适用人数 | 406 | 裙型 | 404 |
| 定制服务 | 403 | 存储容量 | 403 | 成熟状态 | 403 | 是否去皮 | 402 |
| 是否去骨 | 402 | 冲泡方式 | 402 | 赠品 | 401 | 宽度 | 401 |
| 裤长 | 401 | 粗细 | 401 | 礼盒类型 | 400 | 结构 | 400 |
| 色系 | 399 | 净含量 | 376 | 发酵程度 | 321 | 抽数 | 214 |
| 保质期 | 86 | 内容 | 44 | 段位 | 40 | 装订方式 | 11 |

Table 7: Examples in FEW-COMM dataset. We marked the entities with the corresponding entity types.

| |
|---|
| 日本[产地] 黑色[颜色] 数字帆布[材质] 烧饼包灯芯绒[材质] 钱包证件包对开简约大容量[功能功效] 笔袋 |
| 春夏[适用季节] 爆款纯色[颜色] 男女通用[适用性别] 防晒冰袖套跑男[其他属性] 骑行紫外线护臂【蓝色[颜色] 直筒[版型] 无指盒装】 |
| 洁丽雅（grace）浴巾a类纯棉[材质] 加大加厚[其他属性] 成人[适用人群] 家用柔软吸水[功能功效] |
| 精品[品质等级] 霏慕情趣内衣女式[适用性别] 性感透明[颜色] 诱惑镂空蕾丝[材质] 刺绣薄纱[其他属性] 7114/2 |
| 金丝绒[材质] 阔腿裤秋冬[适用季节] 加绒[其他属性] 高腰垂感宽松直筒[版型] 显瘦[功能功效] 百搭休闲拖地长裤子 |
| 绳子拉车绳货车[适用车型] 刹车绳子捆绑带拖车绳紧绳器马扎耐磨[功能功效] 尼龙[材质] 扁带拉紧加粗20米 |
| 情趣丝袜修腿显瘦[功能功效] 蕾丝[材质] 花边白色长筒丝袜高筒[筒高] 情趣连体袜子 |
| 电动电瓶车头盔灰[颜色] 女士[适用性别] 夏季[适用季节] 半盔防晒全盔可爱夏天轻便[其他属性] 安全帽/个 |
| 棉拖鞋女士[适用性别] 家居室内厚底防滑月子鞋冬季[适用季节] 毛绒[材质] 保暖情侣[适用人群] 棉鞋红色[颜色] |
| 时尚布艺围裙厨房无袖[袖长] 口袋围腰成人[适用人群] 格子围裙 |
| 【心中最爱】-33朵玫瑰爱心礼盒[形状形态] 鲜花-送爱人[送礼对象] 花店送花上门[运输服务] |
| 泳帽女士[适用性别] 长发防水[功能功效] 护耳游泳硅胶[材质] 布帽舒适[其他属性] 不勒头帽子游泳泡温泉1个 |
| airism宽松圆领[领型] t恤( 五分袖[袖长] 黑色[颜色] ) |
| 中啡冷萃[冲泡方式] 速溶即溶纯黑[颜色] 小罐胶囊2gx16颗/盒 |