LightNER: A Lightweight Tuning Paradigm for Low-resource NER via Pluggable Prompting

Xiang Chen[1,2*], **Lei Li**[1,2*], **Shumin Deng**[1,2], **Chuanqi Tan**[3], **Changliang Xu**[4]
Fei Huang[3], **Luo Si**[3], **Huajun Chen**[1,2], **Ningyu Zhang**[1,2,†]
[1]Zhejiang University & AZFT Joint Lab for Knowledge Engine, China
[2]Hangzhou Innovation Center, Zhejiang University, [3]Alibaba Group
[4]State Key Laboratory of Media Convergence Production Technology and Systems
{xiang_chen, leili21, 231sm, huajunsir, zhangningyu}@zju.edu.cn,
{chuanqi.tcq, f.huang, luo.si}@alibaba-inc.com, xu@shuwen.com

Abstract

Most NER methods rely on extensive labeled data for model training, which struggles in the low-resource scenarios with limited training data. Existing dominant approaches usually suffer from the challenge that the target domain has different label sets compared with a resource-rich source domain, which can be concluded as class transfer and domain transfer. In this paper, we propose a lightweight tuning paradigm for low-resource NER via pluggable prompting (LightNER). Specifically, we construct the unified learnable verbalizer of entity categories to generate the entity span sequence and entity categories without any label-specific classifiers, thus addressing the class transfer issue. We further propose a pluggable guidance module by incorporating learnable parameters into the self-attention layer as guidance, which can re-modulate the attention and adapt pre-trained weights. Note that we only tune those inserted module with the whole parameter of the pre-trained language model fixed, thus, making our approach lightweight and flexible for low-resource scenarios and can better transfer knowledge across domains. Experimental results show that LightNER can obtain comparable performance in the standard supervised setting and outperform strong baselines in low-resource settings[1].

1 Introduction

Named Entity Recognition (NER) has been a fundamental task of research within the Natural Language Processing (NLP) community. Mostly, the NER task is formulated as a sequence classification task, aiming to assign the labels to each entity in the input sequence. And those entity labels are all based on pre-defined categories, such as location, organization, person. The current

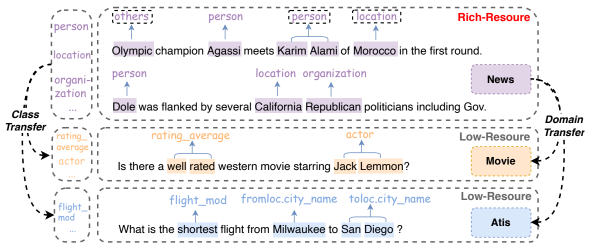

person
location
organi-zation
...
Class Transfer
rating_average
actor
...
flight_mod
...
others
person
person
location
Rich-Resoure
Olympic champion Agassi meets Karim Alami of Morocco in the first round.
person
location
organization
News
Dole was flanked by several California Republican politicians including Gov.
Domain Transfer
rating_average
actor
Low-Resoure
Movie
Is there a well rated western movie starring Jack Lemmon?
flight_mod
fromloc.city_name
toloc.city_name
Low-Resoure
Atis
What is the shortest flight from Milwaukee to San Diego ?

Figure 1: Examples of NER involved in **Class Transfer** and **Domain Transfer** in low-resource setting.

mature methodologies for handling NER is using Pre-trained language models (PLMs) (Devlin et al., 2019) equipped with several NER paradigms to perform extensive training process on large corpus, such as label-specific classifier paradigm (LC) (Strubell et al., 2017; Cui and Zhang, 2019), machine reading comprehension paradigm (MRC) (Yu et al., 2020) and unified generative paradigm (BartNER (Yan et al., 2021)). Unfortunately the resulting models are highly associated with seen categories, which often explicitly memorizing entity values (Agarwal et al., 2021), partially because the output layers require a consistent label set between training and testing. Note that these models require to build a new model from scratch to adapt to a target domain with new entity categories, thus, achieving unsatisfactory performance when the target labeled data is limited.

Unfortunately, this problem is prevalent in real-world application scenarios and draws attention to a challenging but practical research problem: low-resource NER, where the model is built to quickly identify new entities in a completely unseen target domain with only a few supporting samples in the new domain. Overall, low-resource NER (Wiseman and Stratos, 2019; Yang and Katiyar, 2020; Ziyadi et al., 2020) mainly faces two issues as shown in Figure 1: (1) **Class Transfer**. Entity categories can be different across rich-resource settings and low-resource settings. For example, source

[*] Equal Contribution.
[†] Corresponding Author.
[1]Code is in `https://github.com/zjunlp/DeepKE/tree/main/example/ner/few-shot`.

news domain contains the entity categories with "person" ,"location" , etc, while target movie domain adds new categories with "rating_average" and "actor". In this situation, the current mainstream method such as LC, MRC and BARTNER have to refactor a new model and train it from scratch, which is expensive, and unrealistic for real-world settings. (2) **Domain Transfer**. Compared with rich-resource settings, the low-resource setting may have a different textual domain. Intuitively, the sentence in news domain and atis domain contain the different grammar style and allegorical theme, which is not trivial to transfer the model fully trained in the source domain to target domain with few examples.

To address the issue of class transfer, we first reformulate the NER task from sequence labeling to a generative framework with a unified learnable verbalizer to realize **class transfer**. Considering different categories involve varying numbers of words as their descriptions, vanilla mainstream method that assign single classifier for entity may lose important label semantic information. Thus, we propose to construct a unified learnable verbalizer based on generative framework. Different from BartNER (Yan et al., 2021)) that has extra MLP layer in Encoder and classifier in Decoder, our method only contain the original architecture of pre-trained generative model by constructing constructing a unified learnable verbalizer for entity. Therefore, our approach can directly leverage any new or complicated entity types without modifying the network structure.

Recently, prompt-tuning (Schick and Schütze, 2021; Gao et al., 2021; Li and Liang, 2021; Liu et al., 2021c) has emerged to become surprisingly effective for the model adaptation of PLMs, especially in the low-resource setting. However, the prompt-tuning relies on reformulating the paradigm of downstream tasks into new tasks similar to MLM pre-training, which is not efficient for sequence labeling tasks such as NER. Inspired by the success of prompt-learning (Lester et al., 2021) in domain adaptation, we propose a lightweight tuning paradigm with pluggable guidance for NER (LightNER) to tackle the downsides of **domain transfer**. Specifically, we propose to incorporate learnable parameters into the self-attention layer in LMs and regard the parameters as knowledgeable guidance. In particular, we explore lightweight tuning with the pluggable guidance module to urge it to learn domain transfer ability and condition it at inference time.

In light of the limits of the existing techniques, we are interested in building a lightweight tuning framework fot low-resource NER with pluggable prompting. Notably, the modules in LightNER are extremely coupled and indispensable to each other. It is precisely because we design a generative model equipped with a decoupling space to solve the issue class transfer that the pluggable guidance module can realize domain knowledge transfer with lightweight tuning. In a nutshell, LightNER consists of the following contributions:

- We convert sequence labeling to the generative framework and construct decoupling space without any label-specific layers to solve the issue of class transfer. Therefore, the proposed method does not require to build a new model from scratch to adapt to a target domain with new entity categories.

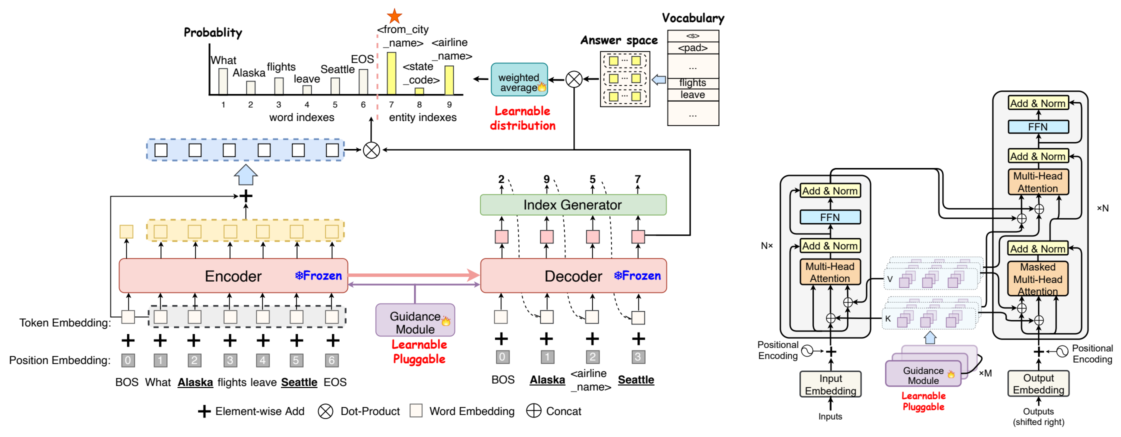
- We propose to incorporate learnable parameters into the self-attention layers as pluggable guidance, which can be seamlessly plugged into the pre-trained generative models to conduct lightweight tuning with cross-domain and cross-task knowledge transfer ability. Therefore, LightNER doesn't need to maintain an LM for each target domain NER tasks and pay for expensive training services.

- We conduct extensive experiments on several benchmark datasets, and by tuning only little parameters, LightNER can achieve comparable results in standard supervised settings and yield promising performance in low-resource settings. Our results also suggest that LightNER has the potential towards cross domain zero-shot generation with pluggable guidance.

2 Related Work

2.1 Named Entity Recognition

The current dominant methods (Ma and Hovy, 2016; Liu et al., 2019; Zhang et al., 2020b; Liu et al., 2021a,b) treat NER as a sequence tagging problem with label-specific classifiers or CRF. Nevertheless, these works still need to modify the model architecture when facing new entity classes; the inability to solve the challenge of class transfer limits its efficiency and transferability, which is not



(a) Overview of LightNER. (b) Attention with guidance.

Figure 2: Overview of our LightNER framework.

suitable for low-resource scenarios. Meanwhile, one crucial research line of low-resource NER is prototype-based methods, which involve meta-learning and have recently become popular few-shot learning approaches in the NER area. Most of the approaches (Fritzler et al., 2019; Wiseman and Stratos, 2019; Yang and Katiyar, 2020; Ziyadi et al., 2020; Henderson and Vulic, 2021; Hou et al., 2020; Lin et al., 2019; Ding et al., 2021) utilize the nearest-neighbor criterion to assign the entity type, which depends on similar patterns of entity between the source domain and the target domain without fully exploiting the potential of PLMs, behaving unsatisfactorily for cross-domain instances.

Recently, Cui et al. (2021) propose template-based BART for few-shot NER, which enumerates all $\hat{n}$-gram possible spans in the sentence and fills them in the hand-crafted templates, classifying each candidate span based on the corresponding template scores. Different from their approach, our framework does not need template engineering and is more friendly with computation complexity.

2.2 Prompt Learning for PLMs

Since the emergence of GPT-3 (Brown et al., 2020), prompt-tuning has received considerable attention. A series of research work (Schick and Schütze, 2021; Schick et al., 2020; Shin et al., 2020; Han et al., 2021; Ben-David et al., 2022; Poth et al., 2021; Ben-David et al., 2022) have emerged, which implies that prompt-tuning can effectively stimulate knowledge from PLMs compared with standard fine-tuning, thus, inducing better performances on few-shot and cross-domain tasks. However, prompt learning mainly focuses on reformulating the downstream tasks' paradigm into completing a cloze task to bridge the gap between pre-training and fine-tuning, lacking an efficient method for NER and other sequence labeling tasks. Different from recent work of prompt tuning for NER (Zhou et al., 2021; Ma et al., 2022), we mainly focus on the issues of domain transfer and class transfer for low-resource NER.

2.3 Lightweight Learning for PLMs

Lightweight fine-tuning is performed to leverage the ability of PLMs with small trainable parameters. On the one hand, several studies consider removing or masking redundant parameters from PLMs (Frankle and Carbin, 2019; Sanh et al., 2020). On the other hand, some researchers (Guo et al., 2021; Zhang et al., 2020a) argue that extra trainable modules should be inserted into PLMs. As a typical approach, adapter-tuning (Houlsby et al., 2019) inserts task-specific layers (adapters) between each layer of PLMs; prefix-tuning (Li and Liang, 2021) prepends a sequence of continuous task-specific vectors to the inputs. However, adapter-tuning adds additional layers into the activation module of LMs, while this modification of the architecture is inconvenient to redeploy the model when switching to a new domain with unseen entity types. Meanwhile, the different label sets among domains make it impossible to transfer prefix-tuning to sequence labeling such as NER.

Apart from Adapter (Houlsby et al., 2019) and prefix-tuning (Li and Liang, 2021) aiming to conduct efficient finetuning, our approach focus on achieving efficient knowledge transfer through a pluggable paradigm.

3 Preliminaries

3.1 Low-resource NER

Given a rich-resource NER dataset $\mathbb{H} = \{(\mathbf{X}_1^H, \mathbf{Y}_1^H), ..., (\mathbf{X}_R^H, \mathbf{Y}_R^H)\}$, where the input is a text sequence of length n, $\mathbf{X}^H = \{x_1^H, \ldots, x_n^H\}$, we use $\mathbf{Y}^H = \{y_1^H, \ldots, y_n^H\}$ to denote corresponding labeling sequence of length n, and adopt $\mathcal{C}^H$ to represent the label set of the rich-resource dataset ($\forall y_i^H, y_j^H \in \mathcal{C}^H$). Traditional NER methods are trained in the standard supervised learning settings, which usually require many pairwise examples, *i.e.,* R is large. However, only a few labeled examples are available for each entity category in real-world applications due to the intensive annotation cost. This issue yields a challenging task of *low-resource* NER, in which given a low-resource NER dataset, $\mathbb{L} = \{(\mathbf{X}_1^L, \mathbf{Y}_1^L), ..., (\mathbf{X}_r^L, \mathbf{Y}_r^L)\}$, the number of labeled data in low-resource NER dataset is quite limited (i.e., $r \ll R$) compared with the rich-resource NER dataset. Regarding the issues of low resource and cross domain, the target entity categories $\mathcal{C}^L$ ($\forall l_i^L, l_j^L \in \mathcal{C}^L$) may be different from $\mathcal{C}^H$, which is challenging for model optimization.

3.2 Label-specific Classifier for NER

Traditional sequence labeling methods usually assign a label-specific classifier over the input sequence, which identifies named entities using BIO tags. A label-specific classifier with parameter $\boldsymbol{\theta} = \{\mathbf{W}_{\mathcal{C}}, \mathbf{b}_{\mathcal{C}}\}$ followed by a softmax layer is used to project the representation $\mathbf{h}$ into the label space. Formally, given $x_{1:n}$, the label-specific classifier method calculates:

$$\begin{aligned} \mathbf{h}_{1:n} &= \textsc{Encoder}(x_{1:n}), \\ q(\boldsymbol{y}|\boldsymbol{x}) &= \textsc{softmax}(\mathbf{h}_i \mathbf{W}_{\mathcal{C}} + \mathbf{b}_{\mathcal{C}})\ (i \in [1, ..., n]), \end{aligned} \tag{1}$$

where $\mathbf{W}_{\mathcal{C}} \in \mathbb{R}^{d \times m}$, $\mathbf{b}_{\mathcal{C}} \in \mathbb{R}^m$ are trainable parameters and m is the numbers of entity categories. We adopt BERT (Devlin et al., 2019) and BART (Lewis et al., 2020) as our Encoder to encoder the representation of text sequence, together with label-specific classifier layer, denoted as **LC-BERT** and **LC-BART** respectively.

4 Methodology

4.1 Task Formulation

Low-resource NER usually involves the class transfer, where new entity categories exist in target domains; however, the traditional sequence labeling method needs a label-specific output layer based on PLMs, hurting its generalization. Therefore, we reformulate the NER as a generative framework to maintain the consistency of architecture and enable the model to handle different entity types. For a given sentence X, we tokenize it into a sequence of tokens $X = \{x_1, x_2, ...x_n\}$. The NER task aims to provide the start and end index of an entity span, along with the entity type, represented by *e*, *t* in our framework, respectively. *e* is the index of tokens and *t* $\in$ {"*person*", "*organization*", ..., } is the set of entity types. Superscript $^{\text{start}}$ and $^{\text{end}}$ denote the start and end index of the corresponding entity token in the sequence. For the generative framework, the target sequence Y consists of multiple base prediction $p_i = \{e_i^{start}, e_i^{end}, t_i\}$ and $Y = \{p_1, p_2, ..., p_l\}$, where l denotes num of entities in X. We take a sequence of tokens X as input and hope to generate the target sequence Y as defined above. The input and output sequence starts and ends with special tokens "<s>" and "</s>". They should also be generated in Y, but we ignore them in equations for simplicity. Given a sequence of tokens X, the conditional probability is calculated as:

$$P(Y|X) = \prod_{t=1}^{3l} p(y_t|X, y_0, y_1, ..., y_{t-1}). \tag{2}$$

4.2 Generative Framework

To conduct **class transfer**, we adopt the seq2seq architecture with the pointer network to model the conditional probability $P(Y|X)$, where the conduction of pointer network is inspired by the See et al. (2017); Yan et al. (2021). Our generative module is shown in Figure 2, consisting of two components:

4.2.1 Encoder

The encoder is to encode X into the hidden representation space as a vector H_{en}.

$$H_{en} = \text{Encoder}(X) \tag{3}$$

where $H_{en} \in \mathcal{R}^{n \times d}$ and d is the hidden state dimension.

4.2.2 Decoder

The decoder part takes the encoder outputs H_{en} and previous decoder outputs $y_1, y_2, ..., y_{t-1}$ as inputs to decode y_t. $y_{i\,i=1}^{t-1}$ indicates the token indexes; an index-to-token converter is applied for conversion.

$$\tilde{y}_i = \begin{cases} X_{y_i}, & \text{if } y_i \text{ is a pointer index} \\ C_{y_i - n}, & \text{if } y_i \text{ is a class index} \end{cases} \tag{4}$$

where $C = [c_1, c_2, ...c_m]$ is the set of entity categories (such as "Person", "Organazation", etc.), which are answer words corresponding to the entity category[2]. After this, we then get the last hidden state for y_t with the converted previous decoder outputs $[\tilde{y}_{i\,i=1}^{t-1}]$.

$$h_t = \text{Decoder}(H_{en}; [\tilde{y}_{i\,i=1}^{t-1}]) \tag{5}$$

where $h_t \in \mathcal{R}^d$; moreover, the probability distribution p_t of token y_t can be computed as follows:

$$\begin{aligned} E_{seq} &= \text{WordEmbed}(X), \\ \tilde{H}_{en} &= \alpha \cdot H_{en} + (1-\alpha) \cdot E_{seq}, \\ p_{seq} &= \tilde{H}_{en} \otimes h_t, \\ p_t &= \text{Softmax}([p_{seq}; p_{tag}]), \end{aligned} \tag{6}$$

where $E_{seq}, \tilde{H}_{en} \in \mathcal{R}^{n \times d}$; $\alpha \in \mathcal{R}$ is a hyperparameter; p_{seq} and p_{tag} refer to the predicted logits on index of entity span and entity categories respectively; $p_t \in \mathcal{R}^{(n+m)}$ is the predicted probability distribution of y_t on all candidate indexes; $[\cdot\,;\,\cdot]$ denotes concatenation in the first dimension. In particular, the details of p_{tag} are in the following subsection.

4.3 Unified Learnable Verbalizer

As for the prediction of entity categories in NER, it is challenging to manually find appropriate tokens in the vocabulary to distinguish different entity types. Besides, some entity type may be complicated or very long in the specific target domain, such as $return\_date.month\_name$ in ATIS (Hakkani-Tür et al., 2016) and $restaurant\_name$ in MIT Restaurant (Liu et al., 2013).

To address the above issues in class transfer, we construct a unified learnable verbalizer containing multiple label words related to each entity class and leverage the weighted average approach for the utilization of the decoupling space $\mathcal{V}$. Concretely, we define a mapping $\mathcal{M}$ from the label space of entity categories $\mathcal{C}$ to the unified learnable verbalizer $\mathcal{V}$, i.e., $\mathcal{M}\colon \mathcal{C} \mapsto \mathcal{V}$. We utilize $\mathcal{V}_c$ to represent the subset of $\mathcal{V}$ that is mapped by a specific entity type c, $\mathcal{V} = \cup_{c \in \mathcal{C}} \mathcal{V}_c$. Take the above $c =$ "return_date.month_name" as example, we set $\mathcal{V}_c = \{$"return","date","month","name"$\}$ according to decomposition of c. Since the direct average function may be biased, we adopt learnable weights β to average the logits of label words in answer space as the prediction logit:

$$\begin{aligned} E_{tag} &= \text{WordEmbed}(\mathcal{M}(\mathcal{C})), \\ p_{tag} &= Concat[\sum_{v \in \mathcal{V}_c} {\beta_v}^c * E_{tag}^c \otimes h_t] \end{aligned} \tag{7}$$

where ${\beta_v}^c$ denotes the weight of entity type c; $\sum_{v \in \mathcal{V}_c} {\beta_v}^c = 1$; $p_{tag} \in \mathcal{R}^m$. Through the construction of the unified learnable verbalizer, LightNER can perceive semantic knowledge in entity categories without modifying the PLM.

4.4 Pluggable Guidance Module

4.4.1 Parameterized Setting

Specifically, LightNER adds two sets of trainable embedding matrices $\{\phi^1, \phi^2, .., \phi^N\}$ for the encoder and decoder, respectively, and sets the number of transformer layers as N, where $\phi_\theta \in \mathbb{R}^{2 \times |P| \times d}$ (parameterized by θ), $|P|$ is the length of the prompt, d represents the $dim(h_t)$, and 2 indicates that ϕ is designed for the key and value. In our method, the LM parameters are fixed, and the prompt parameters θ and the learnable distribution of β are the only trainable parameters.

4.4.2 Pluggable Guidance Layer

LightNER inherits the architecture of the transformer (Vaswani et al., 2017), which is a stack of identical building blocks wrapped up with a feed-forward network, residual connection, and layer normalization. As a specific component, we introduce the pluggable guidance layer over the original query/key/value layer to achieve flexible and effective knowledge transfer. Given an input token sequence $X = \{x_1, x_2, ..., x_n\}$, following the above formulation, we can incorporate the representation of the guidance module into x with the calculation of self-attention. In each layer l, the input sequence representation $\boldsymbol{X}^l \in \mathbb{R}^{nd}$ is first projected into the query/key/value vector:

$$\boldsymbol{Q}^l = \boldsymbol{X}^l \boldsymbol{W}^Q, \boldsymbol{K}^l = \boldsymbol{X}^l \boldsymbol{W}^K, \boldsymbol{V}^l = \boldsymbol{X}^l \boldsymbol{W}^V, \tag{8}$$

where $\boldsymbol{W}_l^Q$, $\boldsymbol{W}_l^K$, $\boldsymbol{W}_l^V \in \mathbb{R}^{d \times d}$. Then, we can redefine the attention operation as:

$$Attention^l = softmax(\frac{\boldsymbol{Q}^l [\boldsymbol{\phi}_k^l; \boldsymbol{K}^l]^T}{\sqrt{d}})[\boldsymbol{\phi}_v^l; \boldsymbol{V}^l]. \tag{9}$$

Based on these representations of inputs and pluggable guidance module, we aggregate them and compute the attention scores to guide the final self-attention flow. Consequently, the guidance module can re-modulate the distribution of attention.

[2]The index of entity categories always starts after the pointer indexes of the given sequence, at $n + 1$.

Traditional Models	P	R	F
Yang et al. (2018)	-	-	90.77
Ma and Hovy (2016)	-	-	91.21
Yamada et al. (2020)	-	-	**94.30**
Gui et al. (2020)	-	-	92.02
Li et al. (2020) †	92.47	93.27	92.87
Yu et al. (2020) ‡	**92.85**	92.15	92.50
LC-BERT	91.93	91.54	91.73
LC-BART	89.60	91.63	90.60
Few-shot Friendly Models	**P**	**R**	**F**
Wiseman and Stratos (2019)	-	-	89.94
Template (Cui et al., 2021)	90.51	93.34	91.90
LightNER	92.39	**93.48**	92.93

Table 1: Model performance on the CoNLL-2003 dataset . "†" indicates that we rerun their code with BERT-LARGE (Devlin et al., 2019). "‡" indicates our reproduction with only the sentence-level context. Although LUKE (Yamada et al., 2020) is pre-trained with a large entity-annotated corpus (Wikipedia), LightNER is highly competitive in rich resource settings even though it is designed for low-resource NER.

5 Experiments

We conduct extensive experiments in standard and low-resource settings. We use CoNLL-2003 (Sang and Meulder, 2003) as the rich-resource domain. Following the settings in Ziyadi et al. (2020) and Huang et al. (2020), we use the Massachusetts Institute of Technology (MIT) Restaurant Review (Liu et al., 2013), MIT Movie Review (Liu et al., 2013), and Airline Travel Information Systems (ATIS) (Hakkani-Tür et al., 2016) datasets as the cross-domain low-resource datasets[3]. Our experiments are evaluated in an exact match scenario, data analysis, and implementation details are presented in the Section Appendix A and B. We also provide the supplementary experimental result for the in-domain low-resource setting as shown in Section Appendix C.1.

5.1 Standard Supervised NER Setting

We adopt the CoNLL-2003 dataset to conduct experiments in the standard supervised settings. A comparison of the results of LightNER and the SOTA methods are listed in Table 1. Mainly, LC-BERT and LC-BART provide a strong baseline. We identify that even though LightNER is designed for the low-resource NER, it is highly competitive with the best-reported score in the rich-resource setting as well, indicating the effectiveness of our decoding strategy and guidance module.

[3]We do not conduct experiemnts on Few-NERD (Ding et al., 2021) since our setting follows (Ziyadi et al., 2020) which is different from the N-way K-shot settting.

5.2 Cross-Domain Low-resource NER Setting

In this section, we evaluate the model performance in the scenarios in which the target entity categories and textual style are specifically different from the source domain, and only limited labeled data are available for training. Precisely, we follow the setting in Cui et al. (2021) to sample a specific number of samples per entity category randomly as the training data in the target domain to simulate the cross-domain low-resource data scenarios. Table 2 lists the results of training models on the CoNLL-2003 dataset as a generic domain and its evaluations on other target domains. The results of LightNER are based on running the experiments five times on random samples and calculating the average of their scores.

Competitive Baselines We consider seven competitive approaches in our experiments. The *prototype-based methods*[4] primarily include the following: (*i*) *Neigh.Tag.* (Wiseman and Stratos, 2019); (*ii*) *Example-based NER* (Ziyadi et al., 2020); (*iii*) *Multi-prototype + NSP* (referred to as *MP-NSP*) is a SOTA prototype-based method reported in (Huang et al., 2020), utilizing noisy supervised pretraining. The *label-specific classifier* mainly include the following: (*iv*) *LC-BERT* and (*v*) *LC-BART* is the adoption of the label-specific classifiers on top of corresponding PLMs. Besides, (*vi*) *Template-based BART* (Cui et al., 2021) recently propose a template-based method for few-shot NER and (*vii*) *BERT-MRC* (Yu et al., 2020) propose to formulate NER as a machine reading comprehension (MRC) task, which is a strong SOTA model for NER. A summary comparison with baselines is shown in Section 4 of Appendix.

Train from Scratch on Target Domain We first consider direct training on the target domain from scratch without any available source domain data. However, prototype-based methods cannot be used in this setting. When compared to the LC-BART, LC-BERT, template-based BART and BERT-MRC, the results of our approach is consistently more persistent, indicating LightNER can better exploit few-shot data. Particularly, LightNER achieve an F1-score of 57.8% in 20-shot setting on MIT Movie, which is higher than the results of LC-BERT and template-based BART in 50-shot setting.

[4]Note that even if the prototype-based methods is training-free in the target domain, they are by no means equivalent to zero-shot setting, since prototype-based methods require labeled data in target domain as supporting examples.

Source	Methods	*MIT Movie*						*MIT Restaurant*						*ATIS*		
		10	20	50	100	200	500	10	20	50	100	200	500	10	20	50
None	LC-BERT	25.2	42.2	49.6	50.7	59.3	74.4	21.8	39.4	52.7	53.5	57.4	61.3	44.1	76.7	90.7
	LC-BART	10.2	27.5	44.2	47.5	54.2	64.1	6.3	8.5	51.3	52.2	56.3	60.2	42.0	72.7	87.5
	Template	37.3	48.5	52.2	56.3	62.0	74.9	46.0	57.1	58.7	60.1	62.8	65.0	71.7	79.4	92.6
	BERT-MRC†	18.7	48.3	55.5	62.5	80.2	82.1	12.3	37.1	53.5	63.9	65.5	70.4	35.3	63.2	90.2
	LightNER	**41.7**	**57.8**	**73.1**	**78.0**	**80.6**	**84.8**	**48.5**	**58.0**	**62.0**	**70.8**	**75.5**	**80.2**	**76.3**	**85.3**	**92.8**
CoNLL03	Neigh.Tag.	0.9	1.4	1.7	2.4	3.0	4.8	4.1	3.6	4.0	4.6	5.5	8.1	2.4	3.4	5.1
	Example.	29.2	29.6	30.4	30.2	30.0	29.6	25.2	26.1	26.8	26.2	25.7	25.1	22.9	16.5	22.2
	MP-NSP	36.4	36.8	38.0	38.2	35.4	38.3	46.1	48.2	49.6	49.6	50.0	50.1	71.2	74.8	76.0
	LC-BERT	28.3	45.2	50.0	52.4	60.7	76.8	27.2	40.9	56.3	57.4	58.6	75.3	53.9	78.5	92.2
	LC-BART	13.6	30.4	47,8	49.1	55.8	66.9	8.8	11.1	42.7	45.3	47.8	58.2	51.3	74.4	89.9
	Template	42.4	54.2	59.6	65.3	69.6	80.3	53.1	60.3	64.1	67.3	72.2	75.7	77.3	88.9	93.5
	BERT-MRC†	20.2	50.8	56.3	62.9	81.5	82.3	15.8	39.5	54.8	65.8	68.8	73.5	40.5	66.7	91.8
	LightNER	**62.9**	**75.6**	**78.8**	**82.2**	**84.5**	**85.7**	**58.1**	**67.4**	**69.5**	**73.7**	**78.4**	**81.1**	**86.9**	**89.4**	**93.9**

Table 2: Model performance (F1 score) in the cross-domain low-resource setting. "†" indicates that we rerun their public code in this setting. All of our experiments and baselines adopt large version of LMs.

Transfer Knowledge from a General Domain to Specific Domains We observe that the performance of prototype-based methods remains approximately the same as the number of labeled data increases, while LightNER continues to improve when the number of target-domain data increases. Table 2 shows that on all three target-domain datasets, LightNER significantly outperforms the other three types of baselines in the case of both 10 and 500 instances per entity type, From the perspective of quantifying the **knowledge transferred**, when the number of instances is 10, the performance of our model increase the F1-scores to 21.2%, 9.6%, and 10.6% on the MIT movie, MIT restaurant, and ATIS datasets, respectively, which are better than the results of knowledge transferred by *LC-BERT*. This demonstrates that our model is more successful in transferring the knowledge learned from the source domain.

Source	Methods	*MIT Restaurant*		
		10	20	50
None	Ours [BART]	48.5	58.0	62.0
	- pluggable module	**50.3**	**59.4**	**63.5**
	- unified learnable verbalizer	45.5	55.5	59.8
	Ours (Full-params Tuning)	49.5	59.0	62.8
	LC-BERT	21.8	39.4	52.7
	LC-BERT+[P-tuning]	24.9	41.2	53.5
	LC-BERT+[Adapter]	11.5	14.3	21.2
	Ours+[Adapter]	43.3	52.3	58.5
CoNLL03	Ours [BART]	**58.1**	**67.4**	**69.5**
	- pluggable module	54.5	64.2	67.8
	- unified learnable verbalizer	48.7	58.8	62.5
	Ours (Full-params Tuning)	53.7	63.5	66.9
	LC-BERT	27.2	40.9	56.3
	LC-BERT+[P-tuning]	30.3	46.8	58.2
	LC-BERT+[Adapter]	13.0	16.2	21.8
	Ours+[Adapter]	46.8	58.2	62.5

Table 3: Performance of Ablation and Variants Study.

5.3 Ablation and Comparison

As shown in the above experiments that our LightNER possess the outstanding ability of knowledge transfer in the low-resource setting, we demonstrate that the pluggable guidance module contributes to the cross-domain improvement. To this end, we ablate the pluggable module and unified learnable verbalizer to validate the effectiveness. - *pluggable module* indicates the entire parameter (100%) tuning without our proposed pluggable module. - *unified learnable verbalizer* donates our model only randomly assigns one token in the vocabulary to represent the type. From Table 3, we notice that only - *pluggable module* in the vanilla few-shot setting performs a little better than LightNER, but **decreases significantly in the cross-domain few-shot setting**. However, - *unified learnable verbalizer* drop both in the two settings. It further demonstrates that the design of the pluggable module is parameter-efficient and **beneficial for knowledge transfer**, while unified learnable verbalizer can **handle class transfer**, which is also essential for low-resource NER.

We further compare LightNER with several variants of our method: (i)*Ours (Full-params Tuning)*; (ii)*LC-BERT+[P-tuning]*: we set the length of continuous template words to be 10 for *P-tuning*; (iii)*LC-BERT+[Adapter]*; (iv)*Ours+[Adapter]*; **Firstly**, compared with LightNER, training all the parameters of our model merely improve a little in in vanilla few-shot setting, but drops significantly in cross-domain few-shot settings, which reveals that our pluggable module with LMs fixed is the vital for transferring knowledge across domains. **Secondly**, we observe that LC-BERT equipped with P-tuning achieves a few improvements both in vanilla few-shot and cross-domain few-shot settings. While Adapter makes performance drop significantly because LC-BERT cannot handle the class transfer, thus the few tuned parameters yield unsatisfactory performance. **Finally**, we replace

Methods	*NER → POS*			*POS → NER*		
	10	20	Full	10	20	Full
LC-BERT	44.3/46.2	53.7/54.3	91.4/91.7	37.9/38.3	48.4/48.6	91.7/91.3
LightNER	45.5/50.6	54.4/57.7	91.3/93.2	46.5/51.8	61.8/65.2	92.9/93.5

Table 4: Model performance in the cross-task setting. Number before and after "/" donate the F1 scores of training from scratch on target domain and transferring from source task to target task respectively.

Target	CoNLL			Movie			Restaurant		
Source	M	R	Mix	C	R	Mix	C	M	Mix
LC-BERT	0.2	0.4	0.0	0.5	0.3	0.0	0.3	0.2	0.0
Template	0.1	0.2	0.0	0.3	0.2	0.0	0.1	0.0	0.0
LightNER	**8.5**	**8.8**	**15.8**	**12.6**	**9.0**	**18.9**	**11.0**	**8.5**	**18.4**

Table 5: Cross-domain zero-shot performancefootnoteIn zero-shot setting, the weight of the unified learnable verbalizer is average operation.. C, M, and R refer to the dataset of CoNLL03, Movie, and Restaurant, respectively. The Mix column refers to the methods of averaging the parameters from the other two source domains (average the prompt for LightNER).

the pluggable module with the Adapter to validate the effectiveness of our module. The fact that *Ours+[Adapter]* performs significantly better than *LC-BERT+[Adapter]* demonstrates the superiority of our generative framework. Besides, *Ours+[Adapter]* behaves unsatisfactorily in a cross-domain low-resource setting, which reveals its poor ability of knowledge transfer for NER.

5.4 Detailed Model Analysis

The Transferability Across Task Although our LightNER is designed for NER, it is easy to generalize to other sequence tagging tasks without any modification network structure.

Thus, we try to train on full data of the source task, and then simply **load the pluggable guidance module** to further train the model on the target task. As shown in Table 4, we find LC-BERT has an extensive performance drop of all tasks in a cross-task setting, and we believe this is due to the task-specific classifier head hindering the generalization. The excellent performance in the cross-task setting proves that LightNER can adapt to other sequence labeling tasks and incredibly transfer knowledge across tasks.

Cross-Domain Zero-Shot Analysis with Mixed Guidance Parameters We leverage one dataset as the source domain and conduct the zero-shot experiments on target domains. From Table 5, we observe that our method can achieve F1-scores of approximately 10% in the cross-domain zero-shot setting, significantly higher than other methods. we further attempt to investigate the performance of mixing different pluggable guidance module. Specifically, we directly average the parameters of prompts from two source domains as a mixed prompt for the target domain and insert it into the generative framework to evaluate the target performance. From Table 5, we notice that mixed prompt achieves promising improvement, which is close to the addition of the results of the original two sources prompt-based model. We argue that this finding may also inspire future research directions of prompt-tuning and data augmentation.

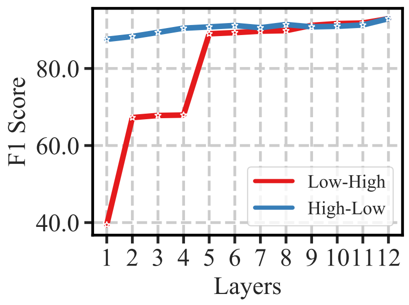


Figure 3: Performances on CoNLL03 as the layers of guidance module varies.

Low-high Layer vs. High-low Layer In the aforementioned experiments, we assign the pluggable guidance module to all layers in PLM. However, it is intuitive to investigate which layer is more sensitive with our approach. Intuitively, basic syntactic information may appear earlier in the PLM, while high-level semantic information emerges in higher-level layers (Clark et al., 2019). We conduct experiments by applying our pluggable module from the lowest to the highest layer and from the highest to the lowest layer separately. These two progressive methods are briefly denoted as low-high and high-low, respectively. As Figure 3 shows, the performance on CoNLL-2003 is close to the original result obtained after adding full-layer guidance module for tuning. This phenomenon also appears in the cross-domain few-shot setting. (Detailed results refer to Section 3 of Appendix C.2.). This proves that guidance module applied to higher layers of LMs can better stimulate knowledge from PLMs for downstream tasks more efficiently.

6 Conclusion and Future Work

In this paper, we propose a lightweight tuning paradigm for low-resource NER via pluggable prompting (LightNER), which can accomplish the class transfer and domain transfer for low-resource NER without modifying the PLM's parameters and architecture. Note that we only tune pluggable guidance module with the whole parameter of the PLMs fixed, thus, making our approach lightweight and flexible for low-resource scenarios and can better transfer knowledge across domains and tasks. Experimental results reveal that LightNER can obtain competitive results in the rich-resource setting and outperform baselines in the low-resource setting. In the future, we plan to leverage knowledge graphs to enhance the pluggable guidance module for better knowledge transfer performance.

Acknowledgments

We want to express gratitude to the anonymous reviewers for their kind comments. This work was supported by National Natural Science Foundation of China (No.62206246, 91846204 and U19B2027), Zhejiang Provincial Natural Science Foundation of China (No. LGG22F030011), Ningbo Natural Science Foundation (2021J190), and Yongjiang Talent Introduction Programme (2021A-156-G).

Ethics/Broader Impact Statement

The ability to extract useful and important information from text, without the need for human input or control, has a wide range of practical and industrial applications. Named Entity Recognition is an important information extraction task that can benefit many NLP applications, e.g., information retrieval, dialog generation, and question answering. However, traditional fine-tuning-based approaches achieve poor performance in situations in which there are sparse data available. Our simple-yet-effective approach makes it possible to obtain better performance in these low-resource settings and we detail its advantages as:

(i) The few-shot setting is realistic (the number of labeled instance per class K can be any variable);

(ii) No necessity of prompt engineering;

(iii) Extensible to any pre-trained LMs (e.g., BERT or GPT-2).

Our motivation is to develop an applicable, generalizable lightweight tuning paradigm for the NLP community and our work is but a small step towards this direction.

References

Oshin Agarwal, Yinfei Yang, Byron C. Wallace, and Ani Nenkova. 2021. Interpretability analysis for named entity recognition to understand system predictions and how they can improve. *Comput. Linguistics*, 47(1):117–140.

Eyal Ben-David, Nadav Oved, and Roi Reichart. 2022. PADA: Example-based Prompt Learning for on-the-fly Adaptation to Unseen Domains. *Transactions of the Association for Computational Linguistics*, 10:414–433.

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

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using BART. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 1835–1845. Association for Computational Linguistics.

Leyang Cui and Yue Zhang. 2019. Hierarchically-refined label attention network for sequence labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 4113–4126. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. 2021. Few-nerd: A few-shot named entity recognition dataset. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 3198–3213. Association for Computational Linguistics.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Alexander Fritzler, Varvara Logacheva, and Maksim Kretov. 2019. Few-shot classification in named entity recognition task. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019, Limassol, Cyprus, April 8-12, 2019*, pages 993–1000. ACM.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 3816–3830. Association for Computational Linguistics.

Tao Gui, Jiacheng Ye, Qi Zhang, Zhengyan Li, Zichu Fei, Yeyun Gong, and Xuanjing Huang. 2020. Uncertainty-aware label refinement for sequence labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 2316–2326. Association for Computational Linguistics.

Demi Guo, Alexander M. Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4884–4896. Association for Computational Linguistics.

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM. In *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12, 2016*, pages 715–719. ISCA.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2021. PTR: prompt tuning with rules for text classification. *CoRR*, abs/2105.11259.

Matthew Henderson and Ivan Vulic. 2021. Convex: Data-efficient and few-shot slot labeling. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 3375–3389. Association for Computational Linguistics.

Yutai Hou, Wanxiang Che, Yongkui Lai, Zhihan Zhou, Yijia Liu, Han Liu, and Ting Liu. 2020. Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 1381–1393. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Jiaxin Huang, Chunyuan Li, Krishan Subudhi, Damien Jose, Shobana Balakrishnan, Weizhu Chen, Baolin Peng, Jianfeng Gao, and Jiawei Han. 2020. Few-shot named entity recognition: A comprehensive study. *CoRR*, abs/2012.14978.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online,* *July 5-10, 2020*, pages 5849–5859. Association for Computational Linguistics.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2019. Sequence-to-nuggets: Nested entity mention detection via anchor-region networks. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5182–5192. Association for Computational Linguistics.

Jingjing Liu, Panupong Pasupat, Scott Cyphers, and James R. Glass. 2013. Asgard: A portable architecture for multilingual dialogue systems. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pages 8386–8390. IEEE.

Kun Liu, Yao Fu, Chuanqi Tan, Mosha Chen, Ningyu Zhang, Songfang Huang, and Sheng Gao. 2021a. Noisy-labeled NER with confidence estimation. In *Proceedings of NAACL*, pages 3437–3445. Association for Computational Linguistics.

Kun Liu, Yao Fu, Chuanqi Tan, Mosha Chen, Ningyu Zhang, Songfang Huang, and Sheng Gao. 2021b. Noisy-labeled NER with confidence estimation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 3437–3445. Association for Computational Linguistics.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021c. GPT understands, too. *CoRR*, abs/2103.10385.

Yijin Liu, Fandong Meng, Jinchao Zhang, Jinan Xu, Yufeng Chen, and Jie Zhou. 2019. GCDT: A global context enhanced deep transition architecture for sequence labeling. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2431–2441. Association for Computational Linguistics.

Ruotian Ma, Xin Zhou, Tao Gui, Yiding Tan, Linyang Li, Qi Zhang, and Xuanjing Huang. 2022. Template-free prompt tuning for few-shot NER. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 5721–5732. Association for Computational Linguistics.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Clifton Poth, Jonas Pfeiffer, Andreas Rücklé, and Iryna Gurevych. 2021. What to pre-train on? efficient intermediate task selection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 10585–10605. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 142–147. ACL.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Timo Schick, Helmut Schmid, and Hinrich Schütze. 2020. Automatically identifying words that can serve as labels for few-shot text classification. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 5569–5578. International Committee on Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 255–269. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083. Association for Computational Linguistics.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4222–4235. Association for Computational Linguistics.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate entity recognition with iterated dilated convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2670–2680. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Sam Wiseman and Karl Stratos. 2019. Label-agnostic sequence labeling by copying nearest neighbors. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5363–5369. Association for Computational Linguistics.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6442–6454. Association for Computational Linguistics.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various NER subtasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 5808–5822. Association for Computational Linguistics.

Jie Yang, Shuailong Liang, and Yue Zhang. 2018. Design challenges and misconceptions in neural sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 3879–3889. Association for Computational Linguistics.

Yi Yang and Arzoo Katiyar. 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6365–6375. Association for Computational Linguistics.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6470–6476. Association for Computational Linguistics.

Jeffrey O. Zhang, Alexander Sax, Amir Roshan Zamir, Leonidas J. Guibas, and Jitendra Malik. 2020a. Side-tuning: A baseline for network adaptation via additive side networks. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part III*, volume 12348 of *Lecture Notes in Computer Science*, pages 698–714. Springer.

Ningyu Zhang, Shumin Deng, Zhen Bi, Haiyang Yu, Jiacheng Yang, Mosha Chen, Fei Huang, Wei Zhang, and Huajun Chen. 2020b. Openue: An open toolkit of universal extraction from text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 1–8. Association for Computational Linguistics.

Xin Zhou, Ruotian Ma, Tao Gui, Yiding Tan, Qi Zhang, and Xuanjing Huang. 2021. Plug-tagger: A pluggable sequence labeling framework using language models. *CoRR*, abs/2110.07331.

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

A Detailed Statistics of Datasets

We take the standard split of CoNLL03 by following Sang and Meulder (2003), and splits MIT Movie Review, MIT Restaurant Review and ATIS by following Liu et al. (2013). Table 6 presents detailed statistics of our datasets. The standard precision, recall and F1 score are used for model evaluation.

Dataset	# Train	# Test	# Entity
CoNLL03	12.7k	3.2k	4
MIT Restaurant	7.6k	1.5k	8
MIT Review	7.8k	2k	12
ATIS	4.6k	850	79

Table 6: Statistic of datasets.

Models	PER	ORG	LOC*	MISC*	Overall
LC-BERT	76.25	75.32	61.55	59.35	68.12
LC-BART	75.70	73.59	58.70	57.30	66.82
Template	84.49	72.61	71.98	73.37	75.59
LightNER	**90.96**	**76.88**	**81.57**	**82.08**	**78.97**

Table 7: In-domain low-resource performance on the CoNLL-2003 dataset. * indicates the low-resource entity type.

B Experimental Details

This section details the training procedures and hyperparameters for each of the datasets. Considering the instability of the few-shot learning, we run each experiment 5 times on the random seed [1, 2, 49, 4321, 1234] and report the averaged performance. We utilize Pytorch to conduct experiments with 1 Nvidia 3090 GPUs. All optimizations are performed with the AdamW optimizer with a linear warmup of learning rate over the first 10% of gradient updates to a maximum value, then linear decay over the remainder of the training. We set the hyper-parameter α as 0.5. And weight decay on all non-bias parameters is set to 0.01. We describe the details of the training hyper-parameters in the following sections.

B.1 Standard Supervised Setting

For all models, we fix the batch size as 16 and search for the learning rates in varied intervals [1e-5, 5e-5]. We train the model for 30 epochs and do evaluation after 20 epoch. We choose the model performing the best on the validation set and evaluate it on the test set.

B.2 Low-Resource Setting

We fix the batch size as 16 and search for the learning rates in varied intervals [3e-5, 5e-5]. We train the model for 30 epochs and do evaluation after 20 epoch. We choose the model performing the best on the validation set and evaluate it on the test set.

B.3 Cross-Task Setting

We fix the batch size as 8 and search for the learning rates in varied intervals [2e-5, 5e-5]. We train the model for 30 epochs and do evaluation after 25 epoch. We choose the model performing the best on the validation set and evaluate it on the test set.

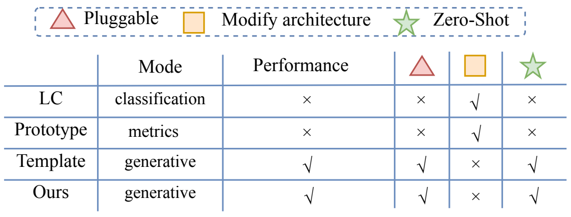
C Supplementary Experimental Results

C.1 In-Domain Low-Resource NER Setting

Following (Cui et al., 2021), we construct few-shot learning scenarios on CoNLL-2003 by down-sampling, which limits the number of training instances for certain specific categories. Particularly, we choose " LOC" and "MISC" as the low-resource entities and "PER" and "ORG" as the rich-resource entities. The rich and low-resource entity categories have the same textual domain. Specifically, we downsample the CoNLL-2003 training set and generate 4,001 training instances, including 2,496 "PER," 3,763 "ORG," 50 "MISC," and 50 "LOC" entities. As shown in Table 7, our method outperforms other methods for both rich- and low-resource entity types. This proves that our proposed method has a more substantial performance for in-domain few-shot NER and demonstrates that it can effectively handle the class transfer, which is a challenging aspect in few-shot NER tasks.

C.2 The Performance in the Low-Resource Setting When the Prompt Layer Varies

Intuitively, basic syntactic information may appear earlier in the PLM, while high-level semantic information emerges in higher-level layers. Table 8 shows that the performance of prompts within highest 1 layer is better than lowest 1 layer overall, and the performance of highest 6 layers is close to the result of all 12 layers. This proves that prompts applied to higher layers of LMs can better stimulate knowledge from PLMs for downstream tasks more efficiently.



△ Pluggable □ Modify architecture ☆ Zero-Shot

	Mode	Performance	△	□	☆
LC	classification	×	×	√	×
Prototype	metrics	×	×	√	×
Template	generative	√	√	×	√
Ours	generative	√	√	×	√

Figure 4: We show the formulations of different NER models and illustrate their corresponding strengths. Zero-shot refers to zero-shot learning ability; LC is short for label-specific classifier (vanilla sequence labeling)

Algorithm 1 Decoding Algorithm to Convert the Entity Index Sequence into Entity Spans

```
Require: $n$, the number of tokens in $X$; $m$, the number of entity types; target sequence $Y = [y_1, ..., y_{3l}]$, $l$ is the number of entities; and we have $y_t \in [1, n+m]$
Ensure: Entity spans $E = \{(e_1^{start}, e_1^{end}, t_1), ..., (e_i^{start}, e_i^{end}, t_i)\}$
 1: $E = \{\}, e = [], i = 1$
 2: while $i <= 3l$ do
 3:     $y_i = Y[i]$
 4:     if $y_i > n$ then
 5:         $E.add((e, C_{y_i - n}))$
 6:         $e = []$
 7:     else
 8:         $e.append(y_i)$
 9:     end if
10:     $i += 1$
11: end while
12: return $E$
```

C.3 Impact of Length of Guidance Module

We set the length of prompts as 10 in the above experiment and analyze whether the impact of the length of the prompt. From Figure 5, we notice that

Source	Methods	*MIT Movie*			*MIT Restaurant*			*ATIS*		
		10	20	50	10	20	50	10	20	50
None	Template	37.3	48.5	52.2	46.0	57.1	58.7	71.7	79.4	92.6
	LightNER(lowest 1 layer)	16.3	20.3	30.5	14.6	23.4	25.4	30.6	38.3	44.2
	LightNER(highest 1 layer)	29.5	38.4	45.5	35.4	45.3	50.5	60.1	69.8	78.7
	LightNER(highest 6 layers)	38.5	50.5	69.8	44.3	55.7	59.8	70.2	80.2	88.4
	LightNER(all 12 layers)	**41.7**	**57.8**	**73.1**	**48.5**	**58.0**	**62.0**	**76.3**	**85.3**	**92.8**
CoNLL03	Template	42.4	54.2	59.6	53.1	60.3	64.1	77.3	88.9	93.5
	LightNER(lowest 1 layer)	24.3	30.5	35.4	15.6	22.4	27.5	37.9	44.5	48.3
	LightNER(highest 1 layer)	44.6	59.3	74.3	39.4	45.2	51.7	59.7	68.5	79.2
	LightNER(highest 6 layers)	55.8	69.7	75.8	50.7	62.7	66.7	79.2	86.3	91.8
	LightNER(all 12 layers)	**62.9**	**75.6**	**78.8**	**58.1**	**67.4**	**69.5**	**86.9**	**89.4**	**93.9**

Table 8: Performances in cross-domain low-resource setting as the prompt layer varies.

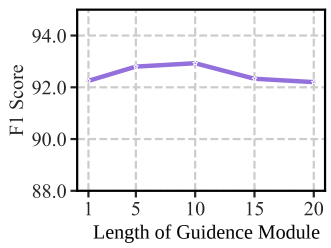


Figure 5: Performances on CoNLL03 as the length of guidance module varies.

a longer prompt implies more trainable parameters but does not guarantee more expressive power. It also reveals that our pluggable guidance module is stable; as the length changes, the performance fluctuation does not exceed 1%.

D Comprehensive Comparison

We carry out a comprehensive comparison with related methods as shown in Figure 4. For a given sequence, the computational complexity of our LightNER is $O\left(n^2 d\right)$ and Tamplate-based BART is $O\left(nm\hat{n} \cdot n^2 d\right)$,where d donates the dimension of the LMs; $\mathbf{n}, \mathbf{m}, \hat{\mathbf{n}}$ imply the length of input, number of entity classes and n-grams, respectively. Note that our approach does not need to enumerate all possible spans and construct templates, which is efficient than the Template-based method (Cui et al., 2021). Moreover, we only tune 2.2% parameters of the whole model (the tuned params divided by params of the LM), making it memory efficient during training.

E The decoding algorithm for converting process

The decoding algorithm for converting the predicted index sequence to entity spans is shown in Algorithm 1.

F Sampling strategy in low-resource setting

F.1 Cross-Domain

We simulate the cross-domain low-resource data scenarios by random sampling training instances from a large training set as the training data in the target domain. We use different numbers of instances for training, randomly sampling a number of instances per entity type (10, 20, 50, 100, 200, 500 instances per entity tag for MIT Movie and MIT restaurant, and 10, 20, 50 instances per entity tag for ATIS). For different instances per entity tag, we sample five times on the random seed [1, 2, 49, 4321, 1234] and report the averaged performance.

In order to alleviate the problem that an instance usually contains multiple entities, we first sort the entity tags according to the number of instances included. Then we sample instances in the sequence of the sorted order. After once sampling, we will update the status(the remaining number of instances to be sampled) of the entity tags. If an entity tag exceeds the limit after the sampling, discard this sampling.

F.2 Cross-Task

Since the CoNLL-2003 dataset contains both entity tag and POS information, we use the same sampling data in cross-task setting. For different instances per entity tag, we sample five times on the random seed [1, 2, 49, 4321, 1234] and report the averaged performance.