# Learning In-context Learning for Named Entity Recognition

**Jiawei Chen**[1,4,*], **Yaojie Lu**[1,†], **Hongyu Lin**[1], **Jie Lou**[3], **Wei Jia**[3], **Dai Dai**[3], **Hua Wu**[3], **Boxi Cao**[1,4], **Xianpei Han**[1,2,†], **Le Sun**[1,2]
[1]Chinese Information Processing Laboratory [2]State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences, Beijing, China
[3]Baidu Inc., Beijing, China
[4]University of Chinese Academy of Sciences, Beijing, China
{jiawei2020,yaojie,hongyu,boxi2020,xianpei,sunle}@iscas.ac.cn
{loujie,jiawei07,daidai,wu_hua}@baidu.com

## Abstract

Named entity recognition in real-world applications suffers from the diversity of entity types, the emergence of new entity types, and the lack of high-quality annotations. To address the above problems, this paper proposes an in-context learning-based NER approach, which can effectively inject in-context NER ability into PLMs and recognize entities of novel types on-the-fly using only a few demonstrative instances. Specifically, we model PLMs as a meta-function $\lambda_{\text{instruction, demonstrations, text}}.\mathcal{M}$[1], and a new entity extractor can be implicitly constructed by applying new instruction and demonstrations to PLMs, i.e., $(\lambda.\mathcal{M})(\text{instruction, demonstrations}) \rightarrow \mathcal{F}$ where $\mathcal{F}$ will be a new entity extractor, i.e., $\mathcal{F}$: text $\rightarrow$ entities. To inject the above in-context NER ability into PLMs, we propose a meta-function pre-training algorithm, which pre-trains PLMs by comparing the (instruction, demonstration)-initialized extractor with a surrogate golden extractor. Experimental results on 4 few-shot NER datasets show that our method can effectively inject in-context NER ability into PLMs and significantly outperforms the PLMs+fine-tuning counterparts.

## 1 Introduction

Named entity recognition (NER) aims to detect and classify named entities in text, such as *People*, *Disease*, and *Movie*. Traditional NER methods (Lample et al., 2016; Li et al., 2020c; Yan et al., 2021) have achieved remarkable success when entity types are pre-defined and massive high-quality annotations are provided. Unfortunately, real-world NER still suffers from the diversity of entity types (e.g., the extraction of *Movie* is very different to *Disease*), the emergence of new entity types (e.g., *Virus* of Cov-19 ), and the lack of high-quality annotations.

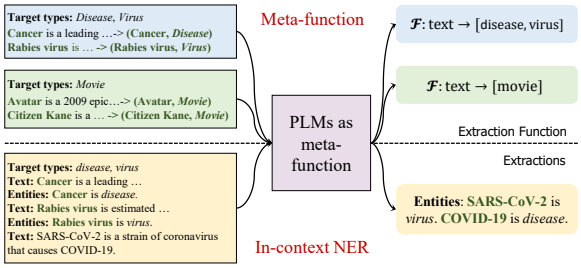


Figure 1: Illustration of in-context NER, which uses instruction, demonstrations, and text as input to identify entities. The in-context learning model can be regarded as a meta-function that takes instruction and demonstrations as input and produces an entity extractor capable of identifying the desired entities (Akyürek et al., 2022).

To address these problems, recent studies often employ few-shot learning techniques, including fine-tuning-based and metric-based methods. Fine-tuning-based methods extract entities of new types by adjusting model weights using new instances (Ma et al., 2022a; Chen et al., 2022a; Das et al., 2022). The main drawbacks of these methods are that re-training is often expensive (especially for large-scale models) and new entity types cannot be addressed on-the-fly. Metric-based methods are free from updating parameters and identifying entities by learning to compare query instances with support instances (or prototypes) (Yang and Katiyar, 2020; Tong et al., 2021). These methods are limited to the matching architectures and are sensitive to domain shift since they do not fully explore the information of target domain (Ma et al., 2022c).

In this paper, we propose an in-context learning-

*This work was partially done when Jiawei Chen interned at Baidu.

†Corresponding authors.

[1]This paper represents functions using lambda-calculus (Barendregt, 1992), and each function is represented as $\lambda_{x,y,z}.M$, where $x, y, z$ are variables and $M$ is function definition/abstraction. The function can apply to arguments such as $(\lambda_{x,y,z}.M)(x = A, y = B, z = C)$ (fully applied) or $(\lambda_{x,y,z}.M)(x = A, y = B)$ (partially applied).



*Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*
*Volume 1: Long Papers*, pages 13661–13675
July 9-14, 2023 

based NER approach, which can effectively address the above problems by injecting in-context NER ability into PLMs and then recognizing entities of new types on-the-fly using only a few demonstrative instances. Specifically, we model PLMs as a meta-function (Akyürek et al., 2022) for NER $\lambda_{\text{instruction, demonstrations, text}}.\mathcal{M}$, and a new entity extractor can be implicitly constructed by applying new instruction and demonstrations to PLMs, i.e., $(\lambda.\mathcal{M})(\text{instructions, demonstrations}) \rightarrow \mathcal{F}$ where $\mathcal{F}$ will be a new entity extractor $\mathcal{F}$: text $\rightarrow$ entities. For example, in Figure 1, our method can construct entity extractors of new *Disease* and *Virus* types on-the-fly by applying PLMs using demonstrations such as "Text: Cancer is a leading cause of death worldwide. Entities: Cancer is disease". Furthermore, we propose a meta-function pre-training algorithm to inject the above in-context NER ability into PLMs. The algorithm pre-trains PLMs by comparing the implicitly (instruction, demonstration)-constructed extractor with an explicitly fine-tuned surrogate golden extractor. The comparison ensures that the meta-function $(\lambda.\mathcal{M})$ will generate an entity extractor $\mathcal{F}$ from instructions and demonstrations as accurately as possible.

The proposed method can seamlessly leverage the powerful language understanding and generation capabilities of large-scale PLMs (Brown et al., 2020), effectively address diverse and new entity types through in-context learning, and only requires a couple of demonstrations for each entity type. Compared to fine-tuning methods, our method does not require expensive retraining, and new entity types can be extracted on-the-fly, with no need for model weight adjusting. Compared with metric-based methods, our method can dynamically utilize the information entailed in instruction and demonstrations rather than be limited to the fixed metric space.

To verify the effectiveness of our method, we further pre-train PLMs using a large-scale distantly annotated NER dataset from Wikipedia and Wikidata. Experimental results on 4 few-shot NER benchmarks show that our method can effectively inject in-context NER ability into PLMs and significantly outperforms the PLMs+fine-tuning counterparts[2].

In general, this paper's main contributions are:

- We propose an in-context NER method that can effectively extract entities of novel types on-the-fly using only a few demonstrative instances.
- We design a meta-function pre-training algorithm, which models PLMs as a meta-function and injects in-context NER ability into PLMs by comparing the (instruction, demonstration)-constructed extractor with a surrogate golden extractor.
- How to inject in-context ability into small models is an important research direction of NLP in the big model era. Our work can benefit new directions for future works.

[2]Our source codes are openly available at https://github.com/chen700564/metaner-icl

## 2 Related work

**Few-shot NER** Few-shot learning is a promising technique for low-resource NER. Currently, there are two main categories of FS-NER methods: fine-tuning-based methods and metric-based methods. Fine-tuning-based FS-NER methods re-train NER models using new instances. Metric-based methods identify entities by pre-training to compare query instances with support instances (Snell et al., 2017; Fritzler et al., 2019; Yang and Katiyar, 2020; Tong et al., 2021; Wang et al., 2022; Ji et al., 2022) using given NER datasets. FS-NER is a challenging task, and several improvements have been proposed to enhance its performance. These include leveraging label information (Hou et al., 2020; Wang et al., 2021a; Lu et al., 2022b; Ma et al., 2022a; Chen et al., 2022a; Yang et al., 2022), designing new paradigms such as decomposition methods (Ji et al., 2022; Ma et al., 2022c; Yang et al., 2022), prompt-based methods (Cui et al., 2021; Liu et al., 2022; Ma et al., 2022b), and demonstration-based methods (Lee et al., 2022; Zhang et al., 2022a); , and proposing new learning strategies like meta-learning (Li et al., 2020a,b; de Lichy et al., 2021; Ma et al., 2022c), contrastive learning (Das et al., 2022), and self-training (Huang et al., 2021; Wang et al., 2021b). In this paper, we address FS-NER via in-context learning (Gutiérrez et al., 2022), which empowers PLMs with in-context learning ability and entities of new entity types can be extracted on-the-fly.

**In-context learning** The in-context learning ability has been observed in large-scale PLMs such as GPT-3 (Brown et al., 2020), and has been widely applied in different tasks such as "chain of thought" reasoning (Wei et al., 2022). Recent studies

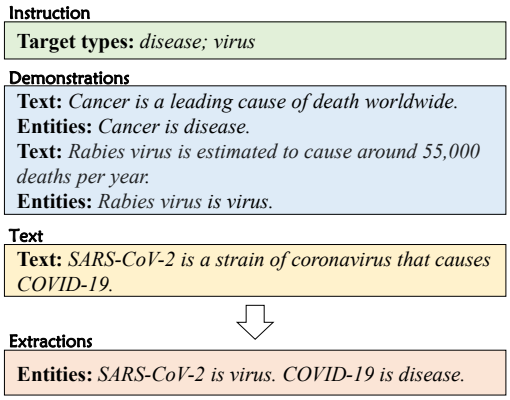


Figure 2: The formats of input and output of in-context few-shot NER. The input is formed by instruction, demonstrations, and text.

aim to enhance in-context learning by selecting valuable demonstrations (Liu et al., 2021; Rubin et al., 2022), optimizing the order of demonstrations (Lu et al., 2022a), and calibrating output distributions (Zhao et al., 2021). Some studies try to replicate in-context learning in smaller models (Min et al., 2022a; Chen et al., 2022b). Additionally, some researchers attempt to replicate in-context learning using smaller models (Min et al., 2022b; Chan et al., 2022). Furthermore, there are efforts to understand the underlying mechanisms (Akyürek et al., 2022) of in-context learning which suggest that it can be compared to a meta-function and facilitate implicit fine-tuning (Dai et al., 2022; von Oswald et al., 2022). This paper is inspired by previous studies and considers in-context named entity recognition (NER) as a meta-function. To enhance the ability of pre-trained language models (PLMs) to perform in-context NER, we propose an effective pre-training algorithm. Unlike MetaICL (Min et al., 2022a), which only transforms multi-task learning into the form of in-context learning for pre-training, our approach also includes meta-function pre-training (Section 4.3) based on the underlying mechanisms of in-context learning.

## 3 In-context Named Entity Recognition

This section describes how to recognize entities through in-context NER. In in-context learning, the model will read the information of target entity types from both instruction and demonstrations, and then extract entities of target types within the text. In this way, new entity types can be extracted on-the-fly, without the need for model retraining.

Concretely, this paper formulates in-context NER as a sequence-to-sequence generation process. The input $X = [I; D; T]$ includes instruction $I$, demonstrations $D$, and text $T$ while the output is a list of extracted entities $Y = [e_1, ..., e_n]$. Figure 2 shows an example, where an in-context NER model will identify that the target entity types are *Disease* and *Virus*, distill the knowledge about these types from demonstrations(e.g., the context patterns of a disease), and finally recognize "SARS-CoV-2" as virus and "COVID-19" as disease using the above knowledge. The details are described as follows.

**Instruction** The instruction is a sequence of target entity types, guiding the model to extract what entity types (Min et al., 2022a). The instruction for target entity types $\{l_1, \ldots, l_n\}$ is $I$ ="Target types: $l_1; \ldots; l_n$". For example, in Figure 2 the instruction is "Target types: disease; virus".

**Demonstrations** Demonstrations provide the intra-class knowledge of target entity types (e.g., entity semantics and context patterns) and illustrate the form of outputs. As shown in Figure 2, the demonstrations contain the illustrative instances for different target types, and each instance is "Text: {text} Entities: {extractions}", where {extractions} are entities presented in the {text}.

**Extractions** The output of the extraction process is a list of entities, denoted as $Y = [e_1, \ldots, e_n]$ where $e_i$ is $i$-th extracted entities. Each extraction $e$ is represented as "ENTITY is *type*". For instance, in Figure 2, the extraction "COVID-19 is disease." indicates that "COVID-19" is an entity mention with the type "Disease". This natural language-like representation allows us to better utilize the text generation capabilities of pre-trained language models. During inference, we locate all mentions in the text and further output their locations.

**Architecture** Given the above task formulation, we employ an encoder-decoder network like T5 (Raffel et al., 2020), where the encoder encodes <instruction, demonstrations, text> and the decoder generates all extractions as a tokenized text sequence $Y = [y_1, \ldots, y_n]$.

The success of in-context NER depends on two critical abilities: the in-context learning ability and the extraction ability. For in-context learning, the models should be able to implicitly construct accurate extractors of new entity types by following the instruction and capturing the knowledge in demon-

strations. In this way, we can see a PLM as a meta-function, i.e., a function of extractors whose input is (instruction, demonstrations) and whose output is an entity extractor. For extraction, the models should be able to locate specific spans and categorize them into target entity types. The following section demonstrates how to inject such an in-context learning ability into PLMs and construct an effective in-context NER model.

## 4 Meta-Function Pre-training for In-Context NER

In this section, we will explain how to incorporate in-context named entity recognition (NER) capabilities into pre-trained language models (PLMs). Although large-scale PLMs like GPT-3 have demonstrated the ability to learn in-context, this capability is not always controllable or predictable. Additionally, unlike classification and question-answering tasks that align with the pre-training objective of language models (i.e., producing natural text output), NER requires more complex span extraction and type specification. As a result, Gutiérrez et al. (2022) show that LMs aren't well-suited for in-context NER tasks. In this paper, we propose meta-function pre-training, an algorithm that can inject in-context NER ability into PLMs in a controllable and predictable way.

Specifically, we model PLMs as a meta-function (Akyürek et al., 2022) for NER $\lambda_{\text{instruction, demonstrations, text}}.\mathcal{M}$, and a new entity extractor can be implicitly constructed by applying new instructions and demonstrations to PLMs, i.e., $(\lambda.\mathcal{M})(\text{instructions, demonstractions}) \rightarrow \mathcal{F}$ where $\mathcal{F}$ will be a new entity extractor $\mathcal{F}$:text $\rightarrow$ entities. Based on the meta-function formulation, we further pre-train PLMs for in-context NER abilities by:

- optimizing PLMs via a meta-function loss, so that the implicitly (instruction, demonstration)-constructed extractor $\mathcal{F}$ will be as close as an explicitly fine-tuned surrogate golden extractor;
- optimizing PLMs via an extraction loss, so that the in-context NER can effectively locate and categorize entities in a text.

The details are described in the following.

### 4.1 Pre-training Settings

**Pre-training Corpus Construction** To continually pre-train PLMs for in-context NER, we first collect an in-context pre-training NER corpus $\mathcal{D}_{\text{in-context}} = \{x_1, x_2, ..., x_n\}$, where each $x$ is an in-context NER task represented as a tuple = (instruction, demonstrations, text, entities).

Specifically, to sample in-context NER task $x$, we use traditional NER corpus $\mathcal{D}_{\text{NER}}$ where each NER instance is a (text, entities) pair as follows:

1. **In-context Task Sampling**: To construct an in-context NER task x = (instruction, demonstrations, text, entities): (1) we first sample $N$ target entity types from $\mathcal{D}_{\text{NER}}$ to form instruction and sample $K$ instances for each type to form demonstrations; (2) then we sample the text and the entities of x by either randomly sample an instance from $N$ target entity types, or randomly sample from instances of other entity types, i.e., their extractions are NIL. We sample NIL instances because in real-world applications many instances will not contain target entities, and NIL instances are sampled with a predefined proportion $\gamma$.
2. **Type Anonymization**: To ensure the models rely on in-context demonstrations for entity knowledge and avoid overfitting to entity type names, we anonymize entity types by randomly substituting them with a set of type indicators {<type1>, ..., <type99>}, rather than directly using the original type names such as *Disease* and *Virus*. We found this anonymization strategy can significantly improve the in-context learning ability of PLMs. Specifically, we randomly substitute each entity type name with pre-defined 99 type indicators {<type1>, ..., <type99>}, and the substitute probability for each name is 80%.

**Pre-training Loss** Based on the in-context pre-training corpus $\mathcal{D}_{\text{in-context}}$, we pre-train our in-context NER model by optimizing the loss:

$$\mathcal{L} = \alpha\mathcal{L}_{\text{meta-function}} + \mathcal{L}_{\text{extraction}} \quad (1)$$

where $\mathcal{L}_{\text{meta-function}}$ is the meta-function loss which ensures PLMs can implicitly generate accurate entity extractors (Section 4.2), $\mathcal{L}_{\text{extraction}}$ is the extraction loss which ensures PLMs have good extraction ability (Section 4.3), $\alpha$ is the coefficient of meta-function loss.

### 4.2 Meta-function Pre-training

As mentioned above, a good in-context NER model should be able to implicitly construct an accurate entity extractor by partially applying PLMs with

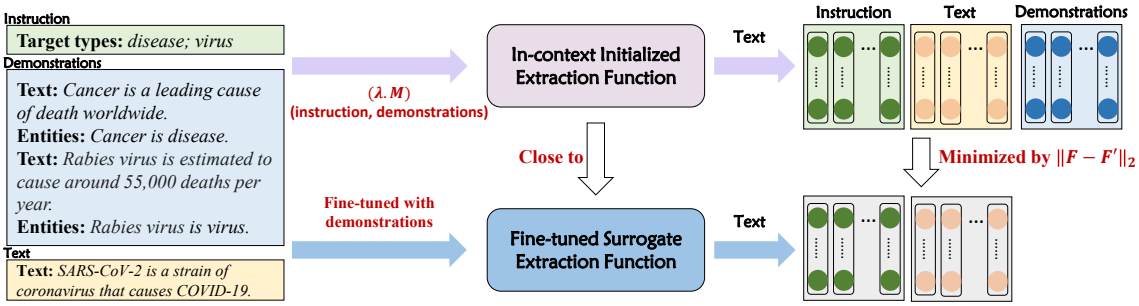


Figure 3: Overview of our meta-function pre-training. Our goal is to ensure that the extractor $\mathcal{F}$(instruction,demonstrations) closely resembles the golden extraction function. To obtain the golden extraction function, we use a surrogate strategy and the surrogate extraction function is the fine-tuned encoder using demonstrations.

instruction $I$ and demonstrations $D$:

$$(\lambda.M)(I, D) = \mathcal{F} \tag{2}$$

For example, given the instruction and demonstrations in Figure 2, we want PLMs to implicitly build an accurate extractor for *Disease* and *Virus*. Therefore if we know the golden extraction function $\mathcal{F}^*$ for target entity types, we can optimize PLMs for in-context NER ability by minimizing the distance $\|\mathcal{F}^* - \mathcal{F}\|$.

Unfortunately, the golden extraction function $\mathcal{F}^*$ is unknown. In this paper, we approximate $\mathcal{F}^*$ using a surrogate extractor which is the fine-tuned counterpart using demonstrations $D$. That is, for each in-context pre-training task $x$, we first recover all NER (text, entities) instances from $x$ as $x'$, then we fine-tune the model and use the fine-tuned encoder $\mathcal{F}'$ as the surrogate of $\mathcal{F}^*$. The overall meta-function pre-training is shown in Figure 3.

Formally, given instruction $I$, demonstration $D$, and text $T$, we first feed them into the encoder and obtain the feature of $I$ and $T$,

$$\mathbf{l}_1, ..., \mathbf{l}_n, \mathbf{d}_1, ..., \mathbf{d}_m, \mathbf{t}_1, ..., \mathbf{t}_k = Encoder(I; D; T) \tag{3}$$

Then we obtain the feature of the implicitly generated function $\mathcal{F}$ using the features of instruction $I$ and text $T$, and ignore the features of $D$: $\mathbf{F} = [\mathbf{l}_1, ..., \mathbf{l}_n, \mathbf{t}_1, ..., \mathbf{t}_k]$. In Figure 3, the feature $\mathbf{F}$ can be seen as the output of *Disease* and *Virus* extractor $\mathcal{F}$.

To obtain the feature of the fine-tuned counterpart, we perform a one-step gradient descent[3] on the encoder using the instances in the demonstration $D$ and get the surrogate encoder, which can be seen as an approximation of golden $\mathcal{F}^*$. Note that this fine-tuning operation is performed after the model has been copied, so there is no impact on the parameters of the original model. In the example in Figure 3, $Encoder'$ is a *Disease* and *Virus* extractor. After performing one-step updating, we feed instruction and text $[I; T]$ into the surrogate encoder to get their features:

$$\mathbf{F}' = Encoder'(I; T) \tag{4}$$

where $\mathbf{F}' = \{\mathbf{l}'_1, \ldots, \mathbf{l}'_n, \mathbf{t}'_1, \ldots, \mathbf{t}'_k\}$ is features of instruction $I$ and text $T$. In the example in Figure 3, the feature $\mathbf{F}'$ can be seen as the estimated output of golden extractor $\mathcal{F}^*$ for *Virus* and *Disease* entity types.

Then, we pre-train our in-context NER model to be a good meta-function by making the output of $F$ and $F^*$ consistent, i.e., minimizing the distance between $\mathbf{F}$ and $\mathbf{F}'$. The meta-function loss is:

$$\mathcal{L}_{\text{meta-function}} = \frac{1}{n+k} \sum_{i=1}^{n+k} d(\mathbf{F}_i, \mathbf{F}'_i) \tag{5}$$

where $d(\cdot)$ is euclidean distance. Note that when calculating the gradient of $\mathcal{L}_{\text{meta-function}}$, $\mathbf{F}'$ is seen as constant. To this end, the meta-function gradient can be estimated as:

$$\nabla\theta_{\text{encoder}} = \frac{\partial \mathcal{L}_{\text{meta-function}}}{\partial X} \tag{6}$$

where $\theta_{\text{encoder}}$ is the parameters of the encoder and $X = [I; D; T]$ is the input. The estimated gradient will be used to update the parameters of the encoder.

[3]We use "one-step gradient" because it strikes a balance between cost and effectiveness. While we believe that more steps may result in better performance, it would require additional time which we plan to explore in future work.

In this way, the in-context NER models will be trained to be a good meta-function (Akyürek et al., 2022), which can also be seen as an ability for implicit fine-tuning (Dai et al., 2022; von Oswald et al., 2022).

### 4.3 Extraction Function Pre-training

Besides the in-context learning ability, we also pre-train PLMs to be good extractors via extraction loss. Given instruction $I$, demonstrations $D$, and text $T$, the sequence-to-sequence entity extractor directly models the generation probability token by token in an auto-regressive way. Formally, we optimize the model parameters $\theta$ by minimizing the negative likelihood of in-context instances:

$$\mathcal{L}_{\text{extraction}} = -\log \prod_{i=1}^{|Y|} P(y_i|y_{<i}, X, \theta) \quad (7)$$

And the extraction gradient is computed as:

$$\nabla\theta = \frac{\partial \mathcal{L}_{\text{extraction}}}{\partial X} \quad (8)$$

To learn the above extraction ability, we design two extraction pre-training tasks, including an entity extraction task and a pseudo extraction language modeling task:

**Entity Extraction Task.** This task is used to train the ability to extract entities from text, we use both in-context NER settings whose input is (instruction, demonstrations, text) and traditional NER settings whose input is (instruction, text), and output is entities. Note that type anonymization is only conducted in in-context NER setting.

**Pseudo Extraction Language Modeling Task** . Because there is a mismatch between the entity extraction task and the original language modeling task, and the size of the NER corpus is usually far smaller than the text corpus for language modeling pre-training, we design a pseudo extraction LM task to bridge the above gap. Specifically, we randomly sample unlabeled sentences from the text corpus and automatically build pseudo extraction (instruction, demonstrations, text, pseudo entities) tasks. For instance, given a demonstration sentence such as "*I think this movie is cool and I really like it very much*" and a text "*I do not like it.*": (1) To begin with, we choose some spans from demonstrations (such as "this movie" and "like") and designate them as pseudo entities[4]. We assign random types to these entities from type indicators. For instance, we consider "this movie" as a pseudo entity of type <type2> and "like" as a pseudo entity of type <type14>. (2) The input of the pseudo extraction task is instruction="Target types:<type2>; <type14>"; the demonstrations="Text: *[MASK1] is cool and I really [MASK2] it [MASK3].* Entities: *[MASK1] is <type2>. [MASK2] is <type14>*" where the entities ("this movie" and "like") and other random spans ("very much") in demonstrations are masked. The text="Text: I do not like it." which is not masked. (3) The output of the pseudo extraction task is "*like is <type14>*" since the model will learn from demonstrations that <type14> corresponds to "like". (4) We also conduct traditional NER settings whose input is (instruction, text). The entities in the text will be masked as in demonstrations, e.g. "Target types: this movie; like Text: *I [MASK1] not [MASK2] it.*". The output will be "Entities: *[MASK2] is like.*".

We can see that the pseudo extraction LM task can benefit in-context NER in two ways. Firstly, it can significantly increase the size and diversity of in-context NER pre-training tasks from a large-scale unlabeled corpus. Secondly, this task pre-trains PLMs with a mixture of extraction target and span prediction task, therefore avoiding PLMs overfit to only extraction task.

When pre-training, We transformed the NER and language model tasks into a uniform format and sampled input instances alternately.

## 5 Experiments

This section evaluates our method by conducting experiments on few-shot NER settings.

### 5.1 Experimental Settings

**Pre-training settings.** Following Chen et al. (2022a), we build a large-scale distant NER dataset by aligning Wikipedia and Wikidata. Specifically, our dataset was made from Wikipedia text with hyperlinks to Wikidata, where we labeled entity types using the linked Wikidata item's attributes. Entity types were gathered from Wikidata's SubclassOf and InstanceOf attributes for each span. We filtered ambiguous and low-frequency types (occurrences <100k) to obtain higher-quality demonstrations. Finally, we retained 2046 types and 55 million (text, entities) pairs and use a 40/15 million split for training/validation. We sample 5 million in-context tasks for training and 10k for valida-

[4]We introduce how to select spans in Appendix.

| Models | #Param | CoNLL03 | | WNUT17 | | NCBI-disease | | SEC-filings | | AVE |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 1-shot | 5-shot | 1-shot | 5-shot | 1-shot | 5-shot | 1-shot | 5-shot | |
| **Pre-trained Language Models** | | | | | | | | | | |
| T5v1.1-large | 770M | 38.61 | 44.90 | 25.52 | 26.32 | 26.02 | 37.63 | 41.89 | 53.44 | 36.79 |
| GPT2-xl | 1.5B | 33.69 | 39.55 | 22.63 | 24.86 | 25.54 | 33.25 | 42.83 | **57.05** | 34.93 |
| T5-xl | 3B | 38.99 | 45.74 | 26.39 | 26.31 | 23.10 | 36.78 | 30.58 | 42.22 | 33.76 |
| GPT-J-6B | 6B | 46.14 | 50.10 | 31.41 | 30.93 | 35.82 | 40.98 | 40.12 | 39.61 | 39.39 |
| T5-xxl | 11B | 40.97 | 46.14 | 24.76 | 25.27 | 12.19 | 26.34 | 32.65 | 42.44 | 31.35 |
| OPT-13B | 13B | 46.65 | 51.71 | 27.74 | 28.36 | 23.73 | 34.00 | 41.60 | 43.10 | 37.11 |
| GPT-Neox-20B | 20B | 52.68 | 58.12 | **36.29** | 35.68 | 35.42 | 42.85 | 45.07 | 45.17 | 43.91 |
| OPT-30B | 30B | 42.86 | 44.77 | 25.85 | 27.44 | 22.31 | 32.76 | 40.83 | 46.52 | 35.42 |
| OPT-66B | 66B | 43.83 | 53.89 | 30.77 | 32.00 | 25.87 | 34.58 | 39.15 | 47.01 | 38.39 |
| **Pre-trained NER Models** | | | | | | | | | | |
| ProtoNet | 345M | 30.04 | 60.26 | 9.74 | 23.03 | 24.73 | 42.32 | 16.79 | 23.67 | 28.82 |
| NNShot | 345M | 41.92 | 58.39 | 15.76 | 21.78 | 31.59 | 33.14 | 30.19 | 37.86 | 33.83 |
| StructShot | 345M | 42.34 | 58.44 | 15.78 | 22.05 | 19.87 | 31.48 | 30.40 | 38.44 | 32.35 |
| CONTAINER | 345M | 45.43 | 61.69 | 15.64 | 20.37 | 23.24 | 27.02 | 34.07 | 40.44 | 33.49 |
| MetaNER-base | 220M | 53.94 | 62.59 | 25.55 | 30.41 | 35.00 | 37.24 | 46.88 | 51.39 | 42.88 |
| MetaNER | 770M | **57.40** | **63.45** | 31.59 | **36.52** | **40.01** | **44.92** | **52.07** | 54.87 | **47.60** |

Table 1: Micro-F1 scores of 1-shot and 5-shot in-context NER on test set. For a fair comparison, the results of each model are based on a single frozen model without fine-tuning and the pre-trained NER models are pre-trained using the same dataset as MetaNER.

tion, where each instance with type number $N$ is 10 and instance number $K$ is 10. We employ the T5-v1.1-large (Raffel et al., 2020) model as the initial model for MetaNER and further pre-train 500k steps with learning rate=5e-5 and warm-up steps=10k. In this paper, we refer to the pre-trained model as **MetaNER**.

**Few-shot settings.** Our experiments follow the standard k-shot NER setting Huang et al. (2021): For each entity type, we sample $k$ training instances as in-context demonstrations. We evaluate models by micro-F1 and report the average performance by repeating each experiment 10 times.

We conducts experiments on 4 datasets across differnt domains: (1) CoNLL03 (Sang and Meulder, 2003) from news domain. (2) WNUT17 (Derczynski et al., 2017) from social media domain. (3) NCBI-disease (Doğan et al., 2014) from biology domain. (4) SEC-filings (Alvarado et al., 2015) from finance domain.

**Baselines.** For fair comparison, we use frozen models for all baselines in the in-context learning experiments, i.e., a pre-trained language/NER model is used for entity extraction without fine-tuning. In addition, we will discuss fine-tuning based methods in section 5.3.3. Two kinds of baselines are compared:

1) **Pre-trained language models** include models with different scales and architectures: (1) Encoder-decoder models – T5 models (Raffel et al., 2020), includes T5-v1.1-large (770M), T5-xl (3B) and T5-xxl (11B). (2) Causal LM models – GPT and OPT models (Radford et al., 2019; Zhang et al., 2022b), includes GPT2-xl (1.5B), GPT-j-6B (Wang and Komatsuzaki, 2021), GPT-Neox-20B (Black et al., 2022), OPT-13B, OPT-30B and OPT-66B. Notice that, for PLMs, we use original type names rather than type indicators to capture the label semantics. For encoder-decoder models like T5, we formulate in-context NER as a span corruption task and the model will generate the extraction task. For example, for input "Target entity types: disease. Text: COVID-19 is spreading. Entities: COVID-19 is disease. Text: HIV is spread by three main routes. Entities: <extra_id_0>", the span corruption task requires the decoder to generate the extraction result "<extra_id_0> HIV is disease.".

2) **Pre-trained NER models** are metric-based few-shot methods, includes prototype network (ProtoNet) (Snell et al., 2017), NNshot (Yang and Katiyar, 2020), StructShot (Yang and Katiyar, 2020) and CONTAINER (Das et al., 2022). We employed BERT-Large (Devlin et al., 2019) as the backbone and pre-trained them using the same dataset as MetaNER. For a fair comparison, we

also pre-train a 220M T5-v1.1-base (Raffel et al., 2020) model with our meta-function pre-training algorithm (MetaNER-base).

### 5.2 Main Results

The experimental results are shown in Table 1. We can see that:

1) **Few-shot NER is challenging even for large language models, while MetaNER can achieve good in-context NER performance.** Compare with best-performed PLMs, MetaNER achieves 8.4% F1 improvements. Moreover, due to the gap between language model task and NER task, large language models achieve poor in-context learning performance on some datasets.

2) **Our in-context NER method can achieve robust performance, even under a large source-target domain gap.** Compared with best-performed metric-based NER models, MetaNER-base and MetaNER achieves 26.8% and 40.7% F1 improvement. Specifically, the performance improvement is more significant when source-target domain gap is larger, i.e., the NCBI-disease (biology domain) and SEC-filings (finance domain).

3) **Meta-function pre-training can effectively inject in-context learning ability into both small and large PLMs.** Both MetaNER-base and MetaNER achieve impressive performance in 1-shot and 5-shot settings, which verified that MetaNER can effectively inject in-context NER ability into small PLMs, although currently in-context learning has been seen an ability only emerged only on large language models such as GPT-3.

### 5.3 Detailed Analysis

#### 5.3.1 Ablation Studies

| | CoNLL03 | | | NCBI-disease | | |
|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 |
| MetaNER | 73.59 | 57.19 | **64.34** | **54.96** | **36.85** | **43.79** |
| w/o MF | 68.97 | 57.62 | 62.77 | 38.27 | 35.26 | 36.28 |
| w/o LM | 70.86 | **57.99** | 63.77 | 37.54 | 34.82 | 35.67 |
| w/o anonymization | **74.75** | 52.86 | 61.93 | 47.47 | 35.30 | 40.48 |

Table 2: Ablation studies on dev set. The results are based on 5-shot setting.

To analyze and understand the effect of type anonymization, meta-function pre-training, entity extraction pre-training, and pseudo extraction LM pre-training, we conduct the following ablation experiments: (1) MetaNER w/o MF: remove the meta-function pre-training; (2) MetaNER w/o LM: remove pseudo extraction LM pre-training; (3) MetaNER w/o anonymization: we use the original entity type names in both pre-training and in-context NER, without using type anonymization. The results are shown in Table 2, we can see that:

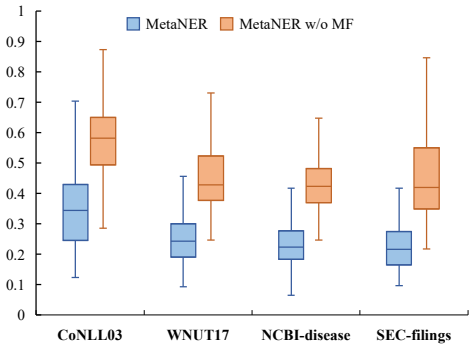


Figure 4: The visualization of feature comparison between meta-function $\mathcal{F}$ and the surrogate extractor $\mathcal{F}'$. The x-axis represents the different datasets, and the y-axis represents the distances between the features from the original encoder and the features from the surrogate encoder.

1) **meta-function pre-training is critical for in-context learning ability.** By removing the meta-function pre-training, the results drop significantly when the domain gaps are larger, i.e., NCBI-disease. At the same time, meta-function pre-training is helpful for the model to make more precise predictions.

2) **The pseudo extraction LM task significantly benefits in-context NER.** We found MetaNER w/o LM results in a performance drop than MetaNER. We believe this is because, although using an automatically constructed pseudo dataset, this task can significantly improve the size and the diversity of in-context NER tasks, meanwhile can retain a good language modeling ability.

3) **Type name anonymization prevents in-context NER model from type name overfitting, and therefore enhances the in-context learning ability.** The ablation of type name anonymization results a 5.7% performance drop in Table 2. We believe this is because type names will let models tend to memorize entity knowledge using type names, thus the model will not learn to capture entity knowledge from demonstrations on-the-fly.

#### 5.3.2 Effects of Meta-function Pre-training

One main idea of this paper is that in-context NER model can be viewed as a meta-function which can

implicitly build new entity extractors. To demonstrate whether meta-function pre-training can train a good meta-function, we sample 1000 instances from each dataset, and show the difference between the (instruction, demonstrations)-initialized entity extractor $\mathcal{F}$ and the surrogate entity extractor $\mathcal{F}'$, i.e., $\|\mathcal{F}' - \mathcal{F}\|$ in Section 4.2 in Figure 4. We can see that meta-function pre-training can equip PLMs with a good meta-function ability, i.e., the (instruction, demonstrations)-initialized entity extractor after pre-training is significantly close to its fine-tuned counterpart.

| | CoNLL03 | | WNUT17 | |
|---|---|---|---|---|
| | 1shot | 5shot | 1shot | 5shot |
| BERT-large (Devlin et al., 2019) | 14.66 | 52.43 | 8.95 | 32.77 |
| T5-v11-large (Raffel et al., 2020) | 11.65 | 42.13 | 12.51 | 39.54 |
| GPT-NEO-20B (Black et al., 2022)* | 52.68 | 58.12 | 36.29 | 35.68 |
| UIE-large (Lu et al., 2022b) | 46.28 | 67.62 | 32.86 | 42.67 |
| SDNet (Chen et al., 2022a) | / | 71.40 | / | 44.10 |
| CONTAINER-FT (Das et al., 2022) | 48.56 | 66.45 | 19.46 | 24.95 |
| MetaNER-ICL* | 57.40 | 63.45 | 31.59 | 36.52 |
| MetaNER-FT | **61.51** | **72.70** | **39.68** | **47.26** |

Table 3: The experiments of fine-tuning based methods. * indicates in-context learning settings. CONTAINER is pre-trained using the same NER dataset as MetaNER. All the models are implemented by us except SDNet.

#### 5.3.3 In-context Learning vs Fine-tuning

MetaNER can also be directly fine-tuned using traditional NER instances. We employed the identical fine-tuning approach as previous works (Huang et al., 2021; Lu et al., 2022b; Chen et al., 2022a). Following Lu et al. (2022b), we also implemented the *Rejection Mechanism* when fine-tuning the T5-v11-large and MetaNER to achieve better few-shot performance.

To compare in-context NER with fined-tuned NER, Table 3 reports the performance of the fine-tuned counterpart of MetaNER – MetaNER-FT(its training is similar to surrogate entity extractor but with multi-step gradient descent until coverage), together with several fine-tuned few-shot NER baselines. We can see that: 1) MetaNER is an effective architecture, which achieves good performance on both in-context learning and fine-tuning settings; 2) Currently, fine-tuning can achieve better performance than their in-context learning counterpart. We believe this is because fine-tuned models' parameters need to be specialized to specific entity types, meanwhile in-context learning needs to generalize to different types on-the-fly, i.e., generalization-specialization trade-off. We believe this also verified the reasonableness of using a fine-tuned surrogate extractor to approximate the golden extractor.

## 6 Conclusion

In this paper, we propose an in-context learning-based NER approach and model PLMs as a meta-function, which can inject in-context NER ability into PLMs and recognize entities of new types on-the-fly using only a few demonstrative instances. Experimental results show that our method is effective for in-context NER. For future work, we will extend our method to different NLP tasks like event extraction and relation extraction.

## Limitations

In-context learning is an useful ability, this paper only focuses on in-context named entity recognition, leaves the learning of other NLP tasks' in-context learning abilities for future work.

Currently, we learn in-context learning via meta-function pre-training, by comparing an in-context extraction function and a fined-tuned surrogate extraction function at the representation level of their encoders. There are two approximation here: one is fined-tuned surrogate extraction function for approximating golden extraction function, and the difference between representations for approximating the divergence between functions. We think the above two approximations can be further improved for better and faster in-context learning.

## Acknowledgements

We sincerely thank the reviewers for their insightful comments and valuable suggestions. This research work is supported by the CAS Project for Young Scientists in Basic Research under Grant No.YSBR-040 and the National Natural Science Foundation of China under Grants no. 62122077, 62106251.

## References

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. 2022. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*.

Julio Cesar Salinas Alvarado, Karin Verspoor, and Timothy Baldwin. 2015. Domain adaption of named entity recognition to support credit risk assessment. In *Proceedings of the Australasian Language Technology Association Workshop 2015*, pages 84–90.

Henk P Barendregt. 1992. Lambda calculi with types.

Ning Bian, Xianpei Han, Bo Chen, Hongyu Lin, Ben He, and Le Sun. 2021. Bridging the gap between language model and reading comprehension: Unsupervised mrc via self-supervision. *arXiv preprint arXiv:2107.08582*.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An open-source autoregressive language model. In *Proceedings of the ACL Workshop on Challenges & Perspectives in Creating Large Language Models*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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*, volume 33, pages 1877–1901. Curran Associates, Inc.

Stephanie CY Chan, Adam Santoro, Andrew K Lampinen, Jane X Wang, Aaditya Singh, Pierre H Richemond, Jay McClelland, and Felix Hill. 2022. Data distributional properties drive emergent few-shot learning in transformers. *arXiv preprint arXiv:2205.05055*.

Jiawei Chen, Qing Liu, Hongyu Lin, Xianpei Han, and Le Sun. 2022a. Few-shot named entity recognition with self-describing networks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5711–5722, Dublin, Ireland. Association for Computational Linguistics.

Mingda Chen, Jingfei Du, Ramakanth Pasunuru, Todor Mihaylov, Srini Iyer, Veselin Stoyanov, and Zornitsa Kozareva. 2022b. Improving in-context few-shot learning via self-supervised training. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3558–3573, Seattle, United States. Association for Computational Linguistics.

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

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Zhifang Sui, and Furu Wei. 2022. Why can gpt learn in-context? language models secretly perform gradient descent as meta optimizers. *arXiv preprint arXiv:2212.10559*.

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

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

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text, NUT@EMNLP 2017, Copenhagen, Denmark, September 7, 2017*, pages 140–147. Association for Computational Linguistics.

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

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.

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

Bernal Jiménez Gutiérrez, Nikolas McNeal, Clay Washington, You Chen, Lang Li, Huan Sun, and Yu Su. 2022. Thinking about gpt-3 in-context learning for biomedical ie? think again. *arXiv preprint arXiv:2203.08410*.

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.

Jiaxin Huang, Chunyuan Li, Krishan Subudhi, Damien Jose, Shobana Balakrishnan, Weizhu Chen, Baolin Peng, Jianfeng Gao, and Jiawei Han. 2021. Few-shot named entity recognition: An empirical baseline study. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10408–10423, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

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

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

Jing Li, Billy Chiu, Shanshan Feng, and Hao Wang. 2020a. Few-shot named entity recognition via meta-learning. *IEEE Transactions on Knowledge and Data Engineering*.

Jing Li, Shuo Shang, and Ling Shao. 2020b. Metaner: Named entity recognition with meta-learning. In *Proceedings of The Web Conference 2020*, pages 429–440.

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

Andy T Liu, Wei Xiao, Henghui Zhu, Dejiao Zhang, Shang-Wen Li, and Andrew Arnold. 2022. Qaner: Prompting question answering models for few-shot named entity recognition. *arXiv preprint arXiv:2203.01543*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022a. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022b. Unified structure generation for universal information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5755–5772, Dublin, Ireland. Association for Computational Linguistics.

Jie Ma, Miguel Ballesteros, Srikanth Doss, Rishita Anubhai, Sunil Mallya, Yaser Al-Onaizan, and Dan Roth. 2022a. Label semantics for few shot named entity recognition. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1956–1971, Dublin, Ireland. Association for Computational Linguistics.

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

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

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022a. MetaICL: Learning to learn in context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2791–2809, Seattle, United States. Association for Computational Linguistics.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022b. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. 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.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30.

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

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2022. Transformers learn in-context by gradient descent. *arXiv preprint arXiv:2212.07677*.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

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

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

Yaqing Wang, Subhabrata Mukherjee, Haoda Chu, Yuancheng Tu, Ming Wu, Jing Gao, and Ahmed Hassan Awadallah. 2021b. Meta self-training for few-shot neural sequence labeling. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1737–1747.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

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

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

Zeng Yang, Linhai Zhang, and Deyu Zhou. 2022. SEE-few: Seed, expand and entail for few-shot named entity recognition. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2540–2550, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Hongxin Zhang, Yanzhe Zhang, Ruiyi Zhang, and Diyi Yang. 2022a. Robustness of demonstration-based learning under limited data scenario. *arXiv preprint arXiv:2210.10693*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022b. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR.

# A Experiment Details

## A.1 Datasets for the extraction language model task

Rather than randomly generating spans to form target labels in instruction, we use informative spans (Bian et al., 2021) as target labels. Unlike informative span selection at passage level for MRC (Bian et al., 2021), we select informative spans at a cross-document level. Specifically, we take 10 Wikipedia documents as a set and select informative spans according to the following rules: (1) spans that have appeared simultaneously in at least two and at most five documents. (2) spans that have appeared in only one document but have appeared in more than two. Rule (1) avoids some

low-information general spans, such as stop words, and rule (2) retains some important spans in each document. Note that we consider at most 4-gram as a span and select the target labels from the informative spans during pre-training.

### A.2 Cost of pre-training

We used one A-100 80g GPU for pre-training the base/large model, which took approximately one to three days. The total FLOPs for the base model are 2.30e+18 and for the large model are 7.64e+18.

## ACL 2023 Responsible NLP Checklist

### A For every submission:

☑ A1. Did you describe the limitations of your work?
*In the 7-th Section*

☒ A2. Did you discuss any potential risks of your work?
*The data used for pre-training is based on publicly and widely used wikidata and wikipedia.*

☑ A3. Do the abstract and introduction summarize the paper's main claims?
*In abstract and the first section*

☒ A4. Have you used AI writing assistants when working on this paper?
*Left blank.*

### B ☒ Did you use or create scientific artifacts?

*Left blank.*

☐ B1. Did you cite the creators of artifacts you used?
*No response.*

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
*No response.*

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
*No response.*

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
*No response.*

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
*No response.*

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
*No response.*

### C ☑ Did you run computational experiments?

*In Section 5*

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
*In Section 5*

*The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.*

☐ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
*Not applicable. We conduct experiments in few-shot settings, which is unable to conduct hyperparameters and we use the hyperparameters as pervious works.*

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
*In Section 5*

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
*In Section 5*

**D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?**

*Left blank.*

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
*No response.*

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
*No response.*

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
*No response.*

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
*No response.*

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
*No response.*