Explore 3
strategies for few → 1. Meta learning: construct prototypes for different entities
shot NER
2. Supervised pre-training on web data to get generic representations
3. Self training

# Few-Shot Named Entity Recognition: A Comprehensive Study

**Jiaxin Huang**[1] [*], **Chunyuan Li**[2] [†], **Krishan Subudhi**[2], **Damien Jose**[2],
**Shobana Balakrishnan**[2], **Weizhu Chen**[2], **Baolin Peng**[2], **Jianfeng Gao**[2], **Jiawei Han**[1]
[1]University of Illinois Urbana-Champaign [2]Microsoft
{jiaxinh3, hanj}@illinois.edu
{chunyl,krkusuk,dajose,shobanab,wzchen,bapeng,jfgao}@microsoft.com

## Abstract

This paper presents a comprehensive study to efficiently build named entity recognition (NER) systems when a small number of in-domain labeled data is available. Based upon recent Transformer-based self-supervised pre-trained language models (PLMs), we investigate three orthogonal schemes to improve the model generalization ability for few-shot settings: (1) meta-learning to construct prototypes for different entity types, (2) supervised pre-training on noisy web data to extract entity-related generic representations and (3) self-training to leverage unlabeled in-domain data. Different combinations of these schemes are also considered. We perform extensive empirical comparisons on 10 public NER datasets with various proportions of labeled data, suggesting useful insights for future research. Our experiments show that (*i*) in the few-shot learning setting, the proposed NER schemes significantly improve or outperform the commonly used baseline, a PLM-based linear classifier fine-tuned on domain labels. (*ii*) We create new state-of-the-art results on both few-shot and training-free settings compared with existing methods. We will release our code and pre-trained models for reproducible research.

## 1 Introduction

Named Entity Recognition (NER) involves processing unstructured text, locating and classifying named entities (certain occurrences of words or expressions) into particular categories of pre-defined entity types, such as persons, organizations, locations, medical codes, dates and quantities. NER serves as an important first component for tasks such as information extraction (Ritter et al., 2012), information retrieval (Guo et al., 2009), question answering (Mollá et al., 2006), task-oriented dialogues (Peng et al., 2020a; Gao et al., 2019) and other language understanding applications (Nadeau and Sekine, 2007; Shaalan, 2014). Deep learning has shown remarkable success in NER in recent years, especially with self-supervised pre-trained language models (PLMs) such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019c). State-of-the-art (SoTA) NER models are often initialized with PLM weights, fine-tuned with standard supervised learning. One classical approach is to add a linear classifier on top of representations provided by PLMs, and fine-tune the entire model with a cross-entropy objective on domain labels (Devlin et al., 2019). Desipite its simplicity, the approach provides strong results on several benchmarks and is served as baseline in this study.

Unfortunately, even with these PLMs, building NER systems remains a labor-intensive, time-consuming task. It requires rich domain knowledge and expert experience to annotate a large corpus of in-domain labeled tokens to make the models work well. However, this is in contrast to the real-world application scenarios, where only very limited amounts of labeled data are available for new domains. For example, a new customer would prefer to provide very few labeled examples for specific domains in cloud-based NER services. The cost of building NER systems at scale with rich annotations (*i.e.*, hundreds of different enterprise use-cases/domains) can be prohibitively expensive. This draws attentions to a challenging but practical research problem: few-shot NER.

To deal with the challenge of few-shot learning, we focus on improving the generalization ability of PLMs for NER from three complementary directions, shown in Figure 1. Instead of limiting ourselves in making use of limited in-domain labeled tokens with the classical approach, (*i*) we create prototypes as the representations for differ-

[*] Work performed during an internship at Microsoft
[†] Corresponding author

Three orthogonal techniques

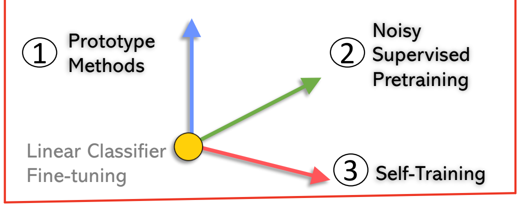


Figure 1: An overview of methods studied in our paper. Linear classifier fine-tuning is a default baseline that updates an NER model from pre-trained Roberta/BERT. We study three orthogonal strategies to improve NER models in the limited labeled data settings.

ent entity types, and assign labels via the nearest neighbor criterion; (*ii*) we continuously pre-train PLMs using web data with noisy labels that is available in much larger quantities to improve NER accuracy and robustness; (*iii*) we employ unlabeled in-domain tokens to predict their soft labels using self-training, and perform semi-supervised learning in conjunction with the limited labeled data.

Our contributions include: (*i*) We present the first systematic study for few-shot NER, a problem that is previously little explored in the literature. Three distinctive schemes and their combinations are investigated. (*ii*) We perform comprehensive comparisons of these schemes on 10 public NER datasets from different domains. (*iii*) Compared with existing methods on few-shot and training-free NER settings , the proposed schemes achieve SoTA performance despite their simplicity. To shed light on future research on few-shot NER, our study suggests that: (*i*) Noisy supervised pre-training can significantly improve NER accuracy, and we will release our pre-trained checkpoints. (*ii*) Self-training consistently improves few-shot learning when the ratio of data amounts between unlabeled and labeled data is high. (*iii*) The performance of prototype learning varies on different datasets. It is useful when the number of labeled examples is small, or when new entity types are given in the training-free settings.

Important findings

## 2 Background on Few-shot NER

**Few-shot NER.** a sequence labeling task, where the input is a text sequence (*e.g.,* sentence) of length $T$, $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T]$, and the output is a corresponding $T$-length labeling sequence $\mathbf{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_T]$, $\boldsymbol{y} \in \mathcal{Y}$ is a one-hot vector indicating the entity type of each token from a pre-defined discrete label space. The training dataset for NER often consists of pair-wise data $\mathcal{D}^{\mathrm{L}} = \{(\mathbf{X}_n, \mathbf{Y}_n)\}_{n=1}^{N}$, where $N$ is the number of training examples. Traditional NER systems are trained in the standard supervised learning paradigms, which usually requires a large number of pairwise examples, *i.e.,* $N$ is large. In real-world applications, the more favorable scenarios are that only a small number of labeled examples are given for each entity type ($N$ is small), because expanding labeled data increases annotation cost and decreases customer engagement. This yields a challenging task *few-shot NER*.

N is small

**Linear Classifier Fine-tuning.** Following the recent self-supervised PLMs (Devlin et al., 2019; Liu et al., 2019c), a typical method for NER is to utilize a Transformer-based backbone network to extract the contextualized representation of each token $\boldsymbol{z} = f_{\boldsymbol{\theta}_0}(\boldsymbol{x})$ . A linear classifier (*i.e.,* a linear layer with parameter $\boldsymbol{\theta}_1 = \{\mathbf{W}, \boldsymbol{b}\}$ followed by a Softmax layer) is applied to project the representation $\boldsymbol{z}$ into the label space $f_{\boldsymbol{\theta}_1}(\boldsymbol{z}) = \mathrm{Softmax}(\mathbf{W}\boldsymbol{z}+\boldsymbol{b})$. In another word, the end-to-end learning objective for linear classifier based NER can be obtained via a function composition $\boldsymbol{y} = f_{\boldsymbol{\theta}_1} \circ f_{\boldsymbol{\theta}_0}(\boldsymbol{x})$, with trainable parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_0, \boldsymbol{\theta}_1\}$. The pipeline is shown in Figure 2(a). The model is optimized by minimizing the cross-entropy:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{(\mathbf{X},\mathbf{Y})\in\mathcal{D}^{\mathrm{L}}} \sum_{i=1}^{T} \mathrm{KL}(\boldsymbol{y}_i || q(\boldsymbol{y}_i|\boldsymbol{x}_i)), \quad (1)$$

where the KL divergence between two distribution is $\mathrm{KL}(p||q) = \mathbb{E}_p \log(p/q)$, and the prediction probability vector for each token is

$$q(\boldsymbol{y}|\boldsymbol{x}) = \mathrm{Softmax}(\mathbf{W} \cdot f_{\boldsymbol{\theta}_0}(\boldsymbol{x}) + \boldsymbol{b}) \quad (2)$$

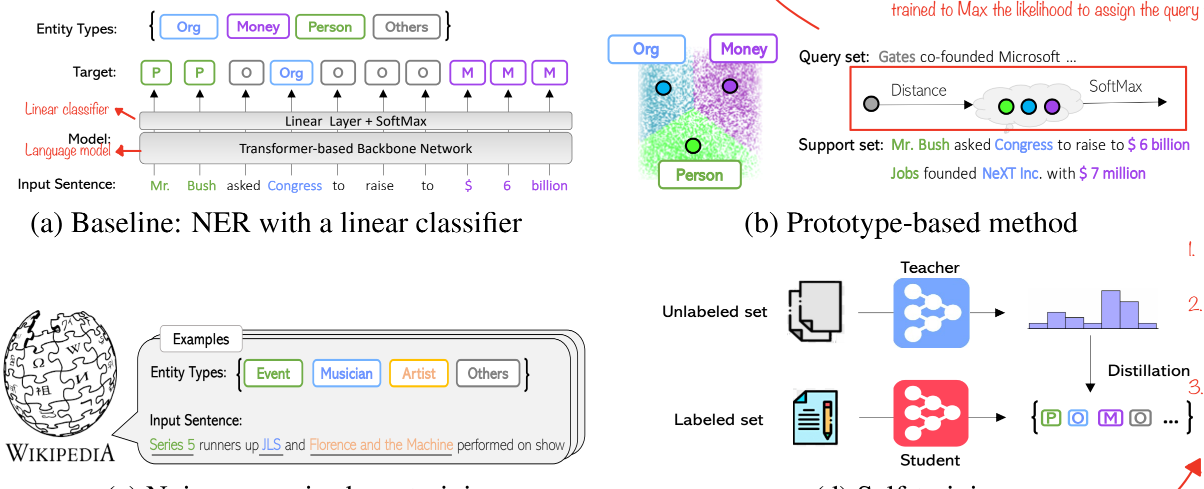
In practice, $\boldsymbol{\theta}_1 = \{\mathbf{W}, \boldsymbol{b}\}$ is always updated, while $\boldsymbol{\theta}_0$ can be either frozen (Liu et al., 2019a,b; Jie and Lu, 2019) or updated (Devlin et al., 2019; Yang and Katiyar, 2020).

## 3 Methods

When only a small number of labeled tokens are available, it renders difficulties for the classical supervised fine-tuning approach: the model tends to over-fit the training examples and shows poor generalization performance on the testing set (Fritzler et al., 2019). In this paper, we provide a comprehensive study specifically for limited NER data settings, and explore three orthogonal directions shown in Figure 1: (*i*) How to adapt meta-learning such as prototype-based methods for few-shot NER? (*ii*) How to leverage freely-available web data as noisy supervised pre-training data? (*iii*)

Small labelled data → Overfitting and poor generalisation

- For person the prototype is the average of The three Bush jobs and Gates
- For query, distance from different prototypes is computed. A model is trained to Max the likelihood to assign the query token to target



Entity Types:
Org
Money
Person
Others
Target:
P P O Org O O O M M M
Linear Layer + SoftMax
Model:
Transformer-based Backbone Network
Input Sentence:
Mr. Bush asked Congress to raise to $ 6 billion
Org
Money
Person
Query set: Gates co-founded Microsoft ...
Distance
SoftMax
Support set: Mr. Bush asked Congress to raise to $ 6 billion
Jobs founded NeXT Inc. with $ 7 million
WIKIPEDIA
Examples
Entity Types:
Event
Musician
Artist
Others
Input Sentence:
Series 5 runners up JLS and Florence and the Machine performed on show
Teacher
Unlabeled set
Distillation
Labeled set
Student
{P O M O ...}

Linear classifier

Language model

(a) Baseline: NER with a linear classifier

(b) Prototype-based method

1. Model is first trained on a small labelled set
2. Same model is then used to predict on a large unlabelled set
3. This joint, labelled plus predicted set is then used to train a student model

(c) Noisy supervised pre-training

(d) Self-training

Figure 2: Illustration of different methods for few-shot NER. In this example, each token in the input sentence is categorized into one of the four entity types. (a) A typical NER system, where a linear classifier is built on top of unsupervised pre-trained Transformer-based networks such as BERT/Roberta. (b) A prototype set is constructed via averaging features of all tokens belonging to a given entity type in the support set (*e.g.*, the prototype for `Person` is an average of three tokens: *Mr.*, *Bush* and *Jobs*). For a token in the query set, its distances from different prototypes are computed, and the model is trained to maximize the likelihood to assign the query token to its target prototype. (c) The Wikipedia dataset is employed for supervised pre-training, whose entity types are related but different (*e.g.*, `Musician` and `Artist` are more fine-grained types of `Person` in the downstream task). The associated types on each token can be noisy. (d) Self-training: An NER system (teacher model) trained on a small labeled dataset is used to predict soft labels for sentences in a large unlabeled dataset. The joint of the predicted dataset and original dataset is used to train a student model.

How to leverage unlabeled in-domain sentences in a semi-supervised manner? Note that these three directions are complementary to each other, can be further used jointly to extrapolate the methodology space in Figure 1.

### 3.1 Prototype-based Methods

Meta-learning (Ravi and Larochelle, 2017) have shown promising results for few-shot image classification (Tian et al., 2020) and sentence classification (Yu et al., 2018; Geng et al., 2019). It is natural to adapt this idea to few-shot NER. The core idea is to use episodic classification paradigm to simulate few-shot settings during model training. Specifically in each episode, $M$ entity types (usually $M < |\mathcal{Y}|$) are randomly sampled from $\mathcal{D}^{\mathrm{L}}$, containing a *support* set $\mathcal{S} = \{(\mathbf{X}_i, \mathbf{Y}_i)\}_{i=1}^{M\times K}$ ($K$ sentences per type) and a *query* set $\mathcal{Q} = \{(\hat{\mathbf{X}}_i, \hat{\mathbf{Y}}_i)\}_{i=1}^{M\times K'}$ ($K'$ sentences per type).

We build our method based on prototypical network (Snell et al., 2017), which introduces the notion of *prototypes*, representing entity types as vectors in the same representation space of individual tokens. To construct the prototype for the $m$-th entity type $\boldsymbol{c}_m$, the average of representations is computed for all tokens belonging to this type in the support set $\mathcal{S}$:

Average of contextual embeddings for all tokens of one entity type

$$\boldsymbol{c}_m = \frac{1}{|\mathcal{S}_m|}\sum_{\boldsymbol{x}\in\mathcal{S}_m} f_{\boldsymbol{\theta}_0}(\boldsymbol{x}), \qquad (3)$$

where $\mathcal{S}_m$ is the tokens set of the $m$-th type in $\mathcal{S}$, and $f_{\boldsymbol{\theta}_0}$ is defined in (2). For an input token $\boldsymbol{x} \in \mathcal{Q}$ from the query set, its prediction distribution is computed by a softmax function of the distance between $\boldsymbol{x}$ and all the entity prototypes. For example, the prediction probability for the $m$-th prototype is:

one hot vector

softmax

$$q(\boldsymbol{y} = \mathbb{I}_m|\boldsymbol{x}) = \frac{\exp\left(-d(f_{\boldsymbol{\theta}_0}(\boldsymbol{x}), \boldsymbol{c}_m)\right)}{\sum_{m'}\exp\left(-d(f_{\boldsymbol{\theta}_0}(\boldsymbol{x}), \boldsymbol{c}_{m'})\right)} \qquad (4)$$

Distance of the query token embedding with average token embedding for all entities

where $\mathbb{I}_m$ is the one-hot vector with 1 for $m$-th coordinate and 0 elsewhere, and $d(f_{\boldsymbol{\theta}_0}(\boldsymbol{x}), \boldsymbol{c}_m) = \|f_{\boldsymbol{\theta}_0}(\boldsymbol{x}) - \boldsymbol{c}_m\|_2$ is used in our implementation. We provide a simple example to illustrate the prototype method in Figure 2(b). In each training iteration, a new episode is sampled, and the model parameter $\boldsymbol{\theta}_0$ is updated via plugging (4) into (1). In testing phase, the label of a new token $\boldsymbol{x}$ is assigned using the nearest neighbor criterion $\arg\min_m d(f_{\boldsymbol{\theta}_0}(\boldsymbol{x}), \boldsymbol{c}_m)$.

### 3.2 Noisy Supervised Pre-training

Generic representations via self-supervised pre-trained language models (Devlin et al.,

New token →context embedding →label using

2019; Liu et al., 2019c) have benefited a wide range of NLP applications. These models are pre-trained with the task of randomly masked token prediction on massive corpora, and are agnostic to the downstream tasks. In other words, PLMs treat each token equally, which is not aligned with the goal of NER: identifying named entities as emphasized tokens and assigning labels to them. For example, for a sentence "*Mr. Bush asked Congress to raise to $ 6 billion*", PLMs treat *to* and *Congress* equally, while NER aims to highlight entities like *Congress* and lowlight their collocated non-entity words like *to*.

This intuition inspires us to endow the backbone network an ability to outweigh the representations of entities for NER. Hence, we propose to employ the large-scale noisy web data `WiNER` (Ghaddar, 2017) for noisy supervised pre-training (NSP). The labels in `WiNER` are automatically annotated on the 2013 English Wikipedia dump by querying anchored strings as well as their coreference mentions in each wiki page to the Freebase. The `WiNER` dataset is of 6.8GB and contains 113 entity types along with over 50 million sentences. Though introducing inevitable noises (*e.g.,* a random subset of 1000 mentions are manually evaluated and the accuracy of automatic annotations reaches 77%, due to the error of identifying coreferences), this automatic annotation procedure is highly scalable and affordable. The label set of `WiNER` covers a wide range of entity types. They are often related but different from entity types in the downstream datasets. For example in Figure 2(c), the entity types `Musician` and `Artist` in Wikipedia are more fine-grained than `Person` in a typical NER dataset. The proposed NSP learns representations to distinguish entities from others. This particularly favors the few-shot settings, preventing over-fitting via the prior knowledge of extracting entities from various contexts in pre-training.

Two pre-training objectives are considered in NSP, respectively: the first one is to use the linear classifier in (2), the other is a prototype-based objective in (4). For the linear classifier, we found that the batch size of 1024 and learning rate of $1e^{-4}$ works best, and for the prototype-based approach, we use the episodic training paradigm with $M = 5$ and set learning rate to be $5e^{-5}$. For both objectives, we train the whole corpus for 1 epoch and apply the Adam Optimizer (Kingma and Ba, 2015) with a linearly decaying schedule with warmup at 0.1. We empirically compare both objectives in experiments, and found that the linear classifier in (2) improves pre-training more significantly.

### 3.3 Self-training

Though manually labeling entities is expensive, it is easy to collect large amounts of unlabeled data in the target domain. Hence, it becomes desired to improve the model performance by effectively leveraging unlabeled data $\mathcal{D}^{\text{U}}$ with limited labeled data $\mathcal{D}^{\text{L}}$. We resort to the recent self-training scheme (Xie et al., 2020) for semi-supervised learning. The algorithm operates as follows:

1. Learn teacher model $\boldsymbol{\theta}^{\texttt{tea}}$ via cross-entropy using (1) with labeled tokens $\mathcal{D}^{\text{L}}$.

2. Generate soft labels using a teacher model on unlabeled tokens:
$$\tilde{\boldsymbol{y}}_i = f_{\boldsymbol{\theta}^{\texttt{tea}}}(\tilde{\boldsymbol{x}}_i), \forall \tilde{\boldsymbol{x}}_i \in \mathcal{D}^{\text{U}} \qquad (5)$$

3. Learn a student model $\boldsymbol{\theta}^{\texttt{stu}}$ via cross-entropy using (1) on labeled and unlabeled tokens:
$$\mathcal{L}_{\text{ST}} = \frac{1}{|\mathcal{D}^{\text{L}}|} \sum_{\boldsymbol{x}_i \in \mathcal{D}^{\text{L}}} \mathcal{L}(f_{\boldsymbol{\theta}^{\texttt{stu}}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$
Labeled data component
$$+ \frac{\lambda_{\text{U}}}{|\mathcal{D}^{\text{U}}|} \sum_{\tilde{\boldsymbol{x}}_i \in \mathcal{D}^{\text{U}}} \mathcal{L}(f_{\boldsymbol{\theta}^{\texttt{stu}}}(\tilde{\boldsymbol{x}}_i), \tilde{\boldsymbol{y}}_i) \qquad (6)$$
Unlabeled data component
where $\lambda_{\text{U}}$ is the weighting hyper-parameter.

A visual illustration for self-training procedure shown in Figure 2(d). It is optional to iterate from Step 1 to Step 3 multiple times, by initializing $\boldsymbol{\theta}^{\texttt{tea}}$ in Step 1 with newly learned $\boldsymbol{\theta}^{\texttt{stu}}$ in Step 3. We only perform self-training once in our experiments for simplicity, which has already shown excellent performance.

## 4 Related Work

**General NER.** NER is a long standing problem in NLP. Deep learning has significantly improve the recognition accuracy. Early efforts include exploring various neural architectures (Lample et al., 2016) such as Bidrectional LSTMs (Chiu and Nichols, 2016) and adding CRFs to capture structures (Ma and Hovy, 2016). Early studies have noticed the importance of reducing the annotation labor, where semi-supervised learning is employed, such as clustering (Lin and Wu, 2009), and combining supervised objective with unsupervised word representations (Turian et al., 2010). PLMs have recently revolutionized NER, where large-scale Transformer-based architectures (Peters et al., 2018; Devlin et al., 2019) are used as backbone network to extract informative representations. Contextualized string embedding (Akbik et al., 2018) is proposed to capture subword structures and polysemous words in different usage. Masked words and entities are jointly trained for prediction in (Yamada et al., 2020) with entity-aware self-attention. These methods are designed for standard supervised learning, and have a limited generalization ability in few-shot settings, as empirically shown in (Fritzler et al., 2019).

**Prototype-based methods** recently become popular few-shot learning approaches in machine learning community. It was firstly studied in the context of image classification (Vinyals et al., 2016; Sung et al., 2018; Zhao et al., 2020), and has recently been adapted to different NLP tasks such as text classification (Wang et al., 2018; Geng et al., 2019; Bansal et al., 2020), machine translation (Gu et al., 2018) and relation classification (Han et al., 2018). The closest related works to ours is (Fritzler et al., 2019) which explores prototypical network on few-shot NER, but only utilizes RNNs as the backbone model and does not leverage the power of large-scale Transformer-based architectures for word representations. Our work is similar to (Ziyadi et al., 2020; Wiseman and Stratos, 2019) in that all of them utilize the nearest neighbor criterion to assign the entity type, but differs in that (Ziyadi et al., 2020; Wiseman and Stratos, 2019) consider every individual token instance for nearest neighbor comparison, while ours considers prototypes for comparison. Hence, our method is much more scalable when the number of given examples increases.

**Supervised pre-training.** In computer vision, it is a *de facto* standard to transfer ImageNet-supervised pre-trained models to small image datasets to pursue high recognition accuracy (Yosinski et al., 2014). The recent work named big transfer (Kolesnikov et al., 2019) has achieved SoTA on various vision tasks via pre-training on billions of noisily labeled web images. To gain a stronger transfer learning ability, one may combine supervised and self-supervised methods (Li et al., 2020c,b). In NLP, supervised/grounded pre-training have been recently explored for natural language generation (NLG) (Keskar et al., 2019; Zellers et al., 2019; Peng et al., 2020b; Gao et al., 2020; Li et al., 2020a). They aim to endow GPT-2 (Radford et al.), an ability of enabling high-level semantic controlling in language generation, and are often pre-trained on massive corpus consisting of text sequences associated with prescribed codes such as text style, content description, and task-specific behavior. In contrast to NLG, to our best knowledge, large-scale supervised pre-training has been little studied for natural language understanding (NLU). There are early works showing promising results by transferring from medium-sized datasets to small datasets in some NLU applications; For example, from MNLI to RTE for sentence classification (Phang et al., 2018; Clark et al., 2020; An et al., 2020), and from OntoNER to CoNLL for NER (Yang and Katiyar, 2020). Our work further increases the supervised pre-training at the scale of web data (Ghaddar, 2017), 1000 orders of magnitude larger than (Yang and Katiyar, 2020), showing consistent improvements.

**Self-training.** Self-training (Scudder, 1965) is one of the earliest semi-supervised methods, and has recently achieved improved performance for tasks such as ImageNet classification (Xie et al., 2020), visual object detection (Zoph et al., 2020), neural machine translation (He et al., 2019), sentence classification (Mukherjee and Awadallah, 2020; Du et al., 2020). It is shown via object detection tasks in (Zoph et al., 2020) that stronger data augmentation and more labeled data can diminish the value of pre-training, while self-training is always helpful in both low-data and high-data regimes. Our work presents the first study of self-training for NER, and we observe similar phenomenons: it *consistently* boosts few-shot learning performance across all 10 datasets.

## 5 Experiments

In this section, we first compare the performance of different combinations of the three schemes on 10 benchmark datasets with various proportions of training data, and then compare our approaches with SoTA methods proposed for settings of few-shot learning and immediate inference for unseen entity types.

### 5.1 Settings

**Methods.** Throughout our experiments, the pre-trained base RoBERTa model is employed as the backbone network. We investigate the following 6 schemes for the comparative study: (*i*) **LC** is

| Datasets | CoNLL | Onto | WikiGold | WNUT | Movie | Restaurant | SNIPS | ATIS | Multiwoz | I2B2 |
|---|---|---|---|---|---|---|---|---|---|---|
| Domain | News | General | General | Social Media | Review | Review | Dialogue | Dialogue | Dialogue | Medical |
| #Train | 14.0k | 60.0k | 1.0k | 3.4k | 7.8k | 7.7k | 13.6k | 5.0k | 20.3k | 56.2k |
| #Test | 3.5k | 8.3k | 339 | 1.3k | 2.0k | 1.5k | 697 | 893 | 2.8k | 51.7k |
| #Entity Types | 4 | 18 | 4 | 6 | 12 | 8 | 53 | 79 | 14 | 23 |

Table 1: Statistics on the 10 public datasets studied in our NER benchmark.

the *linear classifier* fine-tuning method in Section 2, *i.e.*, adding a linear classifier on the backbone, and ① directly fine-tuning on entire model on the target dataset; (*ii*) **P** indicates the ② *prototype-based method* in Section 3.1; (*iii*) **NSP** refers to the *noisy* ③ *supervised pre-training* in Section 3.2; Depending on the pre-training objective, we have **LC+NSP** and **P+NSP**. (*iv*) ④ **ST** is the *self-training* approach in Section 3.3, it is combined with *linear classifier* fine-tuning, denoted as **LC+ST**; (*v*) **LC+NSP+ST**.

**Datasets.** We evaluate our methods on 10 public benchmark datasets, covering a wide range of domains: OntoNotes 5.0 (Ralph et al., 2013), WikiGold[1] (Balasuriya et al., 2009) on general domain, CoNLL 2003 (Sang and Meulder, 2003) on news domain, WNUT 2017 (Derczynski et al., 2017) on social domain, MIT Moive (Liu et al., 2013b) and MIT Restaurant[2] (Liu et al., 2013a) on review domain, SNIPS[3] (Coucke et al., 2018), ATIS[4] (Hakkani-Tür et al., 2016) and Multiwoz[5] (Budzianowski et al., 2018) on dialogue domain, and I2B2[6] (Stubbs and Uzuner, 2015) on medical domain. The detailed statistics of these datasets are summarized in Table 1.

For each dataset, we conduct three sets of experiments using various proportions of the training data: 5-shot, 10% and 100%. For 5-shot setting, we sample 5 sentences for each entity type in the training set and repeat each experiment for 10 times. For 10% setting, we down-sample 10 percent of the training set, and for 100% setting, we use the full training set as labeled data. We only study the self-training method in 5-shot and 10% settings, by using the rest of the training set as unlabeled in-domain corpus.

**Hyper-parameters.** We have described details for noisy supervised pre-training in Section 3.2. For training on target datasets, we set a fixed set of hyperparameters across all the datasets: For the linear classifier, we set batch size = 16 for 100% and 10% settings, batch size = 4 for 5-shot setting. For each episode in the prototype-based method, we set the number of sentences per entity type in support and query set $(K, K')$ to be $(5, 15)$ for 100% and 10% settings, and $(2, 3)$ for 5-shot setting. For both training objectives, we set learning rate = $5e^{-5}$ for 100% and 10% settings, and learning rate = $1e^{-4}$ for 5-shot setting. For all training data sizes, we set training epoch = 10, and Adam optimizer (Kingma and Ba, 2015) is used with the same linear decaying schedule as the pre-training stage. For self-training, we set $\lambda_{\text{U}} = 0.5$.

**Evaluation.** We follow the standard protocols for NER tasks to evaluate the performance on the test set (Sang and Meulder, 2003). Since RoBERTa tokenizes each word into subwords, we generate word-level predictions based on the first word piece of a word. Word-level predictions are then turned into entity-level predictions for evaluation when calculating the f1-score. Two tagging schemas are typically considered to encode chunks of tokens into entities: BIO schema marks the beginning token of an entity as `B-X` and the consecutive tokens as `I-X`, and other tokens are marked as `O`. IO schema uses `I-X` to mark all tokens inside an entity, thus is more defective as there is no boundary tag. In our study, we use BIO schema by default, but report the performance evaluated by IO schema for fair comparison with some previous studies.

## 5.2 Comprehensive Comparison Results

To gain thorough insights and benchmark few-shot NER, we first perform an extensive comparative study on 6 methods across 10 datasets. The results are shown in Table 2. We can draw the following major conclusions: (*i*) By comparing column ① and ② (or comparing ③ and ④), it clearly shows that noisy supervised pre-training provides better results in most datasets, especially in the 5-shot setting, which demonstrates

[1] https://github.com/juand-r/entity-recognition-datasets
[2] https://groups.csail.mit.edu/sls/downloads/
[3] https://github.com/snipsco/nlu-benchmark/tree/master/2017-06-custom-intent-engines
[4] https://github.com/yvchen/JointSLU
[5] https://github.com/budzianowski/multiwoz
[6] https://portal.dbmi.hms.harvard.edu/projects/n2c2-2014/

- Looks like the prototype method is not contributing much to performance improvement
- Self training → good gains
- Combination of LC NSP ST works best and complements the performance gains
- NSP seems to be more powerful than ST if the data is higher

| Datasets | Settings | ① | ② | ③ | ④ | ⑤ | ⑥ |
|---|---|---|---|---|---|---|---|
| | | LC | LC + NSP | P | P + NSP | LC + ST | LC + NSP + ST |
| CoNLL | 5-shot<br>10%<br>100% | 0.535<br>0.855<br>0.919 | 0.614<br>0.891<br>**0.920** | 0.584<br>0.878<br>0.911 | 0.609<br>0.888<br>0.915 | 0.567<br>0.878<br>- | **0.654**<br>**0.895**<br>- |
| Onto | 5-shot<br>10%<br>100% | 0.577<br>0.861<br>0.892 | 0.688<br>**0.869**<br>**0.899** | 0.533<br>0.854<br>0.886 | 0.570<br>0.846<br>0.883 | 0.605<br>0.867<br>- | **0.711**<br>0.867<br>- |
| WikiGold | 5-shot<br>10%<br>100% | 0.470<br>0.665<br>0.807 | 0.640<br>0.747<br>**0.839** | 0.511<br>0.692<br>0.801 | 0.604<br>0.701<br>0.827 | 0.481<br>0.695<br>- | **0.684**<br>**0.759**<br>- |
| WNUT17 | 5-shot<br>10%<br>100% | 0.257<br>0.483<br>0.489 | 0.342<br>0.492<br>0.520 | 0.295<br>0.485<br>0.552 | 0.359<br>0.478<br>**0.560** | 0.300<br>0.490<br>- | **0.376**<br>**0.505**<br>- |
| MIT Movie | 5-shot<br>10%<br>100% | 0.513<br>0.651<br>**0.693** | 0.531<br>0.657<br>0.692 | 0.380<br>0.563<br>0.632 | 0.438<br>0.583<br>0.641 | 0.541<br>0.659<br>- | **0.559**<br>**0.666**<br>- |
| MIT Restaurant | 5-shot<br>10%<br>100% | 0.487<br>0.745<br>0.790 | 0.491<br>0.734<br>**0.793** | 0.441<br>0.713<br>0.787 | 0.484<br>0.721<br>0.791 | 0.503<br>**0.750**<br>- | **0.513**<br>0.741<br>- |
| SNIPS | 5-shot<br>10%<br>100% | 0.792<br>0.945<br>0.970 | 0.824<br>**0.950**<br>**0.972** | 0.750<br>0.879<br>0.923 | 0.773<br>0.896<br>0.956 | 0.796<br>0.946<br>- | **0.830**<br>0.942<br>- |
| ATIS | 5-shot<br>10%<br>100% | **0.908**<br>0.883<br>0.953 | **0.908**<br>0.898<br>**0.956** | 0.842<br>0.785<br>0.929 | 0.896<br>0.896<br>0.943 | 0.904<br>0.898<br>- | 0.905<br>**0.903**<br>- |
| Multiwoz | 5-shot<br>10%<br>100% | 0.123<br>0.826<br>0.880 | 0.198<br>0.830<br>**0.885** | 0.219<br>0.787<br>0.837 | **0.451**<br>0.805<br>0.845 | 0.200<br>0.835<br>- | 0.225<br>**0.841**<br>- |
| I2B2 | 5-shot<br>10%<br>100% | 0.360<br>0.855<br>0.932 | 0.385<br>0.869<br>**0.935** | 0.320<br>0.703<br>0.895 | 0.366<br>0.762<br>0.906 | 0.365<br>0.865<br>- | **0.393**<br>**0.871**<br>- |
| **Average** | 5-shot<br>10%<br>100% | 0.502<br>0.777<br>0.833 | 0.562<br>0.794<br>**0.841** | 0.488<br>0.734<br>0.815 | 0.555<br>0.758<br>0.827 | 0.526<br>0.788<br>- | **0.585**<br>**0.799**<br>- |

Table 2: F1-score on benchmark datasets with various sizes of training data. LC is *linear classifier* fine-tuning method, P is *prototype-based training* using a nearest neighbor objective, NSP is *noising supervised pre-training* and ST is *self-training*. The best results are in **bold**.

that **NSP** endows the model an ability to extract better NER-related features. (*ii*) The comparison between column ① and ③ provides a head-to-head comparison between linear classifier and prototype-based methods: while the prototype-based method demonstrates better performance than **LC** on CoNLL, WikiGold, WNUT17 and Multiwoz in the 5-shot learning setting, it falls behind **LC** on other datasets and in average statistics. It shows that the prototype-based method only yields better results when there is very limited labeled data: the size of both entity types and examples are small. (*iii*) When comparing column ⑤ with ① (or comparing column ⑥ and ②), we observe that using self-training consistently works better than directly fine-tuning with labeled data only, suggesting that **ST** is a useful technique to leverage in-domain unlabeled data if allowed. (*iv*) Column ⑥ shows the highest F1-score in most cases, demonstrating the three proposed schemes in this paper are complementary to each other, and can be combined to yield best results in practice.

### 5.3 Comparison with SoTA Methods

**Competitive methods.** The current SoTA on few-shot NER includes: (*i*) *StructShot* (Yang and Katiyar, 2020), which extends the nearest neighbor classification with a decoding process using abstract tag transition distribution. Both the model and the transition distribution are trained from the source dataset OntoNotes. (*ii*) *L-TapNet+CDT* (Hou et al., 2020) is a slot tagging

| Schema | Methods | CoNLL | I2B2 | WNUT | Average |
|---|---|---|---|---|---|
| IO | SimBERT † | 0.286±0.025 | 0.091±0.007 | 0.077±0.022 | 0.151 |
| | L-TapNet+CDT † | 0.671±0.016 | 0.101±0.009 | 0.238±0.039 | 0.336 |
| | StructShot † | 0.752±0.023 | 0.318±0.018 | 0.272±0.067 | 0.447 |
| | P + NSP | 0.757±0.021 | 0.322±0.033 | 0.442±0.024 | 0.507 |
| | LC + NSP | 0.771±0.035 | 0.371±0.035 | 0.417±0.022 | **0.520** |
| | LC + NSP + ST | 0.779±0.040 | 0.376±0.028 | 0.419±0.028 | **0.525** |
| BIO | P + NSP | 0.756±0.017 | 0.334±0.024 | 0.424±0.012 | 0.505 |
| | LC + NSP | 0.712±0.048 | 0.364±0.032 | 0.403±0.029 | 0.493 |
| | LC + NSP + ST | 0.722±0.011 | 0.369±0.021 | 0.409±0.013 | 0.500 |

Table 3: Comparison of F1-score with SoTA on 5-shot NER tasks. Results of both BIO and IO schemas are reported for fair comparison. The best results are in **bold**. † indicates results from (Yang and Katiyar, 2020).

| Datasets | Methods | *Number of support examples per entity type* | | | | | |
|---|---|---|---|---|---|---|---|
| | | *10* | *20* | *50* | *100* | *200* | *500* |
| ATIS | Neigh.Tag.† | 0.067±0.008 | 0.088±0.007 | 0.111±0.007 | 0.143±0.006 | 0.221±0.006 | 0.339±0.006 |
| | Example† | 0.174±0.011 | 0.198±0.012 | 0.222±0.011 | 0.268±0.027 | 0.345±0.022 | 0.401±0.010 |
| | Prototype | 0.381±0.021 | 0.391±0.022 | 0.376±0.008 | 0.379±0.005 | 0.377±0.006 | 0.376±0.003 |
| | Prototype + NSP | 0.684±0.013 | 0.712±0.014 | 0.716±0.013 | 0.705±0.010 | 0.705±0.006 | 0.708±0.002 |
| | Multi-Prototype | 0.339±0.016 | 0.362±0.018 | 0.366±0.005 | 0.373±0.004 | 0.371±0.005 | 0.372±0.003 |
| | Multi-Prototype + NSP | **0.712**±0.014 | **0.748**±0.011 | **0.760**±0.008 | **0.742**±0.005 | **0.743**±0.003 | **0.746**±0.002 |
| MIT.Restaurant | Neigh.Tag.† | 0.042±0.018 | 0.038±0.008 | 0.037±0.007 | 0.046±0.008 | 0.055±0.011 | 0.081±0.006 |
| | Example.† | 0.276±0.018 | 0.295±0.010 | 0.312±0.007 | 0.337±0.005 | 0.345±0.004 | 0.346 |
| | Prototype | 0.330±0.013 | 0.332±0.013 | 0.332±0.010 | 0.329±0.003 | 0.329±0.004 | 0.331±0.003 |
| | Prototype + NSP | 0.455±0.016 | 0.455±0.012 | 0.455±0.013 | 0.438±0.013 | 0.437±0.008 | 0.438±0.006 |
| | Multi-Prototype | 0.345±0.012 | 0.360±0.015 | 0.371±0.012 | 0.376±0.009 | 0.385±0.005 | 0.386±0.004 |
| | Multi-Prototype + NSP | **0.461**±0.019 | **0.482**±0.011 | **0.496**±0.008 | **0.496**±0.011 | **0.500**±0.005 | **0.501**±0.003 |
| MIT Movie | Neigh.Tag.† | 0.031±0.020 | 0.045±0.019 | 0.041±0.011 | 0.053±0.009 | 0.054±0.007 | 0.086±0.008 |
| | Example.† | **0.401**±0.011 | **0.395**±0.007 | **0.402**±0.007 | **0.400**±0.004 | **0.400**±0.005 | **0.395**±0.007 |
| | Prototype | 0.175±0.007 | 0.168±0.006 | 0.170±0.004 | 0.174±0.003 | 0.173±0.002 | 0.173±0.002 |
| | Prototype + NSP | 0.303±0.011 | 0.293±0.007 | 0.285±0.006 | 0.284±0.002 | 0.282±0.002 | 0.280±0.002 |
| | Multi-Prototype | 0.197±0.007 | 0.207±0.005 | 0.219±0.004 | 0.227±0.002 | 0.229±0.003 | 0.230±0.002 |
| | Multi-Prototype + NSP | 0.364±0.020 | 0.368±0.011 | 0.380±0.006 | 0.382±0.003 | 0.354±0.003 | 0.383±0.002 |

Table 4: F1-score on training-free settings, *i.e.,* predicting novel entity types using nearest neighbor methods. The best results are in **bold**. † indicates results from (Ziyadi et al., 2020; Wiseman and Stratos, 2019).

method which constructs an embedding projection space using label name semantics to well separate different classes. It also includes a collapsed dependency transfer mechanism to transfer label dependency information from source domains to target domains. (*iii*) *SimBERT* is a simple baseline reported in (Yang and Katiyar, 2020; Hou et al., 2020); it utilizes a nearest neighbor classifier based on the contextualized representation output by the pre-trained BERT, without fine-tuning on few-shot examples. The results reported in the StructShot paper use IO schema instead of BIO schema, thus we report our performance on both for completeness.

For fair comparison, following (Yang and Katiyar, 2020), we also continuously pre-train our model on OntoNotes after the noisy supervised pre-training stage. For each 5-shot learning task, we repeat the experiments 10 times by re-sampling few-shot examples each time. The results are reported in Table 3. We observe that our proposed methods consistently outperform the StructShot model across all three datasets, even by simply pre-training the model on large-scale noisily tagged datasets like Wikipedia. Our best model outperforms the previous SoTA by 8% F1-score, which demonstrates that using large amounts of unlabeled in-domain corpus is promising for enhancing the few-shot NER performance.

### 5.4 Training-free Method Comparison

Some real-world applications require immediate inference on unseen entity types. For example, novel entity types with a few examples are frequently given in an online fashion, but updating model weights $\boldsymbol{\theta}$ frequently is prohibitive. One may store some token examples as supports and utilize them for nearest neighbor classification. The setting is referred to as *training-free* in (Wiseman and Stratos, 2019; Ziyadi et al., 2020), as the models identify new entities in a completely unseen target domain using only a few supporting examples in this new domain, without updating $\boldsymbol{\theta}$ in that target domain. Our prototype-based method is able to perform such immediate inference. Two recent works on training-free NER are: $(i)$ *Neighbor-tagging* (Wiseman and Stratos, 2019) copies token-level labels from weighted nearest neighbors; $(ii)$ *Example-based NER* (Ziyadi et al., 2020) is the SoTA on training-free NER, which identifies the starting and ending tokens of unseen entity types.

We observed that our basic prototype-based method, under the training-free setting, does not gain from more given examples. We hypothesize that this is because tokens belonging to the same entity type are not necessarily close to each other, and are often separated in the representation space. Though it is hard to find one single centroid for all tokens in the same type, we assume that there exist local clusters of tokens belonging to the same type. To resolve such issue, we follow (Deng et al., 2020) and extend our method to a version called *Multi-Prototype*, by creating $K/5$ prototypes for each type given $K$ examples per type. (*e.g.,* 2 prototypes per class are used for the 10-shot setting). The prediction score for a testing token belonging to a type is computed via averaging the prediction probabilities from all prototypes of the same type.

We compare with previous methods in Table 4. We observe that multi-prototype methods not only benefit from more support examples, but also surpass neighbor tagging methods and example-based NER by a large margin on two out of three datasets. For the MIT Movie dataset, one entity type can span a large chunk with multiple consecutive words in a sentence, which favors the span-based method like (Ziyadi et al., 2020). For example, the underlined part in the sentence "*what movie does the quote i dont think we are in kansas anymore come from*" is annotated as entity type `Quote`. The proposed methods in this paper can be combined with the span-based approach to specifically tackle this problem, and we leave it as future work. Further, if slightly fine-tuning is allowed, we see that the prototype-based method achieves 0.438 with 5-shot learning in Table 2, better than 0.395 achieved by example-based NER given 500 examples.

## 6 Conclusion

We have presented a comprehensive study on few-shot NER. Three foundational methods and their combinations are systematically investigated: prototype-based methods, noisy supervised pre-training and self-training. They are intensively compared on 10 public datasets under various settings. All of them can improve the PLM's generalization ability when learning from a few labeled examples, among which supervised pre-training and self-training turn out to be particularly effective. The proposed schemes achieve SoTA on both few-shot and training-free settings compared with recently proposed methods. We will release our benchmarks and code for few-shot NER, and hope that it can inspire future research with more advanced methods to tackle this challenging and practical problem.

## References

A. Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING*.

Bang An, Jie Lyu, Zhenyi Wang, Chunyuan Li, Changwei Hu, Fei Tan, Ruiyi Zhang, Yifan Hu, and Changyou Chen. 2020. Repulsive attention: Rethinking multi-head attention as bayesian inference. *arXiv preprint arXiv:2009.09364*.

Dominic Balasuriya, Nicky Ringland, Joel Nothman, Tara Murphy, and J. Curran. 2009. Named entity recognition in wikipedia. In *PWNLP@IJCNLP*.

Trapit Bansal, Rishikesh Jha, Tsendsuren Munkhdalai, and Andrew McCallum. 2020. Self-supervised meta-learning for few-shot natural language classification tasks. *arXiv preprint arXiv:2009.08445*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Ultes Stefan, Ramadan Osman, and Milica Gašić. 2018. Multiwoz - a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

A. Coucke, A. Saade, Adrien Ball, Théodore Bluche, A. Caulier, D. Leroy, Clément Doumouro, Thibault Gisselbrecht, F. Caltagirone, Thibaut Lavril, Maël Primet, and J. Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *ArXiv*, abs/1805.10190.

Jiankang Deng, Jia Guo, Tongliang Liu, Mingming Gong, and Stefanos Zafeiriou. 2020. Sub-center arcface: Boosting face recognition by large-scale noisy web faces. In *European Conference on Computer Vision*.

Leon Derczynski, Eric Nichols, M. Erp, and Nut Limsopatham. 2017. Results of the wnut2017 shared task on novel and emerging entity recognition. In *NUT@EMNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL*.

Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Ves Stoyanov, and Alexis Conneau. 2020. Self-training improves pre-training for natural language understanding. *arXiv preprint arXiv:2010.02194*.

Alexander Fritzler, V. Logacheva, and M. Kretov. 2019. Few-shot classification in named entity recognition task. *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*.

Jianfeng Gao, Michel Galley, and Lihong Li. 2019. Neural approaches to conversational ai. *Foundations and Trends® in Information Retrieval*, 13(2-3):127–298.

Jianfeng Gao, Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Heung-Yeung Shum. 2020. Robust conversational ai with grounded text generation. *arXiv preprint arXiv:2009.03457*.

Ruiying Geng, Binhua Li, Yongbin Li, Xiao-Dan Zhu, Ping Jian, and Jian Sun. 2019. Induction networks for few-shot text classification. In *EMNLP/IJCNLP*.

Abbas Ghaddar. 2017. Winer: A wikipedia annotated corpus for named entity recognition. In *IJCNLP*.

Jiatao Gu, Yong Wang, Yun Chen, Kyunghyun Cho, and V. Li. 2018. Meta-learning for low-resource neural machine translation. *ArXiv*, abs/1808.08437.

J. Guo, G. Xu, X. Cheng, and Hang Li. 2009. Named entity recognition in query. In *SIGIR*.

Dilek Z. Hakkani-Tür, G. Tür, A. Çelikyilmaz, Yun-Nung Chen, Jianfeng Gao, L. Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *INTERSPEECH*.

Xu Han, Hao Zhu, Pengfei Yu, Z. Wang, Y. Yao, Zhiyuan Liu, and M. Sun. 2018. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. *ArXiv*, abs/1810.10147.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2019. Revisiting self-training for neural sequence generation. *arXiv preprint arXiv:1909.13788*.

Yutai Hou, W. Che, Y. Lai, Zhihan Zhou, Yijia Liu, H. Liu, and T. Liu. 2020. Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network. In *ACL*.

Zhanming Jie and Wei Lu. 2019. Dependency-guided lstm-crf for named entity recognition. *ArXiv*, abs/1909.10148.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. 2019. Big transfer (bit): General visual representation learning. *arXiv preprint arXiv:1912.11370*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Chunyuan Li, Xiang Gao, Yuan Li, Xiujun Li, Baolin Peng, Yizhe Zhang, and Jianfeng Gao. 2020a. Optimus: Organizing sentences via pre-trained modeling of a latent space. *arXiv preprint arXiv:2004.04092*.

Chunyuan Li, Xiujun Li, Lei Zhang, Baolin Peng, Mingyuan Zhou, and Jianfeng Gao. 2020b. Self-supervised pre-training with hard examples improves visual representations. *arXiv preprint arXiv:2012.13493*.

Junnan Li, Caiming Xiong, and Steven CH Hoi. 2020c. Mopro: Webly supervised learning with momentum prototypes. *arXiv preprint arXiv:2009.07995*.

Dekang Lin and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1030–1038.

J. Liu, Panupong Pasupat, D. Cyphers, and James R. Glass. 2013a. Asgard: A portable architecture for multilingual dialogue systems. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8386–8390.

J. Liu, Panupong Pasupat, Y. Wang, D. Cyphers, and James R. Glass. 2013b. Query understanding enhanced by hierarchical parsing structures. *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 72–77.

Tianyu Liu, J. Yao, and Chin-Yew Lin. 2019a. Towards improving neural named entity recognition with gazetteers. In *ACL*.

Yanjun Liu, Fandong Meng, Jinchao Zhang, Jinan Xu, Yufeng Chen, and Jie Zhou. 2019b. Gcdt: A global context enhanced deep transition architecture for sequence labeling. In *ACL*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019c. Roberta: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Diego Mollá, Menno Van Zaanen, Daniel Smith, et al. 2006. Named entity recognition for question answering.

Subhabrata Mukherjee and Ahmed Hassan Awadallah. 2020. Uncertainty-aware self-training for text classification with few labels. *arXiv preprint arXiv:2006.15315*.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2020a. Soloist: Few-shot task-oriented dialog with a single pretrained auto-regressive model. *arXiv preprint arXiv:2005.05298*.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020b. Few-shot natural language generation for task-oriented dialog. *arXiv preprint arXiv:2002.12328*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners.

Weischedel Ralph, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, and Nianwen Xue et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium*.

S. Ravi and H. Larochelle. 2017. Optimization as a model for few-shot learning. In *ICLR*.

Alan Ritter, Mausam, Oren Etzioni, and S. Clark. 2012. Open domain event extraction from twitter. In *KDD*.

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

H Scudder. 1965. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*.

Khaled Shaalan. 2014. A survey of arabic named entity recognition and classification. *Computational Linguistics*.

J. Snell, Kevin Swersky, and R. Zemel. 2017. Prototypical networks for few-shot learning. In *NIPS*.

A. Stubbs and Özlem Uzuner. 2015. Annotating longitudinal clinical narratives for de-identification: The 2014 i2b2/uthealth corpus. *Journal of biomedical informatics*, 58 Suppl:S20–9.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Yonglong Tian, Yue Wang, Dilip Krishnan, J. Tenenbaum, and Phillip Isola. 2020. Rethinking few-shot image classification: a good embedding is all you need? *ArXiv*, abs/2003.11539.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174*.

Sam Wiseman and K. Stratos. 2019. Label-agnostic sequence labeling by copying nearest neighbors. *ArXiv*, abs/1906.04225.

Qizhe Xie, E. Hovy, Minh-Thang Luong, and Quoc V. Le. 2020. Self-training with noisy student improves imagenet classification. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, H. Takeda, and Yuji Matsumoto. 2020. Luke: Deep contextualized entity representations with entity-aware self-attention. In *EMNLP*.

Y. Yang and Arzoo Katiyar. 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. *ArXiv*, abs/2010.02405.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, S. Chang, Saloni Potdar, Yu Cheng, G. Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. In *NAACL-HLT*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Advances in Neural Information Processing Systems*.

Yang Zhao, Chunyuan Li, Ping Yu, and Changyou Chen. 2020. Remp: Rectified metric propagation for few-shot learning. *arXiv preprint arXiv:2012.00904*.

M. Ziyadi, Yuting Sun, A. Goswami, Jade Huang, and W. Chen. 2020. Example-based named entity recognition. *ArXiv*, abs/2008.10570.

Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin D Cubuk, and Quoc V Le. 2020. Rethinking pre-training and self-training. *arXiv preprint arXiv:2006.06882*.