TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 14/14 pp976–987
DOI: 10.26599/TST.2022.9010043
Volume 28, Number 5, October 2023

# Decoupled Two-Phase Framework for Class-Incremental Few-Shot Named Entity Recognition

Yifan Chen, Zhen Huang*, Minghao Hu*, Dongsheng Li, Changjian Wang, Feng Liu, and Xicheng Lu

**Abstract:** Class-Incremental Few-Shot Named Entity Recognition (CIFNER) aims to identify entity categories that have appeared with only a few newly added (novel) class examples. However, existing class-incremental methods typically introduce new parameters to adapt to new classes and treat all information equally, resulting in poor generalization. Meanwhile, few-shot methods necessitate samples for all observed classes, making them difficult to transfer into a class-incremental setting. Thus, a decoupled two-phase framework method for the CIFNER task is proposed to address the above issues. The whole task is converted to two separate tasks named Entity Span Detection (ESD) and Entity Class Discrimination (ECD) that leverage parameter-cloning and label-fusion to learn different levels of knowledge separately, such as class-generic knowledge and class-specific knowledge. Moreover, different variants, such as the Conditional Random Field-based (CRF-based), word-pair-based methods in ESD module, and add-based, Natural Language Inference-based (NLI-based) and prompt-based methods in ECD module, are investigated to demonstrate the generalizability of the decoupled framework. Extensive experiments on the three Named Entity Recognition (NER) datasets reveal that our method achieves the state-of-the-art performance in the CIFNER setting.

**Key words:** named entity recognition; deep learning; class-incremental learning; few-shot learning

## 1 Introduction

Named Entity Recognition (NER) is a fundamental task in natural language processing that aims to identify and classify entities in a given sentence. Recently, deep neural networks have achieved promising performance in NER tasks[1–6]. However, these methods typically rely on large-scale labeled samples, and entity categories cannot be modified in real-time. Contrarily, humans can recognize new entities incrementally while retaining previous knowledge[7, 8] and require only a few examples per class. To fill this research gap, we study a practical yet challenging NER setting called Class-Incremental Few-Shot NER (CIFNER) (see Fig. 1). Different from traditional NER tasks, CIFNER requires the following: (1) the NER model would be first trained on base classes with enough training examples; (2) after training with only a few annotated examples on novel classes, the model is expected to perform well on all seen classes.

Recent research has divided the CIFNER task into two parts: Class-Incremental NER (CINER)[9, 10], which adds new classifiers or extends the dimension of old classifier to adapt to new entity classes[10]; and few-shot NER[11–13], which usually retrains the model with all seen data through meta-learning or prompt learning. However, these two group methods cannot be used directly on CIFNER: (1) current CINER methods introduce new parameters to fit novel entity classes[10],

• Yifan Chen, Zhen Huang, Dongsheng Li, Changjian Wang, Feng Liu, and Xicheng Lu are with the College of Computer, National University of Defense Technology, Changsha 410073, China. E-mail: chenyifan@nudt.edu.cn; huangzhen@nudt.edu.cn; dsli@nudt.edu.cn; wangchangjian@nudt.edu.cn; richardlf@nudt.edu.cn; xclu@nudt.edu.cn.
• Minghao Hu is with the Information Research Center of Military Science, PLA Academy of Military Science, Beijing 100091, China. E-mail: huminghao16@gmail.com.
∗ To whom correspondence should be addressed.
Manuscript received: 2022-08-23; accepted: 2022-09-30.

© The author(s) 2023. The articles published in this open access journal are distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/).

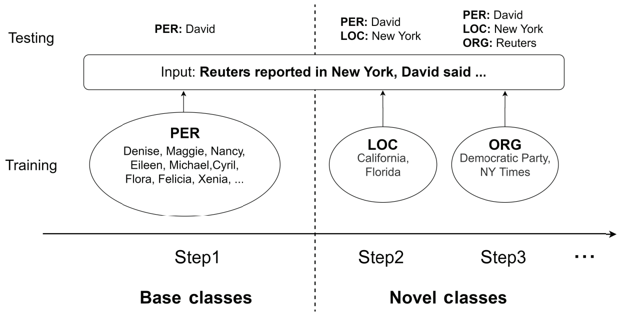



**Fig. 1 Overview of the CIFNER task. Different from CINER, the first step has enough data to train, whereas the incremental steps are all few-shot.**

but these parameters cannot be trained well with only a few new data. Moreover, most CINER methods cannot distinguish which part of parameters should be kept and which part should be updated repeatedly, given that they view knowledge added at the current step equally; (2) few-shot NER methods presume that a class can appear multiple times in different episodes[14] or all classes arrive simultaneously[15], which are different from the class-incremental few-shot scenario.

To solve the above problems, we propose a Decoupled Two-Phase Framework (DTPF), which contains Entity Span Detection (ESD) and Entity Class Discrimination (ECD) to learn class-generic knowledge and class-specific knowledge individually. The ESD module only extracts spans from given sentences to avoid importing new parameters. This module can gain class-generic knowledge because the different classes usually have similar boundary feature. The ECD module is dedicated to classifying extracted entity spans into different classes to specifically learn class-specific knowledge. For both two phases, we adopt parameter-cloning and label-fusion methods to prevent catastrophic forgetting without retaining old data. Moreover, to verify the generalizability of our proposed framework, we explore the conditional random field-based (CRF-based) method and the word-pair-based method in the ESD phase and try three variants in the ECD phase, namely add-based method, natural language inference-based (NLI-based) method and prompt-based method. Our research contributions in this paper can be summarized as follows:

• We propose a DTPF for the CIFNER task to learn class-generic knowledge and class-specific knowledge respectively.

• We explore two variants for the ESD phase and three for the ECD phase, proving the generalizability of our proposed framework.

• We perform substantial experiments on the three NER datasets (Conll2003[16], OntoNotes[17], and Few-NERD[14]). The results show that our proposed approach outperforms existing baselines by a significant margin.

The rest of the paper is organized as follows: Section 2 reviews existing NER models about CIFNER. Section 3 introduces the problem and the definition of CIFNER and describes our proposed framework in detail. Section 4 presents the experimental results. Section 5 concludes this paper.

## 2 Related Work

**Class-Incremental NER**. Incremental learning (also called lifelong learning, continual learning)[18] aims to learn knowledge from a sequence of tasks. Incremental learning has two types: (1) Task-Incremental Learning (TIL) is to learn tasks with different task classes sequentially and predict with task indexes[19, 20]. (2) Class-Incremental Learning (CIL)[21–23] focuses on distinguishing all the seen classes after learning from different subtasks with the same task class. Recently, knowledge distillation has been adopted for CINER[9, 10] to alleviate the catastrophic forgetting problem[7, 8]. However, existing methods of CINER require large-scale labeled data of the novel classes to train the model, which is impractical for real-world applications.

**Few-Shot NER**. Few-shot learning[24–26] is usually modeled as an N-way K-shot problem, and studies on few-shot NER typically adopt meta-learning-based approaches at either the token level[13, 27] or the span level[12, 28]. However, these meta-learning methods with episode training assume that a class can appear multiple times in different episodes. Recently, prompt learning[29, 30] has made great progress in few-shot learning, which introduces prompt information to make better utilize of the structure and prior knowledge of the pre-trained model. However, few-shot NER based on prompt learning[15, 31] presumes all the classes arrive simultaneously, and the dataset has the labels for all classes. As a result, prompt learning is also unable to meet the needs of categories that change over time.

**Class-Incremental Few-Shot NER**. Class-incremental few-shot learning[32–34] aims to identify entities of the categories that have appeared with only a few newly added (novel) class examples, which has recently received great attention. In NER, Wang et al.[35] designed a distillation method for the CRF-based NER models and used data-free distillation to transfer knowledge from the previous model to the current model. In contrast to this method, we seek a general method that applies not only to the CRF-based model but also to models based on other NER decoders, such as the word-pair decoder[36]. The NER model is decoupled into two easier phases to explicitly distinguish class-generic knowledge from class-specific knowledge, and several variants are investigated to validate the generalizability of our proposed framework.

In summary, existing works in CINER that have sufficient training data are primarily concerned with preventing catastrophic forgetting of previous entity knowledge when the model is trained on new tasks. Few-shot learning works almost entirely on prior knowledge to reduce labeled data in training. In contrast to these works, we aim to solve a more challenging and practical problem, CIFNER, which requires the model to incrementally learn novel entity information and retain previous entity knowledge from a sequence of few-shot NER tasks incrementally by using a decomposed two-phase framework.

## 3 Methodology

In this section, the definition of the CIFNER problem is formalized, and our proposed two-phase approach is introduced in detail.

### 3.1 Problem definitions

CIFNER is defined as follows. Assume there is a sequence of subtasks $T = (T_1, T_2, \dots, T_n)$, where each subtask $T_t$ has its dataset $D_t$ splitting into a training set $D_t^{\text{train}}$ and a validation set $D_t^{\text{val}}$. $D_t$ contains samples $(X_d, Y_d), d \in [1, |D_t|]$ and the labels in $Y_t$ belong to the incremental classes set $C^{\text{new}}$ of the subtask $T_t$. Particularly, the first subtask $T_1$ has sufficient data for training, whereas the subsequent new tasks are few-shot. The classes for all the previous $t-1$ subtasks are denoted as $C_{t-1} = \{c_1, c_2, \dots, c_o\}$, and the incremental classes for the new subtask $T_t$ are denoted as $C^{\text{new}} = \{c_{o+1}, c_{o+2}, \dots, c_{o+p}\}$. Thus, $C_t = C_{t-1} + C^{\text{new}} = \{c_1, c_2, \dots, c_{o+p}\}$ means the whole entity classes appearing at the previous $t$ subtasks. Then, we suppose the model $M_1$ performs well for subtask $T_1$ with sufficient data. At the time step $t$, the model $M_t$ is trained on the few data of $T_t$ and expected to have a good performance on $C^{\text{new}}$ for $T_t$ and $C_{t-1}$ for the previous $t-1$ subtasks.

### 3.2 Overall architecture

To design a general CIFNER framework and to learn class-generic and class-specific knowledge from various angles, we build a decoupled two-phase NER framework to solve the CIFNER task. The first phase is ESD, which extracts spans from the row sentence to consummate the class-generic knowledge. The second phase is ECD, which divides the extracted spans into different classes to distinguish class-specific knowledge. To avoid catastrophic forgetting, we use parameter-cloning and label-fusion methods in both phases. Furthermore, we test several variants to ensure that our framework is general. Figure 2 depicts a high-level overview of our

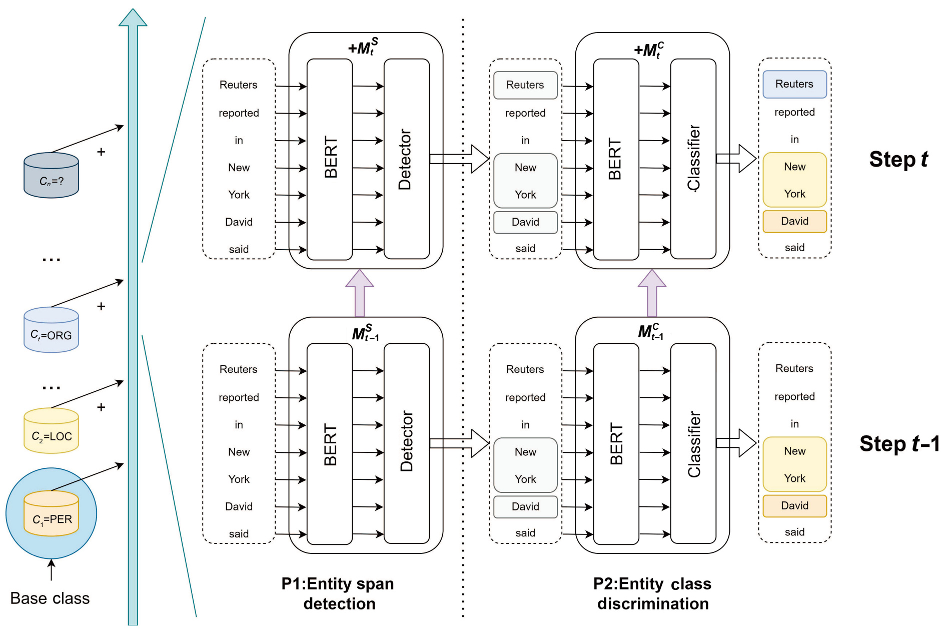



**Fig. 2 Overall architecture of the proposed approach for CIFNER. The first phase of ESD judges whether the candidate span is an entity, and the second phase of ECD determines the type of the extracted entity.**

decoupled framework.

### 3.3 Entity span detection

The ESD module, which takes row sentences as input and predicts candidate entity spans, aims to learn class-generic knowledge of entity boundaries. Given a sentence $X = \{x_1, x_2, \ldots, x_n\}$, the ESD module aims to find entity spans $s_{ij} \in \{(i,j)|i,j \in [0,n], j \geqslant i\}$ from $X$, where $S$ is the set of $s_{ij}$. Given that most recent NER models adopt CRF-based[6] or word-pair-based[36] decoding methods, we explore two different variants in the ESD module.

#### 3.3.1 CRF-based method

The first variant of the ESD module is the CRF-based method, which is inspired by the BERT-CRF model[37]. We first feed $X$ into a pre-trained BERT[2] encoder to obtain token representations $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \in \mathbb{R}^{h\times n}$. Then, we use a linear projection with a CRF[38] on $\boldsymbol{X}$ to obtain the final tags. Note that CRF is applied as a post-process, where transition probabilities have been hard coded to prevent impossible transitions, following ExtendNER[10].

In step $t$, we adopt knowledge distillation to keep knowledge learned in the previous $t-1$ steps. Specifically, for the token labeled as $y = O$, we use Kullback-Leibler (KL) divergence to calculate the loss $L_{\text{KD}}$ between the output distribution of the teacher model $M_{t-1}^S$ and the student model $M_t^S$. Meanwhile, the Cross-Entropy (CE) loss $L_{\text{CE}}$ will be applied when the token is labeled as $y \neq O$. The model will be trained by weighting the two losses:

$$L^S = \alpha L_{\text{CE}} + (1-\alpha) L_{\text{KD}} \tag{1}$$

where $\alpha$ is a hyperparameter to weight the contribution of the two losses.

#### 3.3.2 Word-pair-based method

Recently, word-pair-based methods[36] have reached promising performance in NER. To avoid introducing additional parameters in the incremental processing, we use one of the word-pair-based methods, GlobalPointer[39], to decode token representations $\boldsymbol{X}$. Specifically, we obtain two sequences $\boldsymbol{Q} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n\}$ and $\boldsymbol{K} = \{\boldsymbol{k}_1 \boldsymbol{k}_2, \ldots, \boldsymbol{k}_n\}$ by $\boldsymbol{q}_i = \boldsymbol{W}_q \boldsymbol{x}_i$ and $\boldsymbol{k}_i = \boldsymbol{W}_k \boldsymbol{x}_i$, where $\boldsymbol{W}_q$ and $\boldsymbol{W}_k$ are the learnable parameters. Then, we calculate the score for the span $s_{ij}$ as $s(i,j) = (\boldsymbol{R}_i \boldsymbol{q}_i)^{\text{T}} (\boldsymbol{R}_j \boldsymbol{k}_j) = \boldsymbol{q}_i^{\text{T}} \boldsymbol{R}_{j-i} \boldsymbol{k}_j$, where $\boldsymbol{R}_i$[40] is a type of relative position encoding which satisfies $\boldsymbol{R}_i^{\text{T}} \boldsymbol{R}_j = \boldsymbol{R}_{j-i}$. Eventually, the sentence $X$ can be transformed into an $n \times n$ matrix:

$$S = \begin{pmatrix} s(1,1) & s(1,2) & \ldots & s(1,n) \\ s(2,1) & s(2,2) & \ldots & s(2,n) \\ \vdots & \vdots & \ddots & \vdots \\ s(n,1) & s(n,2) & \ldots & s(n,n) \end{pmatrix} \tag{2}$$

The span $s_{ij}$ would be detected as an entity only if $s(i,j) > 0$ and $j \geqslant i$.

After the $t-1$ step, we would get a trained ESD module $M_{t-1}^S$. Using $M_{t-1}^S$ for $C_{t-1}$ and $D_t$ for $C^{\text{new}}$, we could train a new ESD module $M_t^S$ with $C_t$ to recognize all spans. We make the new model $M_t^S$ have the same layers as $M_{t-1}^S$. $M_t^S$ is a clone of $M_{t-1}^S$ for initializing parameters without adding new parameters.

In the word-pair-based method, we denote $\boldsymbol{Y}(S) \in \mathbb{R}^{n\times n}$ as the true label for $X$ on new classes $C^{\text{new}}$. The item $y_{ij}(S) \in \{0,1\}$ in $\boldsymbol{Y}(S)$ represents whether the span $s_{ij}$ belongs to $C^{\text{new}}$. Given that the new sentence only has labeled data for novel classes, training directly with the true label $\boldsymbol{Y}(S)$ will suffer from catastrophic forgetting even with a good parameter initialization. To alleviate this problem, we propose a simple label-fusion method following Li et al.[41]. Specifically, we first predict the sentence $X$ with $M_{t-1}^S$ to obtain the hard label $\boldsymbol{Y}'(S) \in \mathbb{R}^{n\times n}$ and further combine $\boldsymbol{Y}'(S)$ with the true label $\boldsymbol{Y}(S)$. The principle of combination is as follows:

$$y''_{ij}(S) = y'_{ij}(S) \vee y_{ij}(S) \tag{3}$$

where $y''_{ij}(S) \in \{0,1\}$ is an element of the gold label $\boldsymbol{Y}^c(S)$ being used to train $M_t^S$. That is, $M_t^S$ is trained by minimizing the loss function:

$$\begin{aligned} L^S = &\log(1 + \sum_{y''_{ij}(S)=1} \mathrm{e}^{-s(i,j)}) + \\ &\log(1 + \sum_{y''_{ij}(S)=0} \mathrm{e}^{s(i,j)}) \end{aligned} \tag{4}$$

where we add 1 to ensure $L^S > 0$.

### 3.4 Entity class discrimination

The ECD aims to discriminate the classes of detected spans after ESD. In this section, we explore three methods to incrementally learn class-specific knowledge, namely, add-based method, NLI-based method and prompt-based method.

#### 3.4.1 Add-based method

Inspired by AddNER[10], we classify the extracted spans into different classes in the ECD phase by adding new classifiers when novel classes are added. Each classifier corresponds to one category. We adopt [$\text{E}_1$] and [$\text{E}_2$] to mark the location of the entity span. Specifically, the input of the ECD module becomes

$$X_{\text{in}} = \{[\text{CLS}], x_1, x_2, \dots, x_{i-1}, [\text{E}_1], s_{ij}, [\text{E}_2], x_{j+1}, x_{j+2}, \dots, x_n, [\text{SEP}]\} \quad (5)$$

where $s_{ij}$ denotes the extracted span that needs to be classified. As an example, in Fig. 3a, we convert the input "Reuters reported in New York..." into "[CLS] [$\text{E}_1$] Reuters [$\text{E}_2$] reported in New York..." and use another BERT encoder to obtain the token representations $\boldsymbol{X}$ for each token. Then, we concatenate the token representations of [$\text{E}_1$] and [$\text{E}_2$] to get the entity representations $\boldsymbol{h}$. Finally, we calculate the probability $p(c|s_{ij})$ of the extracted span being in each entity class $c$ with binary classifiers:

$$p(c_1|s_{ij}), p(c_2|s_{ij}), \dots, p(c_{o+p}|s_{ij}) = \text{softmax}(\boldsymbol{W}_1\boldsymbol{h}, \boldsymbol{W}_2\boldsymbol{h}, \dots, \boldsymbol{W}_{o+p}\boldsymbol{h}),$$
$$\boldsymbol{h} = \text{concat}(\text{BERT}(X_{\text{in}})_{[\text{E}_1]}, \text{BERT}(X_{\text{in}})_{[\text{E}_2]}) \quad (6)$$

where $\boldsymbol{W}_1, \boldsymbol{W}_2, \dots, \boldsymbol{W}_{o+p}$ denote the parameters of the classifier corresponding to each class. We consider the class with the largest probability as the label of entity $s_{ij}$. Similar to the CRF-based method, we apply knowledge distillation to keep knowledge of old classes. The difference is that we only adopt KL divergence on the classifiers of old classes. As for novel class knowledge, we apply cross-entropy loss to update the novel classifiers. We also apply parameter-cloning to initialize parameters. Cloned parameters belong to embeddings, the BERT encoder, and the classifiers for $C_{t-1}$.

#### 3.4.2 NLI-based method

Inspired by EFL[30], we classify the extracted spans into different classes in the ECD phase by adopting descriptions of entity classes and modeling it as a Natural Language Inference (NLI) task: whether the template sentence can be inferred from the row sentence. More specifically, the input of the ECD module becomes

$$X_{\text{in}} = [\text{CLS}] + X + [\text{SEP}] + X_{\text{templ}} \quad (7)$$

where $X_{\text{templ}} = s_{ij}$ means $X_{\text{des}}$[15] denotes the template sentence, and $X_{\text{des}}$ is the entity class description. As an example, in Fig. 3b, the input "Reuters reported in New York..." can be converted into "[CLS] Reuters reported in New York... [SEP] Reuters means an [L] entity." where [L] is filled by all label words for $C_t$. We input $X_{\text{in}}$ into the BERT encoder to obtain the hidden representation $h$ and use it to calculate the probability $p(c|s_{ij})$ of the extracted span being in each entity class $c$:

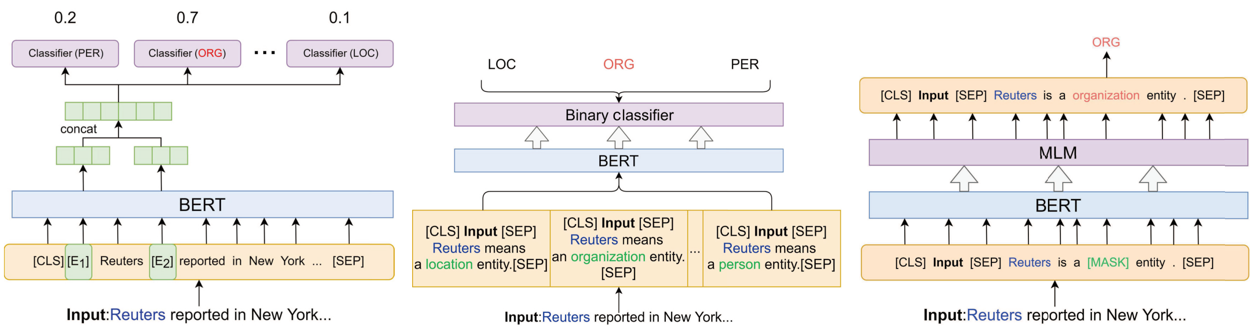
$$p(c|s_{ij}) = \text{softmax}(h),$$
$$h = \boldsymbol{W}(\text{BERT}(X_{\text{in}})_{[\text{CLS}]}) \quad (8)$$

Only when $p(c|s_{ij}) >$ threshold, we judge $s_{ij}$ as an entity owned by class $c$.

The ECD module also applies parameter-cloning to initialize parameters. $M_t^C$ is trained with $C_t$ and has the same layers as $M_{t-1}^C$, of which parameters are initialized from $M_{t-1}^C$.

Similar to the word-pair-based method, we adopt the label-fusion method to avoid catastrophic forgetting. Specifically, we first obtain $\boldsymbol{Y}'(S)$ as shown in Section 3.3 and build a part of inputs $X'_{\text{in}}(C)$ for old classes $C_{t-1}$. The other part of inputs $X_{\text{in}}(C)$ for novel classes $C^{\text{new}}$ is converted from the true label $\boldsymbol{Y}(S)$. Mix $X'_{\text{in}}(C)$ and $X_{\text{in}}(C)$ to obtain the final inputs $X''_{\text{in}}(C)$. Then, we obtain the hard label $\boldsymbol{Y}'(C)$ by predicting $X''_{\text{in}}(C)$ with the $t-1$ ECD module $M_{t-1}^C$. Ultimately, we integrate $\boldsymbol{Y}'(C)$ and the true label $\boldsymbol{Y}(C)$ to obtain the gold label $\boldsymbol{Y}''(C)$ for training with the following strategies to keep old knowledge:

- If $s_{ij}$ belongs to $C^{\text{new}}$, the corresponding input will be labeled 1, whereas others will be labeled 0.



(a) Add-based (b) NLI-based (c) Prompt-based

**Fig. 3 Comparison of different prompt methods. (a) Add-based method, which converts the multi-classification task to many binary classification tasks. (b) NLI-based method, which converts the multi-classification task to a binary classification task. (c) Prompt-based method, which reuses the MLM head and enhances the MLM head by predicting original words.**

• Else if $s_{ij}$ is judged to be one or more classes of $C_{t-1}$, the corresponding inputs will be labeled 1, whereas the others will be labeled 0.

We denote the generated labels as $y''_{ijk}(C) \in \{0,1\}, 1 \leqslant k \leqslant o+p$. Each new sentence has $o+p$ corresponding training samples. We update $M_t^C$ by computing CE, the cross-entropy loss is

$$L^C = \mathrm{CE}(y''_{ijk}(C), p(c|s_{ij})) \tag{9}$$

where $p(c|s_{ij})$ is the output probability distribution of the extracted span belonging to a given entity class.

#### 3.4.3 Prompt-based method

Prompt methods can help few-shot learning[29, 30, 42, 43]; thus, we combine the template sentence with the row sentence to learn class-specific knowledge without adding new parameters. However, using these prompt methods directly requires $o+p$ corresponding judgments for each candidate entity span, causing substantial computational costs. Therefore, we explore the prompt-based method to predict once for each span inspired by TemplateNER[15]. Specifically, the input of the ECD becomes

$$X_{\text{in}} = [\text{CLS}] + X + [\text{SEP}] + X_{\text{temp2}} \tag{10}$$

where $X_{\text{temp2}} = s_{ij}$ is a [MASK] entity, which denotes the template sentence in the prompt-based metric[15]. The goal is to predict which label word is the best replacement in the [MASK] token. In the vocabulary of the pre-trained model, each class will have a corresponding label word. In this way, the Masked Language Model (MLM) head trained during the pre-training phase can be reused to close the gap between fine-tuning and pre-trained. We define $LW_t$ as the set of label words for seen classes at the $t$ step. The input sentence $X_{\text{in}}$ will be sent into the BERT encoder along with the MLM head to produce the hidden representation $\boldsymbol{h}$. Last, the probability $p(c|s_{ij})$ of class $c$ is calculated by applying the softmax function on $h$:

$$p(c|s_{ij}) = \mathrm{softmax}_{LW_t}(\boldsymbol{h}),$$
$$\boldsymbol{h} = \mathrm{MLM}(\mathrm{BERT}(X_{\text{in}})_{[\text{MASK}]}) \tag{11}$$

where $\mathrm{softmax}_{LW_t}$ means probabilities are only calculated on label words in $LW_t$. The category that corresponds to the most likely label word is selected.

Only predicting the [MASK] word has the probability of losing old knowledge as new sentences may not contain old class entities, which may cause overfitting of new classes. To solve the above problem, we propose an improved version of the prompt-based method (see Fig. 3c). When fed with the input sentence "[CLS] Reuters reported in New York... [SEP] Reuters is a [MASK] entity", the ECD module is trained to predict a label word "organization" at the position of the tag "[MASK]" as an indication of the label "ORG". While for other words, the module remains to predict the original words.

We employ knowledge distillation and label-fusion simultaneously to transform old knowledge. When $s_{ij} \in C^{\text{new}}$, we compute the loss of MLM head $L_{\text{MLM}}$ for tokens $\neq$ [MASK] to predict original tokens and $L_{\text{CE}}$ for the [MASK] token. If $s_{ij} \in C_{t-1}$, $L_{\text{CE}}$ is calculated for the [MASK] token with its hard label, while $L_{\text{KD}}$ is utilized for the other tokens to keep knowledge from $M_{t-1}^C$. Finally, we train the ECD module by minimizing the loss:

$$L^C = \gamma_1 L_{\text{CE}} + \gamma_2 L_{\text{KD}} + \gamma_3 L_{\text{MLM}} \tag{12}$$

where $\gamma_1$, $\gamma_2$, and $\gamma_3$ are three hyperparameters to measure the contribution of the three losses.

## 4 Experiment

### 4.1 Experimental setting

#### 4.1.1 Datasets

We perform experiments on three well-known datasets in NER: Conll2003[16], OntoNotes[17], and Few-NERD[14]. Conll2003 includes 22 000 sentences and consists of 4 classes. OntoNotes comprises 104 000 sentences and has 18 classes. To compare with Monaikul et al.[10], we only consider 6 classes on OntoNotes. Few-NERD has 66 classes with 188 000 sentences.

#### 4.1.2 Incremental settings

Following Monaikul et al.[10], we separate the train set and the dev set of Conll2003 into 4 subsets, OntoNotes into 6 subsets, and Few-NERD into 66 subsets using $D_1, D_2, \ldots, D_t$, where $t$ is the number of the subsets. $D_t$ only has annotated labels for one class $C_t - C_{t-1}$. For Conll2003, we randomly select 3 classes as base classes and the remaining 1 as the incremental class, whereas for OntoNotes, we randomly select 4 classes as base classes and the remaining 2 as incremental classes. For Few-NERD, we designated 50 classes as base classes and every 4 classes in the remaining classes as an incremental subtask. The first subtask $T_1$ has enough training data, whereas all subsequent tasks are few-shot. For subtask $T_t, t > 1$, we conduct 5/10/20-shot experiments to verify the effectiveness of our method. Specifically, we use $K$ examples per class to train and other $K$ examples per class to validate, where $K \in \{5, 10, 20\}$. For testing at the $t$ step, we use the test examples only annotated for $C_t$.

We consider two different incremental cases: (1) adding at once, where all novel classes will be added in step $t = 2$ once; (2) adding one by one, where all novel classes divided into subtasks are introduced one at a time to test our approach's ability to learn one entity class at a time.

#### 4.1.3 Baselines and variants

We consider the following three groups of baselines: (1) Few-shot NER. To test our decoupled two-phase NER method, we adopt some few-shot NER models as baselines shown as follows: Proto[26], NNShot[13], and StructShot[13]. To apply these few-shot models to our incremental NER situation, we first represent old class prototypes as average embeddings of pseudo-labels over a few instances. The distances between the token representation and each class prototype or training token representation are then compared. For NNShot, we consider two labeling methods (Inside-Outside (IO) and Begin-Inside-Outside (BIO)). (2) Class-Incremental NER. We also experiment with two class-incremental NER baselines showing high performance with training on large-annotated data. We implement ExtendNER and AddNER[10]. (3) Class-Incremental Few-shot NER. We compare our proposal with the state-of-the-art work of CIFNER, namely, CIF NER[35].

We also consider the following combinations for our DTPF: (1) DTPF (WP+Add), which adopts word-pair-based in ESD and add-based method in ECD; (2) DTPF (WP+NLI), which utilizes the NLI-based method to classify; (3) DTPF (WP+P), which uses the prompt-based method instead of the NLI-based method; (4) DTPF (CRF+Add), replaces the word-pair-based method with the CRF-based method compared with the DTPF (WP+Add); (5) DTPF (CRF+NLI), which adds NLI-based method after CRF-based; (6) DTPF (CRF+P), which uses the prompt-based method instead of the NLI-based method compared with the DTPF (CRF+NLI).

#### 4.1.4 Hyperparameters and evaluation metrics

We train the model with a batch size of 32, a maximum sentence length of 200/300/400 tokens for Conll2003/OntoNotes/Few-NERD, and a learning rate of $5\times10^{-5}$ for 20 epochs with early stopping (patience= 3). A batch size of 2 was used for model training in incremental subtasks, and patience increased to 5. We train and test our model on a V100 GPU with 32 GB of memory for all experiments. We train the ESD and ECD modules independently.

For the CRF-based method, we select $\alpha$ from 0.1, 0.25, 0.5, and 0.75 on Conll2003 and choose the best $\alpha = 0.1$ on Conll2003. Then we set $\alpha = 0.1$ on OntoNotes and Few-NERD. For the prompt-base method, we set $\gamma_1 = \gamma_2 = \gamma_3 = 1$ in all experiments.

At time step $t$, we evaluate the model performance through F1 scores on the test set of all seen classes $T_i, i \in [1, t]$. We report our experimental results for 8/6 sequences of class permutations on Conll2003/OntoNotes as shown in Table 1. Additionally, we randomly choose ten permutations for Few-NERD. We report the average F1 scores over different permutations.

### 4.2 Results on Conll2003 and OntoNotes

#### 4.2.1 Adding at once

Table 2 illustrates the performance of our proposed method and baselines on Conll2003 and OntoNotes when adding new classes all at once. We present the average F1 scores for all classes as well as the incremental classes after the addition of new classes. Our observations are as follows. (1) In the class-incremental few-shot setting, our decoupled two-phase method outperforms all baselines for novel and base classes simultaneously, demonstrating its superiority. Especially, DTPF outperforms AddNER in the 5-shot setting by 10.26% and 16.76% on Conll2003 and OntoNotes, respectively. With fewer shots, the relative gain of DTPF becomes more significant, demonstrating our method's ability to handle fewer-shot tasks. (2) When compared to other methods, few-shot methods (Proto/NNShot/StructShot) produce the worst results. The lack of examples for previous classes leads to

**Table 1 Entity class permutations for each step.**

| Permutations with CoNLL2003 |
|---|
| $P_1$ : PER → LOC → ORG → MISC |
| $P_2$ : PER → MISC → LOC → ORG |
| $P_3$ : LOC → PER → ORG → MISC |
| $P_4$ : LOC → ORG → MISC → PER |
| $P_5$ : ORG → LOC → MISC → PER |
| $P_6$ : ORG → MISC → PER → LOC |
| $P_7$ : MISC → PER → LOC → ORG |
| $P_8$ : MISC → ORG → PER → LOC |
| **Permutations with Ontonotes** |
| $P_1$ : ORG → PER → GPE → DATE → CARD → NORP |
| $P_2$ : DATE → NORP → PER → CARD → ORG → GPE |
| $P_3$ : GPE → CARD → ORG → NORP → DATE → PER |
| $P_4$ : NORP → ORG → DATE → PER → GPE → CARD |
| $P_5$ : CARD → GPE → NORP → ORG → PER → DATE |
| $P_6$ : PER → DATE → CARD → GPE → NORP → ORG |

**Table 2 Results on the Conll2003 dataset and OntoNotes dataset for CIFNER in the setting of adding at once.**

| Model | Conll2003 | | | | | | Ontonotes | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 5-shot | | 10-shot | | 20-shot | | 5-shot | | 10-shot | | 20-shot | |
| | Novel | All | Novel | All | Novel | All | Novel | All | Novel | All | Novel | All |
| Proto | 8.04 | 6.85 | 35.42 | 31.73 | 45.42 | 42.93 | 33.03 | 24.42 | 25.34 | 19.31 | 37.84 | 24.99 |
| NNShot (IO) | 48.88 | 44.59 | 58.54 | 62.30 | 58.30 | 61.92 | 35.26 | 42.93 | 43.72 | 53.31 | 46.38 | 53.57 |
| NNShot (BIO) | 27.18 | 39.12 | 39.63 | 55.89 | 47.33 | 59.52 | 34.12 | 46.48 | 36.15 | 48.17 | 40.30 | 49.7 |
| StructShot | 27.43 | 39.15 | 36.26 | 54.68 | 46.08 | 58.59 | 32.85 | 40.11 | 38.75 | 48.02 | 40.99 | 49.84 |
| AddNER | 53.21 | 66.59 | 61.71 | 73.61 | 70.64 | 76.61 | 26.40 | 55.45 | 54.65 | 72.61 | 62.01 | 73.95 |
| ExtendNER | 37.72 | 41.67 | 52.99 | 66.81 | 64.42 | 71.82 | 37.89 | 56.49 | 53.09 | 65.85 | 48.19 | 67.52 |
| DTPF (CRF+Add) | 52.54 | 68.16 | 63.49 | 71.66 | 71.54 | 74.44 | 35.83 | 58.52 | 54.02 | 60.40 | 63.65 | 73.73 |
| DTPF (CRF+P) | 54.93 | 76.32 | 65.25 | **77.09** | 67.62 | 76.18 | 44.72 | 69.24 | 56.83 | 69.84 | 64.86 | 74.61 |
| DTPF (WP+Add) | 54.55 | 64.44 | 60.51 | 72.69 | 70.81 | 76.31 | 38.14 | 60.35 | 56.20 | 73.48 | 63.61 | 74.69 |
| DTPF (WP+NLI) | 53.83 | 69.56 | 56.56 | 71.25 | 69.26 | 77.89 | 45.30 | **74.82** | 57.28 | 74.17 | 64.71 | 76.98 |
| DTPF (WP+P) | **59.49** | **76.85** | **66.09** | 74.79 | **73.58** | **82.29** | **47.81** | 72.21 | **57.73** | **74.86** | **66.3** | **77.89** |

distorted feature distributions of new data. (3) In the CIFNER scenario, the methods (AddNER/ExtendNER) also suffer from catastrophic forgetting. However, AddNER which provides a classifier for each class, which reaches the best performance except ours, showing that a good parameter initialization is beneficial to CIFNER.

Based on our variants, we discover that DTPF (WP+P) is the superior combination for our proposed framework, particularly in fewer-shot settings. Furthermore, the small difference of 2.61% between DTPF (WP+P) and DTPF (CRF+P) explains why the first phase of ESD is more focused on learning class-generic knowledge. In light of the inconsistent performance (76.48%, 74.11%, and 70.33%) among DTPF (WP+P), DTPF (WP+NLI), and DTPF (WP+Add), the second phase ECD is more focused on learning class-specific knowledge.

### 4.2.2 Adding one by one

To verify the effectiveness of our method in the setting of adding novel classes in turn, we report the average F1 scores for all classes as new classes emerge on Conll2003 and OntoNotes. Specifically, the first subtask $T_1$ has enough training data with one class, whereas all subsequent subtasks are few-shot, and each subtask has only one class. We only consider the word-pair-based method in the ESD phase because the CRF-based method is worse than the word-pair-based method in terms of performance (see Table 2). The results are exhibited in Tables 3 and 4. From the results, we can see that our method has better performance in the condition of adding novel classes in turn. For instance, our proposed method outperforms all baseline

**Table 3 Results of Conll2003 in the setting of adding one by one.**

| Model | 5-shot | | | | | 10-shot | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Step1 | Step2 | **Step3** | Step4 | $Avg \geqslant 2$ | Step1 | Step2 | Step3 | Step4 | $Avg \geqslant 2$ |
| ExtendNER | 88.28 | 25.70 | 22.74 | 20.05 | 22.83 | 88.28 | 51.52 | 38.47 | 33.44 | 41.14 |
| AddNER | 88.59 | 57.53 | 37.18 | 35.51 | 43.41 | 88.59 | 59.72 | 53.51 | 49.30 | 54.18 |
| CIF NER† | 88.35 | **71.31** | **63.76** | 59.37 | **64.18** | 88.35 | **70.75** | 64.60 | 60.02 | 65.12 |
| DTPF (WP+Add) | 88.24 | 60.37 | 45.37 | 40.69 | 48.81 | 88.24 | 62.68 | 57.65 | 50.11 | 56.81 |
| DTPF (WP+NLI) | 87.58 | 61.35 | 55.65 | 53.16 | 56.72 | 87.75 | 65.86 | 60.69 | 58.39 | 61.65 |
| DTPF (WP+P) | 87.75 | 63.73 | 60.04 | **60.30** | 61.36 | 87.75 | 68.27 | **65.55** | **64.55** | **66.12** |

Note: † denotes the results reported in **CIF NER**[35]. The best results are in **bold**.

**Table 4 Results of OntoNotes in the setting of adding one by one.**

| Model | 5-shot | | | | | | 10-shot | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Step1 | Step2 | Step3 | Step4 | Step5 | Step6 | Step2 | Step3 | Step4 | Step5 | Step6 |
| ExtendNER | 82.41 | 22.68 | 11.21 | 13.85 | 9.87 | 8.70 | 35.60 | 28.53 | 30.51 | 23.49 | 22.18 |
| AddNER | 83.32 | 46.62 | 30.45 | 18.95 | 16.54 | 7.73 | 51.36 | 41.96 | 37.56 | 35.45 | 34.59 |
| DTPF (WP+Add) | 82.09 | 53.55 | 37.45 | 25.70 | 18.52 | 9.27 | 58.89 | 47.14 | 41.63 | 40.61 | 36.98 |
| DTPF (WP+NLI) | 81.68 | 53.42 | 41.83 | 39.75 | 38.59 | 34.35 | 60.46 | 52.97 | 45.86 | 43.37 | 39.08 |
| DTPF (WP+P) | 82.69 | **57.92** | **44.56** | **43.57** | **40.45** | **38.08** | **61.75** | **54.02** | **48.49** | **45.64** | **40.35** |

models in both 5-shot and 10-shot settings. Specifically, DTPF surpasses the current state-of-the-art method CIF NER on COnll2003 by 1.00% in the 10-shot setting, demonstrating the superiority of our method on the class-incremental few-shot NER task. Furthermore, when learning a sequence of incremental subtasks, our method loses less performance than baselines, indicating that our method has a more stable capability in incremental steps.

### 4.3 Results on Few-NERD

We run experiments on Few-NERD to validate the effectiveness of our framework on larger datasets with more classes. Table 5 displays the 5/10/20-shot results on Few-NERD when adding at once. We can see that DTPF outperforms CINER methods by 12.93%/5.53%/6.66%, demonstrating our proposed method's strong generalization ability of our proposed method even on datasets with many classes. Furthermore, we use 50 base classes and four classes in each incremental step. Table 6 shows that our method outperforms other baselines by more than 25.51% on Few-NERD in the 5 shot setting, indicating that our decoupled two-phase method can achieve better results even with fewer examples.

### 4.4 Effect of error propagation

The problem of error propagation is a well-known disadvantage of pipeline training. In our final model, we use predicted entity spans to determine entity type, and the NER task must identify both the entity span and its category simultaneously. Thus, if the entity span is not correctly judged in the first phase of ESD, the overall F1 score will suffer regardless of whether the ECD module can identify entity classes. The effect of error propagation in our model is discussed further below.

We compare the performance of our method and two end-to-end models (AddNER and ExtendNER) on the entity span extraction task in the incremental 10-shot setting. Specifically, we apply the same incremental sequences to Conll2003 (see Table 1) and compute the F1 scores that only consider entity spans and not the entity classes. As shown in Table 7, our method DTPF (WP) only decreases by 5.49% after four steps, whereas DTPF (CRF) decreases by 6.24%. However, ExtendNER decreases by 13.79%, and AddNER reaches 17.20%. In the first phase of the NER task, the end-to-end methods exhibit a greater performance drop than pipeline methods. In addition, the error rate in ESD is low, which has a negligible effect on ECD's second phase. We hypothesize that it is because contextual representations of entity span and entity category capture different levels of information (class-generic knowledge and class-specific knowledge) in NER. Consequently, sharing their representations via an end-to-end framework may hinder performance. Decoupling the NER task and fusing entity span information at the input layer of ECD enables more accurate contextual representations, resulting in improved results.

To sum up, our two-phase method is less affected by error propagation and is more competitive than the end-to-end models on the CIFNER task.

## 5 Conclusion

In this paper, we primarily investigate CIFNER, which requires the model to incrementally learn new entity classes with few labeled data while retaining previous knowledge. We proposed a DTPF to solve the CIFNER task, which has the obvious advantage of obtaining better parameter initialization as well as learning class-generic and class-specific knowledge independently in the two phases. In comparison to other methods, sufficient experimental results demonstrate the efficacy of our approach in preventing catastrophic forgetting and learning new entity classes. We intend to improve CIFNER's performance and robustness in the future by incorporating external knowledge into the prompt process or by employing adversarial learning to

**Table 5 Results on Few-NERD dataset for CIFNER in the setting of adding at once.**

| Model | 5-shot | | 10-shot | | 20-shot | |
|---|---|---|---|---|---|---|
| | Novel | All | Novel | All | Novel | All |
| AddNER | 32.03 | 42.36 | 33.10 | 43.84 | 38.02 | 43.89 |
| ExtendNER | 24.65 | 32.37 | 27.30 | 35.66 | 32.84 | 40.83 |
| DTPF (WP+P) | **41.95** | **55.29** | **39.48** | **49.37** | **43.22** | **50.55** |

**Table 6 Results on Few-NERD dataset for CIFNER in the setting of adding in turn.**

| Model | 5-shot | | | | | 10-shot | | | | | 20-shot | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Step1 | Step2 | Step3 | Step4 | Step5 | Step1 | Step2 | Step3 | Step4 | Step5 | Step1 | Step2 | Step3 | Step4 | Step5 |
| AddNER | 60.64 | 47.11 | 31.93 | 23.39 | 21.89 | 60.64 | 42.17 | 28.07 | 22.63 | 19.95 | 60.64 | 41.37 | 30.69 | 23.57 | 22.35 |
| ExtendNER | 59.85 | 15.75 | 11.71 | 11.15 | 9.33 | 59.85 | 21.08 | 14.98 | 14.56 | 14.80 | 59.85 | 27.77 | 20.07 | 18.41 | 18.53 |
| DTPF (WP+P) | 60.14 | **52.34** | **49.75** | **48.96** | **47.4** | 60.14 | **52.39** | **48.23** | **45.79** | **42.44** | 60.14 | **50.18** | **43.32** | **39.19** | **43.64** |

**Table 7 Results on Conll2003 at the first phase ESD in the 10-shot learning setting.**

| Model | Step1 | Step2 | Step3 | Step4 | $Avg \geqslant 2$ |
|---|---|---|---|---|---|
| ExtendNER | 88.05 | 57.86 | 67.05 | 74.26 | 66.39 |
| AddNER | 89.13 | 62.75 | 65.14 | 71.93 | 66.61 |
| DTPF (CRF) | 89.13 | 73.31 | 76.81 | 82.89 | 77.68 |
| DTPF (WP) | 88.69 | **73.60** | **77.11** | **83.20** | **77.98** |

mine class-generic and class-specific knowledge more deeply.

## Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 62006243).

### References

[1] J. P. C. Chiu and E. Nichols, Named entity recognition with bidirectional LSTM-CNNs, *Trans. Assoc. Comput. Linguist.*, vol. 4, pp. 357–370, 2016.

[2] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in *Proc. 2019 Conf. North American Chapter of the Association for Computational Linguistics*, Minneapolis, MN, USA, 2019, pp. 4171–4186.

[3] X. Li, H. Yan, X. Qiu, and X. Huang, FLAT: Chinese NER using flat-lattice transformer, in *Proc. 58th Ann. Meeting of the Association for Computational Linguistics*, Virtual, 2020, pp. 6836–6842.

[4] X. Ma and E. Hovy, End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF, in *Proc. 54th Ann. Meeting of the Association for Computational Linguistics*, Berlin, Germany, 2016, pp. 1064–1074.

[5] H. Yan, B. Deng, X. Li, and X. Qiu, TENER: Adapting transformer encoder for named entity recognition, arXiv preprint arXiv: 1911.04474, 2019.

[6] S. Zhao, M. Hu, Z. Cai, H. Chen, and F. Liu, Dynamic modeling cross-and self-lattice attention network for Chinese NER, in *Proc. 35th AAAI Conf. on Artificial Intelligence, 33rd Conf. on Innovative Applications of Artificial Intelligence, The 11th Symp. on Educational Advances in Artificial Intelligence*, Virtual, 2021, pp. 14515–14523.

[7] R. M. French, Catastrophic forgetting in connectionist networks, *Trends Cognit. Sci.*, vol. 3, no. 4, 1999, pp. 128–135.

[8] M. McCloskey and N. J. Cohen, Catastrophic interference in connectionist networks: The sequential learning problem, *Psychol. Learn. Motiv.– Adv. Res. Theory*, vol. 24, pp. 109–165, 1989.

[9] G. Castellucci, S. Filice, D. Croce, and R. Basili, Learning to solve NLP tasks in an incremental number of languages, in *Proc. 59th Ann. Meeting of the Association for Computational Linguistics and the 11th Int. Joint Conf. on Natural Language Processing*, Virtual, 2021, pp. 837–847.

[10] N. Monaikul, G. Castellucci, S. Filice, and O. Rokhlenko, Continual learning for named entity recognition, in *Proc. 35th AAAI Conf. on Artificial Intelligence, 33rd Conf. on Innovative Applications of Artificial Intelligence, The 11th Symp. on Educational Advances in Artificial Intelligence*, 2021, pp. 13570–13577.

[11] A. Fritzler, V. Logacheva, and M. Kretov, Few-shot classification in named entity recognition task, in *Proc. 34th ACM/SIGAPP Symp. on Applied Computing (SAC'19)*, Limassol, Cyprus, 2019, pp. 993–1000.

[12] P. Wang, R. Xu, T. Liu, Q. Zhou, Y. Cao, B. Chang, and Z. Sui, An enhanced span-based decomposition method for few-shot sequence labeling, in *Proc. 2022 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Seattle, WA, USA, 2022, pp. 5012–5024.

[13] Y. Yang and A. Katiyar, Simple and effective few-shot named entity recognition with structured nearest neighbor learning, in *Proc. 2020 Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, Virtual, 2020, pp. 6365–6375.

[14] N. Ding, G. Xu, Y. Chen, X. Wang, X. Han, P. Xie, H. Zheng, and Z. Liu, Few-NERD: A few-shot named entity recognition dataset, in *Proc. 59th Ann. Meeting of the Association for Computational Linguistics and the 11th Int. Joint Conf. on Natural Language Processing*, Virtual, 2021, pp. 3198–3213.

[15] L. Cui, Y. Wu, J. Liu, S. Yang, and Y. Zhang, Template-based named entity recognition using BART, in *Proc. 59th Findings of the Association for Computational Linguistics*, 2021, pp. 1835–1845.

[16] E. F. T. K. Sang and F. De Meulder, Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition, in *Proc. 7th Conf. on Natural Language Learning at HLT-NAACL 2003*, Edmonton, Canada, 2003, pp. 142–147.

[17] E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel, OntoNotes: The 90% solution, in *Proc. 2006 Human Language Technology Conf. of the NAACL*, New York City, NY, USA, 2006, pp. 57–60.

[18] Z. Chen and B. Liu, Lifelong machine learning, second edition. Cham, Germany: Springer, 2016.

[19] Z. Ke, B. Liu, H. Xu, and L. Shu, CLASSIC: Continual and contrastive learning of aspect sentiment classification tasks, in *Proc. 2021 Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, Virtual, 2021, pp. 6871–6883.

[20] F. K. Sun, C. H. Ho, and H. Y. Lee, LAMOL: Language modeling for lifelong language learning, arXiv preprint arXiv: 1909.03329, 2019.

[21] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A Rusu, A. Pritzel, and D. Wierstra, Pathnet: Evolution channels gradient descent in super neural networks, arXiv preprint arXiv: 1701.08734, 2017.

[22] Z. Li and D. Hoiem, Learning without forgetting, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, 2018.

[23] S. A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, iCaRL: Incremental classifier and representation learning, in *Proc. 2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 2001–2010.

[24] T. Gao, X. Han, Z. Liu, and M. Sun, Hybrid attention-based prototypical networks for noisy few-shot relation classification, in *Proc. 33$^{rd}$ AAAI Conf. on Artificial Intelligence*, Honolulu, HI, USA, 2019, pp. 6407–6414.

[25] R. Geng, B. Li, Y. Li, J. Sun, and X. Zhu, Dynamic memory induction networks for few-shot text classification, in *Proc. 58$^{th}$ Ann. Meeting of the Association for Computational Linguistics*, Virtual, 2020, pp. 1087–1094.

[26] J. Snell, K. Swersky, and R. Zemel, Prototypical networks for few-shot learning, in *Proc. 31$^{st}$ Int. Conf. on Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 4080–4090.

[27] Y. Hou, W. Che, Y. Lai, Z. Zhou, Y. Liu, H. Liu, and T. Liu, Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network, in *Proc. 58$^{th}$ Ann. Meeting of the Association for Computational Linguistics*, 2020, pp. 1381–1393.

[28] D. Yu, L. He, Y. Zhang, X. Du, P. Pasupat, and Q. Li, Few-shot intent classification and slot filling with retrieved examples, in *Proc. 2021 Conf. of the North American Chapter of the Association for Computational Linguistics*, 2021, pp. 734–749.

[29] T. Schick and H. Schütze, Exploiting cloze-questions for few-shot text classification and natural language inference, in *Proc. 16$^{th}$ Conf. of the European Chapter of the Association for Computational Linguistics*, Virtual, 2021, pp. 255–269.

[30] S. Wang, H. Fang, M. Khabsa, H. Mao, and H. Ma, Entailment as few-shot learner, arXiv preprint arXiv: 2104.14690, 2021.

[31] R. Ma, X. Zhou, T. Gui, Y. Tan, L. Yi, Q. Zhang, and X. Huang, Template-free prompt tuning for few-shot NER, in *Proc. 2022 Conf. of the North American Chapter of the Association for Computational Linguistics*, Seattle, WA, USA, 2022, pp. 5721–5732.

[32] S. Dong, X. Hong, X. Tao, X. Chang, X. Wei, and Y. Gong, Few-shot class-incremental learning via relation knowledge distillation, in *Proc. 35$^{th}$ AAAI Conf. on Artificial Intelligence*, 2019, pp. 1255–1263.

[33] J. Pérez-Rúa, X. Zhu, T. M. Hospedales, and T. Xiang, Incremental few-shot object detection, in *Proc. 2020 IEEE/CVF Conf. on Computer Vision and Pattern Recognition* (*CVPR*), Seattle, WA, USA, 2020, pp. 13843–13852.

[34] X. Tao, X. Hong, X. Chang, S. Dong, X. Wei, and Y. Gong, Few-shot class-incremental learning, in *Proc. 2020 IEEE/CVF Conf. on Computer Vision and Pattern Recognition* (*CVPR*), Seattle, WA, USA, 2020, pp. 12180–12189.

[35] R. Wang, T. Yu, H. Zhao, S. Kim, S. Mitra, R. Zhang, and R. Henao, Few-shot class-incremental learning for named entity recognition, in *Proc. 60$^{th}$ Ann. Meeting of the Association for Computational Linguistics*, Dublin, Ireland, 2022, pp. 571–582.

[36] J. Li, H. Fei, J. Liu, S. Wu, M. Zhang, C. Teng, Ji D, and F. Li, Unified named entity recognition as word-word relation classification, in *Proc. 36$^{th}$ AAAI Conf. on Artificial Intelligence*, 2022, pp. 10965–10973.

[37] Y. Chang, L. Kong, K. Jia, and Q. Meng, Chinese named entity recognition method based on BERT, in *Proc. 2021 IEEE Int. Conf. on Data Science and Computer Application* (*ICDSCA*), Dalian, China, 2021, pp. 294–299.

[38] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, Conditional random fields: Probabilistic models for segmenting and labeling sequence data, in *Proc. 18$^{th}$ Int. Conf. on Machine Learning* (*ICML*), Virtual, 2001, pp. 282–289.

[39] J. Su, GlobalPointer: Handle NER in a unified way, (in Chinese), https://spaces.ac.cn/archives/8373, 2021.

[40] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, RoFormer: Enhanced transformer with rotary position embedding, arXiv preprint arXiv: 2104.09864, 2021.

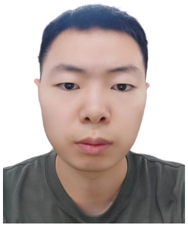
[41] Z. Z. Li, D. W. Feng, D. S. Li, and X. C. Lu, Learning to select pseudo labels: A semi-supervised method for named entity recognition, *Front. Inf. Technol. Electron. Eng.*, vol. 21, no. 6, pp. 903–916, 2020.

[42] T. Schick and H. Schütze, It's not just size that matters: Small language models are also few-shot learners, in *Proc. 2021 Conf. of the North American Chapter of the Association for Computational Linguistics*, Virtual, 2021, pp. 2339–2352.

[43] T. Le Scao and A. Rush, How many data points is a prompt worth? in *Proc. 2021 Conf. of the North American Chapter of the Association for Computational Linguistics*, Virtual, 2021, pp. 2627–2636.


**Zhen Huang** received the BS and PhD degrees from the National University of Defense Technology (NUDT), Changsha, China, in 2006 and 2012, respectively. He was a visiting student with Eurecom, France, in 2009. From 2012 to 2016, he was an assistant professor in the Science and Technology on Parallel and Distributed Laboratory, NUDT, where he is currently an associate professor. He has authored over 70 publications and is an academic editor of the *International Journal of Intelligent Systems*. His research interests include natural language processing, data mining, and knowledge graph.

**Minghao Hu** received the BS and PhD degrees from the National University of Defense Technology (NUDT), Changsha, China, in 2015 and 2019, respectively. He is currently an assistant researcher in the Information Research Center of Military Science, PLA Academy of Military Science. His research interests lie in natural language processing, question answering, and knowledge graph, and he has published more than 20 papers. His doctoral thesis received the Excellence Doctoral Thesis Award from the Chinese Information Processing Society of China.



**Yifan Chen** obtained the BS degree from the National University of Defense Technology, Changsha, China, in 2020. He is currently pursuing the MS degree at the National University of Defense Technology, Changsha, China. His research interests include natural language processing and information extraction.

**Dongsheng Li** obtained the PhD degree in computer science and technology from the National University of Defense Technology, Changsha, China, in 2005. He was awarded the Chinese National Excellent Doctoral Dissertation in 2008. He is a professor and doctoral supervisor at the College of Computer, National University of Defense Technology, Changsha, China. His research interests include distributed systems, cloud computing, and big data processing.

**Changjian Wang** obtained the PhD degree in computer science from National University of Defense Technology in 2015. He is currently an associate professor at the National University of Defense Technology (NUDT), Changsha, China. His current research interests include medical image analysis, natural language processing, and big data.

**Feng Liu** acquired the BS and PhD degrees from the National University of Defense Technology, China, in 1999 and 2006, respectively. He is currently a professor of engineering at the National University of Defense Technology. His research interests include distributed computing technology and artificial intelligence.

**Xicheng Lu** obtained the BS degree in computer science from Harbin Military Engineering Institute, China, in 1970. He was a visiting scholar at the University of Massachusetts, MA, USA from 1982 to 1984. He is now a professor at the College of Computer, National University of Defense Technology (NUDT), China. He has served as a member of the editorial boards of several journals and cochaired many professional conferences. He has been an academician of the Chinese Academy of Engineering since 1999. His research interests include distributed computing, computer networks, parallel computing, and so on.