Toward Recognizing More Entity Types in NER: An Efficient Implementation using Only Entity Lexicons

Minlong Peng[1], **Ruotian Ma**[1], **Qi Zhang**[1], **Lujun Zhao**[2],
Mengxi Wei[2], **Changlong Sun**[2], **Xuanjing Huang**[1]
School of Computer Science, Shanghai Key Laboratory of Intelligent Information Processing,
Fudan University, Shanghai, China[1]
Alibaba Group, Hangzhou, China[2]
{mlpeng16,rtma19,qz,xjhuang}@fudan.edu.cn[1]
{lujun.zlj,cangqing.wmx}@alibaba-inc.com[2], changlong.scl@taobao.com[2]

Abstract

In this work, we explore the way to quickly adjust an existing named entity recognition (NER) system to make it capable of recognizing entity types not defined in the system. As an illustrative example, consider the case that a NER system has been built to recognize person and organization names, and now it requires to additionally recognize job titles. Such a situation is common in the industrial areas, where the entity types required to recognize vary a lot in different products and keep changing. To avoid laborious data labeling and achieve fast adaptation, we propose to adjust the existing NER system using the previously labeled data and entity lexicons of the newly introduced entity types. We formulate such a task as a partially supervised learning problem and accordingly propose an effective algorithm to solve the problem. Comprehensive experimental studies on several public NER datasets validate the effectiveness of our method.

1 Introduction

Named Entity Recognition (NER) is a type of information extraction task that seeks to identify entity names from unstructured text and categorize them into a predefined list of types. It plays an important role in many downstream tasks such as knowledge base construction (Riedel et al., 2013; Shen et al., 2012), machine translation (Babych and Hartley, 2003), and search (Zhu et al., 2005), etc. In this field, the supervised methods, ranging from the conventional graph models (McCallum et al., 2000; Malouf, 2002; McCallum and Li, 2003; Settles, 2004) to the dominant deep neural methods (Collobert et al., 2011; Huang et al., 2015; Lample et al., 2016; Gridach, 2017; Liu et al., 2018; Zhang and Yang, 2018; Jiang et al., 2019; Gui et al., 2019), have achieved great success. However, these supervised methods usually require large scale labeled data to achieve good performance, while the annotation of NER data is often laborious and time-consuming.

In the real world, there are many, or more strictly speaking, infinite numbers of entity types. It is impossible for a NER system to cover all entity types (Ling and Weld, 2012; Mai et al., 2018). Therefore, in the industrial area, it often happens that some entity types required to recognize by the clients are not defined in the previously designed NER system. In such a case, we need to quickly adjust the existing NER system to make it capable of recognizing the new entity types required by the clients. In this literature, we refer to the existing NER system as the *source system*, and refer to the adjusted system as the *target system*. The NER tasks defined in the two systems are referred to as the *source task* and the *target task*, respectively. The goal of this work is quickly transferring from the source task to the target task.

Suppose the new entity types defined in the target task are classified into class K (e.g, GPE and non-GPE are all annotated as the location type) in the source task. A common practice to build the target system is sampling some examples from the training data of the source task and asking the annotators to re-annotate words of class K in these examples. Then, it finetunes the model pretrained on the source task (with the output layer being replaced) using the re-annotated data to perform the target task. However, it is worth noting that the NER labels of words are context-dependent. To re-annotate the words of class K, the annotators need to read the whole sentence rather than the fragmental words of class K. This is still laborious and time-consuming, making it not an ideal choice when fast adaptation is required or the required entity types by the clients vary a lot and keep changing.

In this work, we propose to transfer from the

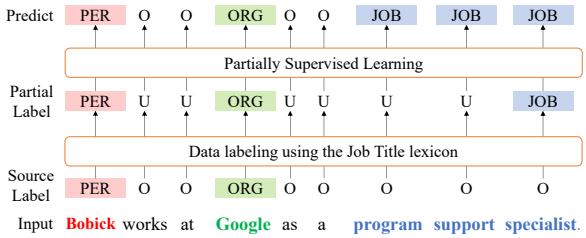

Figure 1: Applying of our method to an illustrative sentence for additionally introducing *Job Title* in the target task. Here, words in blue color constitute a job title. "Partial Label" corresponds to the automatically obtained partial labels using the job title lexicon, where "U" means the label of the word is unknown (can be "O" or "JOB"). "Predict" denotes the expected labels predicted by our model.

source task to the target task using only the labeled data of the source task and entity lexicons of the newly introduced entity types. Note that the collection of entity lexicons is often much easier than data annotation. For example, we can ask the language experts familiar with NER to provide some common mentions of the new entity types, or we can collect some confident mentions of the types from the internet to construct the lexicons. In some cases, we can even ask the clients of the target system to provide the lexicons, and usually, they are more willing to do so than annotate data.

To perform the transfer task using the entity lexicons, we formulate the task as a partially supervised learning problem. Figure 1 depicts the general process of our method, where the target task needs to additionally recognize job titles, which are annotated as the Other (O) class in the source task. Specifically, for the job title type, an entity lexicon of the type is collected. The lexicon is used to automatically re-annotate the training words of the *Other* class in the source task ("O" of the "Source Label" in Figure 1), obtaining some labeled data of the new entity type ("JOB" of the "Partial Label" in Figure 1). The rest words of the *Other* class of the source task not being annotated by the lexicons form the unlabeled data ("U" of the "Partial Label" in Figure 1). Note that, the unlabeled data contains both words of the new entity types and words not belonging to any entity type in the target task, and there is no labeled data for the *Other* class in the target task. Based on the obtained labeled and unlabeled data, a multi-class classifier is trained to perform the target task using a partially supervised (PS) learning algorithm. In this classifier, the constituted words of an entity

$\mathbb{T}_s$, $\mathbb{T}_t$	the source, target task
n_s	the number of classes defined in $\mathbb{T}_s$
n_t	the number of new entity types defined in $\mathbb{T}_t$
$e_i, i \leq n_s$	the i-th predefined entity type in $\mathbb{T}_s$
e_{n_s+j}	the j-th new entity type defined in $\mathbb{T}_t$
$\mathbb{L}_j$	the lexicon of e_{n_s+j}
$\mathcal{D}^s$, $\mathcal{D}^t$	the labeled, partially labeled data for $\mathbb{T}_s$ and $\mathbb{T}_t$
$\mathcal{D}^t_i$	the obtained labeled data of class i in $\mathcal{D}^t$
$\mathcal{D}^t_u$	the unlabeled data in $\mathcal{D}^t$
π_i	the ratio of class i data in $\mathcal{D}^t$
π'_{n_s+j}	the ratio of class $(n_s + j)$ data in $\mathcal{D}^t_u$

Table 1: Some important notations used throughout this work.

mention correspond to the same class label *without distinction of their positions in the mention*, and words not belonging to any entity type are grouped into a single class ("O" of "Predict" in Figure 1).

The contribution of this work is threefold: **1)** We explore fast transferring from a source NER system (task) to a target NER system (task). This setting has a wide range of applications in the real world but has been rarely studied. **2)** We propose to perform the task using only labeled data of the source task and entity lexicons of the newly introduced entity types, avoiding laborious and time-consuming data labeling. **3)** We formulate the task as a partially supervised learning problem and accordingly propose an effective algorithm to address it.

2 Approach

2.1 Task Definition

In the setting of this work, there is a source and a target NER system, in which a source and a target NER task, $\mathbb{T}_s$ and $\mathbb{T}_t$, are defined. For $\mathbb{T}_s$, a labeled dataset, $\mathcal{D}^s$, is available, in which n_s classes are defined (each class corresponds to an entity type or the Other class). Compared with $\mathbb{T}_s$, the target task, $\mathbb{T}_t$, needs to additionally recognize some new entity types. Without loss of generality, we assume that the newly introduced n_t entity types all belong to class K in the source task. For an intuitive understanding, consider introducing the two entity types, Government Organization and Company, which are all defined as the Organization type in the source task. In this case, the Organization type defined in the source class is the class K in the source task, while the Government Organization and Company are two sub-classes of class K. In this work, we present a way to perform $\mathbb{T}_t$ using only $\mathcal{D}^s$ and the entity lexicons of the new entity types. Table 1 lists some

Class Label	1	$\cdots$	$n_{i\neq O}$	$\cdots$	n_s	n_s+1	$\cdots$	n_s+n_t	K
Labeled Data	$\mathcal{D}_1^t$	$\cdots$	$\mathcal{D}_{i\neq O}$	$\cdots$	$\mathcal{D}_{n_s}^t$	$\mathcal{D}_{n_s+1}^t$	$\cdots$	$\mathcal{D}_{n_s+n_t}^t$	✗
Unlabeled Data	✗					$\mathcal{D}_u^t$			

Table 2: Obtained labeled and unlabeled data for each class in the target task. The labeled data of each predefined entity type is copied from the training data of the source task, while the labeled data of each new entity type is automatically obtained using the lexicons. Note that, there is no labeled data for class K of the source task. Thus, a fully supervised learning algorithm is not applicable to train the classifier.

important notations used throughout this work for convenient reference.

2.2 Label Assignment

We apply the normal multi-label assignment mechanism for performing $\mathbb{T}_t$, instead of the prevalent BIO or BIOES mechanism. That is, the constituted words of a mention of the entity type e_i are all classified to class i without distinction of their positions in the mention. This is because the labeled words by the lexicons may not cover all the constituted words of an entity mention, which means that we cannot distinguish the type, B (beginning), I (internal), or E (end), the words labeled by the lexicons belong to.

2.3 Method Overview

Based on the above label assignment mechanism, we train a (n_s+n_t)-class classifier to perform the target task, $\mathbb{T}_t$. In the classifier, the n_s entity types predefined in $\mathbb{T}_s$ are denoted as $e_i, i=1,\cdots,n_s$ and mapped to class $1,\cdots,n_s$, respectively. The n_t new entity types introduced in $\mathbb{T}_t$ are denoted as $e_{n_s+j}, j=1,\cdots,n_t$ and mapped to class $n_s+j, j=1,\cdots,n_t$, respectively. The challenge for training the classifier is that in $\mathcal{D}^s$, words of the newly introduced n_t entity types are all classified to the same class K in the source task.

For training the classifier, we construct a partially labeled dataset $\mathcal{D}^t$ from $\mathcal{D}^s$ using the lexicons of the newly introduced entity types. Specifically, let $\mathcal{D}_i^t \subseteq \mathcal{D}^t$ denote the labeled data of class i in $\mathcal{D}^t$. $\mathcal{D}_i^t, i \neq K$ is constructed using words of class i in $\mathcal{D}^s$. While for obtaining the labeled data of a new entity type e_j, we use its corresponding entity lexicon $\mathbb{L}_j$ to scan words of class K in $\mathcal{D}^s$ and find out some confident words of the entity type to construct labeled data $\mathcal{D}_{n_s+j}^t$ of class n_s+j. This process applies n_t times to obtain the labeled data of the n_t new entity types. The rest words of class K in $\mathcal{D}^s$ not being selected by the lexicons form the unlabeled data set $\mathcal{D}_u^t \subseteq \mathcal{D}^t$ in the target task, which contains both words of the new entity types (the lexicon cannot cover all its corresponding entities in the data) and words not belonging to any of the newly introduced entity types.

Table 2 lists the available labeled and unlabeled data for each class in the target task after the above process. *It is worth noting from the table that there is no labeled data for class K in the target task.* This means that it is impossible to train the classifier using a normal supervised learning algorithm. To address this challenge, we introduce a novel partially supervised learning algorithm to train the classifier as described in §2.6.

2.4 Obtain the Partially Labeled Data using the Entity Lexicons

In this section, we detail the construction of the partially labeled dataset $\mathcal{D}^t$ for the target task. As illustrated before, the labeled data $\mathcal{D}_i^t, i \leq n_s$ of class i can be easily obtained from $\mathcal{D}^s$ according to the data labeling of $\mathbb{T}_s$. Thus, in the following, we focus on obtaining $\mathcal{D}_{n_s+j}^t, j=1,\cdots,n_t$ and $\mathcal{D}_u^t$ using the entity lexicons. Following the idea of (Peng et al., 2019), we apply the maximum matching algorithm (Xue, 2003) to obtain words that match with the lexicon $\mathbb{L}_j$ and belong to class K in $\mathcal{D}^s$ to construct $\mathcal{D}_{n_s+j}^t$. As summarized in algo. 1, this algorithm is a greedy search routine that walks through a sequence of class K words trying to find the longest string that matches with an entry of the lexicons. Note that in algo. 1, l_w is intuitively set to 4, and the "for" loop is broken in step 12 because a mention must not occur in multiple lexicons, which is guaranteed by $\mathbb{L}_j \cap \mathbb{L}_k = \emptyset$ if $j \neq k$.

2.5 Model Architecture

For a sentence $s = [w_1,\cdots,w_l]$ with l words, we first get the contextualized representations of words using the BERT model (Devlin et al., 2019):

$$\mathbf{h}_1,\cdots,\mathbf{h}_l = \text{BERT}(w_1,\cdots,w_l). \quad (1)$$

Algorithm 1 Data Labeling using the Lexicons

Input: entity lexicons $\mathbb{L}_j$, for $j = 1, \cdots, n_t$ with $\mathbb{L}_j \cap \mathbb{L}_k = \emptyset$ if $j \neq k$, a word sequence $s = \{w_1, \cdots, w_n\} \in$ class K in $\mathbb{D}^s$, and the maximum mention length l_w
Result: the partially labeled dataset $\mathcal{D}^t$
Initialize: $i \leftarrow 1$
while $i \leq n$ **do**
  for $k \in [l_w, \cdots, 0]$ **do**
    $b \leftarrow false$
    for $j \in [1, \cdots, n_t]$ **do**
      if $\{w_i, \cdots, w_{i+k}\} \in \mathbb{L}_j$ **then**
        assign $\{w_i, \cdots, w_{i+k}\}$ to $\mathcal{D}^t_{n_s+j}$.
        $i \leftarrow i + k + 1$
        $b \leftarrow true$
        break
    if b **then**
      break
    if $k == 0$ **then**
      assign w_i to $\mathcal{D}^t_u$
      $i \leftarrow i + 1$

Based on the obtained word representations, we apply a multi-layer perceptron (MLP), f^c, whose last layer activation function is set to softmax, to perform label inference:

$$f^c(\mathbf{h}_i) = \text{MLP}(\mathbf{h}_i). \tag{2}$$

In the following, we denote f as the classifier, with $f(w_i) = f^c(\mathbf{h}_i)$ being a $(n_s + n_t)$-dimensional probability vector.

2.6 Partially Supervised Learning for Model Training

In this section, we discuss how to train the (n_s+n_t)-class classifier using the partially labeled dataset $\mathcal{D}^t$. In the following, $\ell(f(w), i)$ denotes the classification loss defined on the input-label pair (w, i), π_i denotes the ratio of class i data in $\mathcal{D}^t$, and

$$\mathcal{L}^i_j = \frac{1}{|\mathcal{D}^t_j|} \sum_{w \in \mathcal{D}^t_j} \ell(f(w), i)$$

denotes the classification loss defined on the dataset-label pair $(\mathcal{D}^t_j, i)$.

Theoretical foundation. Suppose the labeled data of class K is available and denoted as $\mathcal{D}^t_K$. Then, we can train the classifier on the normal fully supervised learning loss, which is defined as follows:

$$\mathcal{L}_{sup} = \sum_{i=1}^{n_s+n_t} \pi_i \mathcal{L}^i_i. \tag{3}$$

Here, we assume that the value of π_i is known and will discuss its estimation in the next section.

However, due to the absence of $\mathcal{D}^t_K$, we cannot directly obtain the value of $\mathcal{L}^K_K$ and consequently, cannot obtain $\mathcal{L}_{sup}$. To address this problem, we propose a method to estimate $\mathcal{L}^K_K$ using the available labeled and unlabeled data. Specifically, based on the unlabeled data $\mathcal{D}^t_u$, we can obtain the loss defined on the dataset-label pair $(\mathcal{D}^t_u, K)$ as follows:

$$\mathcal{L}^K_u = \frac{1}{|\mathcal{D}^t_u|} \sum_{w \in \mathcal{D}^t_u} \ell(f(w), K).$$

Note that, $\mathcal{D}^t_u$ consists of unlabeled data from class $(n_s + 1)$ to class $n_s + n_t$ and class K. Thus, the right term of the above equation can be factorized as follows:

$$\sum_{j=1}^{n_t} \underbrace{\frac{|\mathcal{D}^t_u(n_s + j)|}{|\mathcal{D}^t_u|}}_{\pi'_{n_s+j}} \times \underbrace{\frac{\sum_{w \in \mathcal{D}^t_u(n_s+j)} \ell(f(w), K)}{|\mathcal{D}^t_u(n_s + j)|}}_{\approx \mathcal{L}^K_{n_s+j}},$$

where $\mathcal{D}^t_u(n_s + j)$ denotes the class $(n_s + j)$ data in $\mathcal{D}^t_u$, and $\pi'_{n_s+j} = \frac{|\mathcal{D}^t_u(n_s+j)|}{|\mathcal{D}^t_u|}$ denotes the ratio of class $(n_s + j)$ in $\mathcal{D}^t_u$. Based on this factorization and the assumption that the data distribution in $\mathcal{D}^t_{n_s+j}$ is close to the data distribution in $\mathcal{D}^t_u(n_s + j)$, we have that:

$$\begin{aligned} \mathcal{L}^K_u \approx & \sum_{j=1}^{n_t} \pi'_{n_s+j} \mathcal{L}^K_{n_s+j} \\ & + \pi'_K \mathcal{L}^K_K. \end{aligned} \tag{4}$$

By reformulating the approximate equation (4), we can obtain an approximation of $\mathcal{L}^K_K$ by:

$$\mathcal{L}^K_K \approx \frac{\mathcal{L}^K_u - \sum_{j=1}^{n_t} \pi'_{n_s+j} \mathcal{L}^K_{n_s+j}}{\pi'_K}, \tag{5}$$

which can be calculated using the unlabeled data and the labeled data of the new entity types. In addition, according to the theoretical and empirical analysis of (du Plessis et al., 2014; Peng et al., 2019), training over this approximate value of $\mathcal{L}^K_K$ is expected to be equivalent to training over its true value if ℓ is *upper-bounded*.

Practical loss definition. According to the above analysis, we implement the classification loss ℓ by the mean square error (MSE):

$$\ell\left(f(w), i\right) = \sum_{j \neq i} (f(w)[j])^2 + (1 - f(w)[i])^2, \tag{6}$$

where $f(w)[i]$ denotes the i-th dimension value of $f(w)$. Here, we implement ℓ with the mean square error instead of the popular cross-entropy loss because the mean square error is upper-bounded (by 1), which is critical for the estimation of $\mathcal{L}_K^K$, while the cross-entropy loss is not (the cross-entropy loss can be infinitely large). The empirical training loss is defined as follows:

$$\begin{aligned} \mathcal{L}_{ps} = & \sum_{i \neq K}^{n_s+n_t} \pi_i \mathcal{L}_i^i \\ & + \underbrace{\frac{\pi_K}{\pi'_K}(\mathcal{L}_u^K - \sum_{j=1}^{n_t} \pi'_{n_s+j} \mathcal{L}_{n_s+j}^K)}_{\pi_K \mathcal{L}_K^K}. \end{aligned} \tag{7}$$

In addition, following the practice of (Kiryo et al., 2017; Peng et al., 2019), we constrain:

$$\mathcal{L}_u^K - \sum_{j=1}^{n_t} \pi'_{n_s+j} \mathcal{L}_{n_s+j}^K > 0, \tag{8}$$

during the minimization of $\mathcal{L}_{ps}$. An intuitive understanding of this constrain is that the loss for class (K) should be non-negative.

Class ratio estimation. To obtain the value of $\mathcal{L}_{ps}$, it is necessary to know the value of the class ratio π_i. Here, we present our method to estimate π_i. For $i \leq n_s$, π_i is estimated by:

$$\pi_i \leftarrow |\mathcal{D}_i^t|/|\mathcal{D}^t|, i = 1, \cdots, n_s,$$

since class i data is fully labeled in $\mathcal{D}^t$. For estimating π_{n_s+j} and $\pi'_{n_s+j}, j = 1, \cdots, n_t + 1$, we apply an iteration strategy. In particular, we first initialized π_{n_s+j} and π'_{n_s+j} for $j \leq n_t$ by $|\mathcal{D}_{n_s+j}^t|/|\mathcal{D}^t|$, and initialize π_K and π'_K by $|\mathcal{D}_u^t|/|\mathcal{D}^t|$ and 1, respectively. Based on this, we train the classifier f and then re-estimate π_{n_s+j} and π'_{n_s+j} using the trained classifier as follows:

$$\begin{aligned} \pi_{n_s+j} &\leftarrow \frac{1}{|\mathcal{D}^t|} \sum_{w \in \mathcal{D}^t} f(w)[n_s + j], \\ \pi'_{n_s+j} &\leftarrow \frac{1}{|\mathcal{D}_u^t|} \sum_{w \in \mathcal{D}_u^t} f(w)[n_s + j], \end{aligned} \tag{9}$$

This process iterates several times to get the final estimations of π_{n_s+j} and π'_{n_s+j}. Note that, according to the theoretical analysis of Kato et al. (2018), π_{n_s+j} and π'_{n_s+j} will converge to fixed values.

2.7 Lexicon Adaptation

It has been proved to be an effective technique to improve the model performance by iteratively enriching the lexicons in a self-training style (Peng et al., 2019). We follow this technique in our method. In particular, we use the trained classifier to perform label prediction for words of $\mathcal{D}_u^t$. Among the predicted entity mentions of the new entity types, we add the frequently occurred ones into the lexicons, which are then used for data labeling in the next iteration. This process repeats several times until the lexicons do not change.

2.8 Label Inference

For a query sentence, it first performs label prediction for the constituted words using the trained classifier f as follows:

$$y(w) = \arg\max_i f(w)[i]. \tag{10}$$

The consecutive words being predicted to be of the same class form an entity mention. For example, for the sentence $s = \{w_1, w_2, w_3, w_4, w_5\}$, if the predicted label sequence is $\{1, 1, 3, 4, 4\}$ with $n_s = 2$ and $n_t = 1$, then $\{w_1, w_2\}$ and $\{w_3\}$ are treated as entity mentions of type e_1 and type e_3, respectively.

3 Related Work

NER is a well studied natural language processing (NLP) task. Once a time, many NER systems are knowledge-based (Nadeau et al., 2006; Gerner et al., 2010; Liu et al., 2015). They do not require annotated training data but heavily rely on background knowledge (rules) and lexicon resources. They work well when the lexicon is exhaustive, but fail when the lexicon is incomplete. Precision is generally high for these systems, but recall is often low due to incomplete lexicons.

Current state-of-the-art NER systems are mainly based on annotated data and machine learning approaches. The lexicons introduced in some of these systems are mainly for extracting some external features (Liu et al., 2015; Agerri and Rigau, 2016; Chiu and Nichols, 2016). This field has been previously dominated by the graph

Dataset	#Sent	#Word	Entity Types in Class K	%Mention by Type
CoNLL03 (en)	14,041	203,621	PER;LOC	.283/.304
CoNLL02 (sp)	8,323	264,715	PER;LOC	.231/.262
MUC-7	3,405	76,987	PER;LOC	.275/.306
Twitter	4,000	64,439	PER;LOC	.363/.334
OntoNotes4.0 (cn)	15,700	49,190	GPE;LOC	.371/.069

Table 3: Task information built on five public NER datasets, including the sentence number (#Sent) and word number (#Word) in $\mathcal{D}^s$ (also $\mathcal{D}^t$), entity types comprising of class K of the source task (also the newly introduced entity types in the target task), and the mention ratio of each entity type (e.g., 28.3% entity mentions are of the person type in CoNLL03 (en)).

models like Hidden Markov Models (HMM) (Zhou and Su, 2002), Maximum Entropy Markov Models (MEMM) (Malouf, 2002; McCallum et al., 2000), and Conditional Random Field (CRF) (McCallum and Li, 2003). Starting with (Collobert et al., 2011), neural network NER systems with minimal feature engineering have become popular. Such models do not require exhausted feature engineering. Various neural architectures have been proposed, like the bi-directional long short-term memory network (LSTM) plus a CRF layer (Huang et al., 2015), the convolutional neural network (CNN) plus a CRF layer, the combination of LSTM and CNN (Chiu and Nichols, 2016), and the BERT based LSTM+CRF model (Jiang et al., 2019; Hakala and Pyysalo, 2019).

One of the most related works is (Peng et al., 2019). This compared work proposes to perform NER using entity lexicons and unlabeled data. For this purpose, a distinct binary classifier is trained for each entity type using the unbiased positive-unlabeled (PU) learning algorithm (du Plessis et al., 2014; Kiryo et al., 2017). At the inference time, the recognition results of the binary classifiers for different entity types are combined to make the final decision. The difference between the compared work and our work is that, in the compared work, the mention recognition for one entity type is performed independently to the other types through a binary classifier. Consequently, it has to resolve the conflict between the recognition results of different binary classifiers for different entity types using a heuristic method at the inference time. While, in this work, the mention recognition for different entity types are performed simultaneously using a single model. This way, the recognition for different different entity types can enhance each other, and it can also avoid heuristically resolving the recognition conflict at the inference time.

4 Experiments

4.1 Datasets

Following the experimental setting of the most related work (Peng et al., 2019), we performed the experiments on the four public NER datasets, including Conll03 (en) in English (Tjong Kim Sang and De Meulder, 2003), CoNLL02 (sp) (Sang and Erik, 2002) in Spanish, MUC-7 (Chinchor, 1998), Twitter (Zhang et al., 2018) in English, and OntoNotes4.0 (Weischedel et al., 2011) in Chinese. For the former four datasets, we treated the location (LOC) and person (PER) types as the newly introduced entity types in the target task, and treated the rest entity types as the predefined entity types in the source task. While for OntoNotes4.0, we treated the GPE (countries, cities, states) and location (non-GPE locations, mountain ranges, bodies of water) types as the newly introduced entity types in the target task, which are all classified as the location type in the source task. Table 3 shows this setting and some statistic information of these datasets.

4.2 Lexicon Collection

We used the same entity lexicons of the person and location types as (Peng et al., 2019) to perform the experiments. According to the illustration of the refereed work, the collection of these lexicons is quite easy. For example, the lexicon of the person type is constructed from 2,000 popular English names in England and Wales in 2015 from ONS, and the lexicon of the location type is constructed from names of countries and their top two popular cities and 200 popular mountain names. The resultant person and location lexicons contain 2,000 distinct person names and 948 location

Dataset	Type	Label Based				Lexicon Based				
		CRF	BiLSTM	BiLSTM+CRF	BERT	Match	bnPU	AdaPU	bnPS (**our**)	AdaPS (**our**)
CoNLL03 (en)	PER	93.12	94.21	95.71	**98.02**	12.06	90.15	93.02	94.77	**95.53**
	LOC	91.15	91.76	93.02	**93.94**	53.44	84.77	85.75	88.67	**90.28**
	Overall	92.08	92.87	94.26	**95.03**	31.55	87.48	89.32	91.59	**92.89**
CoNLL02 (sp)	PER	86.77	88.93	90.41	**96.82**	31.25	87.49	89.83	94.49	**95.14**
	LOC	80.30	75.43	80.55	**86.45**	47.12	75.26	76.97	80.06	**80.48**
	Overall	83.02	81.10	84.62	**90.94**	39.69	80.59	82.42	86.13	**86.69**
MUC-7	PER	87.50	85.71	84.55	**90.93**	18.28	86.16	86.52	**93.33**	92.56
	LOC	83.83	79.48	83.43	**90.25**	56.59	76.72	77.36	79.40	**80.33**
	Overall	85.56	82.50	83.92	**90.54**	41.18	80.45	81.09	85.28	**85.54**
Twitter	PER	80.86	80.61	80.77	**85.32**	27.24	81.28	81.51	**82.22**	82.00
	LOC	75.39	73.52	72.56	**81.08**	36.93	74.96	75.02	76.24	**76.45**
	Overall	78.22	77.24	76.85	**83.27**	31.60	78.24	78.39	**79.39**	79.38
OntoNotes4.0 (cn)	GPE	64.37	65.73	68.21	**79.66**	44.42	68.17	68.15	**72.37**	71.27
	LOC	25.03	25.92	35.29	**43.65**	23.17	33.28	34.33	36.40	**37.22**
	Overall	61.02	61.81	65.15	**77.93**	40.33	65.79	66.22	**70.66**	69.78

Table 4: Testing chunk-level F1 on the target task. The four label-based methods are fully supervised and trained on the fully re-annotated data of the source task. While the five lexicon-based methods train the model using only the existing labels of the source task and entity lexicons of the new entity types. The best performance in each group is marked in a **boldface**.

names, respectively. We refer you to the referred work for more information about the lexicons. Here, we address that it can only label a small part of the mentions of the person and location types using the lexicons.

4.3 Compared Methods

In the following, we refer to **SourceBERT** as the BERT based model trained on the *source task*. The compared methods can be divided into two groups. The first group of methods perform the target task using only $\mathcal{D}^s$ and the entity lexicons of the new entity types, including the **Match** method that directly uses the lexicons to search for the mentions of the new entity types according to algorithm 1, and the **bnPU** method as well as its lexicon-adapted version **AdaPU** proposed by (Peng et al., 2019). For these methods, we combined their recognition result with that of SourceBERT to perform entity recognition. In particular, for a query sentence, we first perform label inference using SourceBERT and then apply these methods to words being predicted to be the "O" class by SourceBERT to further identity mentions of the new entity types. This practice also applies to our proposed method **AdaPS** as well as its variant, **bnPS** without lexicon adaptation.

The second group of methods are fully supervised, including the benchmark CRF model **Stanford NER (CRF)** (Lafferty et al., 2001; Finkel et al., 2005), the bi-directional long short-term memory network with the CRF layer **BiLSTM+CRF** or not **BiLSTM** (Huang et al., 2015), and the **BERT** based model (Devlin et al., 2019) described in the "Model architecture" section. These supervised models were trained on the fully re-annotated $\mathcal{D}^s$ according to the data labeling criteria of the target task.

4.4 Implementation

Implementation of the fully supervised methods except BERT fellow the protocol of (Peng et al., 2019). The BERT model was initialized using the bert-base-cased[1] model for the three English datasets, and initialized it using the bert-multilingual-base-cased[2] model for CoNLL02 (sp) and OntoNotes4.0 (cn); f^c was implemented with a one-layer MLP ($768 \xrightarrow{\text{softmax}} n_s + n_t$). Parameter updating was implemented using the Adam (Kinga and Adam, 2015) optimizer with learning rate set to be 5e-5. For a fair comparison with our methods, we replaced the BiLSTM-based sequence modeling layer of bnPU and AdaPU with the BERT module, which showed better performance.

[1] https://s3.amazonaws.com/models.huggingface.co/bert/bert-base-cased-pytorch_model.bin

[2] https://s3.amazonaws.com/models.huggingface.co/bert/bert-base-multilingual-cased-pytorch_model.bin

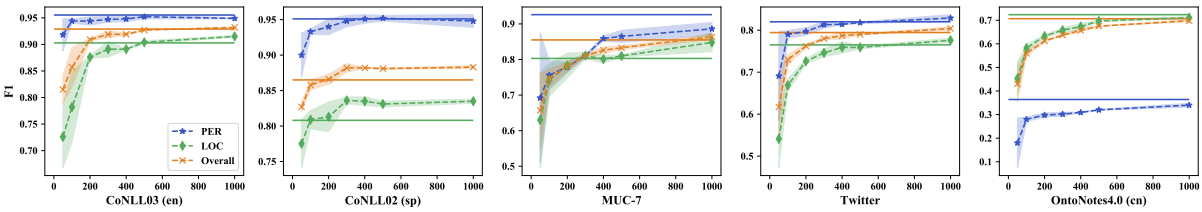

Figure 2: Testing chunk-level F1 (mean ± std. over 4 runs) of the BERT model against the re-annotated sentence number for its finetuning (best view in color). The dot line denotes the performance of the BERT model, while the solid line in the same color denotes the corresponding performance of our method, AdaPS. Note that AdaPS does not use re-annotated data, thus its performances stays the same along the x-axis.

4.5 Results

Following the protocol of most previous works, we apply the *chunk-level* (exact mention match) F1 to evaluate the model performance. We report the F1 score on the mention set of each new entity type, as well as the overall F1 score on the mention set of all new entity types. Note that, our methods and the other lexicon-based baselines are only applied to words being predicted as class K class by SourceBERT. Thus, their performance should be the same for the predefined entity types and determined by SourceBERT.

General performance. Table 4 shows the model performance on the four tested datasets. From the table, we can observe that: **1)** Our methods, AdaPS and bnPS, consistently outperform their PU-learning based counterparts AdaPU and bnPU. This shows the advantage of our methods over the PU-learning baselines. **2)** Compared with bnPU and bnPS, AdaPU and AdaPS can achieve further improvement on most of the four tested tasks. This verifies the effectiveness of lexicon adaptation. However, the improvement of AdaPS over bnPS is much smaller than the improvement of AdaPU over bnPU. Possible explanation is that bnPS has achieved much better performance than bnPU, thus achieving further improvement over bnPS will be harder than over bnPU. **3)** The performance of the Match baseline is quite poor (mainly due to the small recall). This observation is consistent with the reported result in previous works, and shows the insufficiency of the purely lexicon-matching strategy. **4)** Compared with BiLSTM and BiLSTM+CRF, the BERT based model achieves much better performance on the four tested tasks. This shows the effectiveness of the pretrained BERT model for NER. **5)** Our method AdaPS and bnPS can achieve quite comparable performance with the fully supervised BERT model, which requires to re-annotate $\mathcal{D}^s$. In addition, enhanced by the pretrained BERT model, our methods even outperform the fully supervised CRF, BiLSTM, and BiLSTM+CRF models on the CoNLL03 (sp), MUC-7, and Twitter datasets. This shows the efficiency of our methods in transferring from the source task to the target task.

Compared with model finetuning. In this study, we explore how much re-annotated data it requires for the BERT model to achieve similar performance as our proposed method, AdaPS. Figure 2 show the performance of BERT when using varying sizes of randomly sampled re-annotated data to finetune SourceBERT (with the output layer replaced). From the figure, we can see that: **1)** Concerning about the overall F1 score, it averagely requires to re-annotate about 500, 200, 750, 750, and 1,000 sentences of $\mathcal{D}^s$ to achieve similar performance as our method on CoNLL03 (en), CoNLL02 (sp), MUC-7, Twitter, and OntoNotes4.0 (cn), respectively. **2)** To achieve similar performance as our method for all new entity types, it averagely requires to re-annotate about 500, 500, 1,000, and 750 sentences of $\mathcal{D}^s$ On CoNLL03 (en), CoNLL02 (sp), MUC-7, and Twitter, respectively. **3)** On OntoNotes4.0 (cn), it requires to re-annotate more data for the location type than for the GPE type. This is because the occurring frequency of mentions of the location type is much lower than the occurring frequency of mentions of the GPE type. Thus, it requires to annotate more data for the location type to cover enough mentions of the type.

Influence of SourceBERT for label inference. As mentioned in the "compared methods" section, we combined the recognition results of our

Dataset	Type	bnPS - SourceBERT	bnPS
CoNLL03 (en)	Predefined	82.58	**83.43**
	New	91.27	**91.59**
	Overall	87.51	**88.02**
CoNLL02 (sp)	Predefined	81.39	**83.13**
	New	**86.52**	86.13
	Overall	83.98	**84.61**
MUC-7	Predefined	79.01	**80.74**
	New	84.54	**85.28**
	Overall	82.03	**83.19**
Twitter	Predefined	47.83	**48.33**
	New	**79.56**	79.39
	Overall	**69.96**	69.68

Table 5: Testing chunk-level F1 of bnPS on the mention set of the predefined entity types (Predefined), the new entity types (New), and both of them (Overall), when combining with SourceBERT (bnPS) or not (bnPS-SourceBERT) to perform the target task.

method with those of the SourceBERT model to perform entity recognition for the target task. Here, we study the influence of SourceBERT on the recognition results. Table 5 shows the performance of our method, bnPS, when using the trained classifier f only and when additionally using SourceBERT to perform entity recognition for the target task. From the table, we can see that: **1)** It can consistently improve the recognition performance of our method for the predefined entity types by introducing the SourceBERT model, and on three of the four tested tasks, it can also improve the overall recognition performance of our method. **2)** For the newly introduced entity types, the improvement introduced by SourceBERT is relatively smaller, and the improvement is even negative on some tasks.

Let $p(x)$ denote the data distribution of the target domain and $p(x|\mathcal{D}_{in})$ denote the data distribution modeled based on the target data $\mathcal{D}_{in}$. According to the setting of this work, the size of $\mathcal{D}_{in}$ should be small. This means $p(x|\mathcal{D}_{in}) \neq p(x|\mathcal{D}_{in})$. Or more specifically, there are quite a few regions $x \in \mathcal{X}$ that $p(x) > \delta$ while $p(x|\mathcal{D}_{in}) < \delta$, where $\delta > 0$ is a threshold described in the following.

Note that the anomaly detection method will only extract examples $x \in \mathcal{X}$ where $p(x \in \mathcal{X}||\mathcal{D}_{in}) > \delta$ as the target data. This means that the method is still not able to address the long-tail distribution problem introduced by the small size of the task data. In addition, the distribution of the selected data is determined by the general domain data but not the target data. This means that the method is also sensitive to the selection of the general domain.

Let $p(x)$ denote the data distribution of the target domain and $p(x|\mathcal{D}_{in})$ denote the data distribution modeled based on the target data $\mathcal{D}_{in}$. According to the setting of this work, the size of $\mathcal{D}_{in}$ should be small. This means that there are quite a few regions $x \in \mathcal{X}$ that $p(x) > \delta$ while $p(x|\mathcal{D}_{in}) < \delta$, where $\delta > 0$ is a threshold described in the following. Note that the anomaly detection method will only extract examples $x \in \mathcal{X}$ that $p(x \in \mathcal{X}|\mathcal{D}_{in}) > \delta$ as the target data. This means that the method is still not able to address the long-tail distribution problem introduced by the small size of the target data. In addition, the distribution of the selected data is determined by the general domain data but not the target data. This means that the method is sensitive to the selection of the general domain.

5 Conclusion

In this work, we address the task to introduce one or more new entity types to an existing NER system, for which a dataset has been previously labeled. To avoid laborious and time-consuming data labeling, we propose a partially supervised learning algorithm to perform the task using only the labeled data of the existing NER system and entity lexicons of the new entity types. Experimental studies on four public NER datasets show that our method can achieve quite comparable performance with the fully supervised methods using some easily collected lexicons. This makes our method a good choice for fast entity type introduction.

Acknowledgements

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by China National Key RD Program (No. 2017YFB1002104, 2018YFC0831105, 2018YFB1005104), National Natural Science Foundation of China (No. 61976056, 61751201, 61532011), Shanghai Municipal Science and Technology Major Project (No.2018SHZDZX01), Science and Technology Commission of Shanghai Municipality Grant (No.18DZ1201000, 17JC1420200).

References

Rodrigo Agerri and German Rigau. 2016. Robust multilingual named entity recognition with shallow semi-supervised features. *Artificial Intelligence*, 238:63–82.

Bogdan Babych and Anthony Hartley. 2003. Improving machine translation quality with automatic named entity recognition. In *Proceedings of the 7th International EAMT workshop on MT and other Language Technology Tools*, pages 1–8. Association for Computational Linguistics.

Nancy Chinchor. 1998. Overview of muc-7. In *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29-May 1, 1998*.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association of Computational Linguistics*, 4(1):357–370.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

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.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics.

Martin Gerner, Goran Nenadic, and Casey M Bergman. 2010. Linnaeus: a species name identification system for biomedical literature. *BMC bioinformatics*, 11(1):85.

Mourad Gridach. 2017. Character-level neural network for biomedical named entity recognition. *Journal of biomedical informatics*, 70:85–91.

Tao Gui, Yicheng Zou, Qi Zhang, Minlong Peng, Jinlan Fu, Zhongyu Wei, and Xuan-Jing Huang. 2019. A lexicon-based graph neural network for chinese ner. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1039–1049.

Kai Hakala and Sampo Pyysalo. 2019. Biomedical named entity recognition with multilingual bert. In *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, pages 56–61.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Shaohua Jiang, Shan Zhao, Kai Hou, Yang Liu, Li Zhang, et al. 2019. A bert-bilstm-crf model for chinese electronic medical records named entity recognition. In *2019 12th International Conference on Intelligent Computation Technology and Automation (ICICTA)*, pages 166–169. IEEE.

Masahiro Kato, Liyuan Xu, Gang Niu, and Masashi Sugiyama. 2018. Alternate estimation of a classifier and the class-prior from positive and unlabeled data. *arXiv preprint arXiv:1809.05710*.

D Kinga and J Ba Adam. 2015. A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, volume 5.

Ryuichi Kiryo, Gang Niu, Marthinus C du Plessis, and Masashi Sugiyama. 2017. Positive-unlabeled learning with non-negative risk estimator. In *Advances in Neural Information Processing Systems*, pages 1675–1685.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Xu, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. *AAAI Conference on Artificial Intelligence*.

Shengyu Liu, Buzhou Tang, Qingcai Chen, and Xiaolong Wang. 2015. Effects of semantic features on machine learning-based drug name recognition systems: word embeddings vs. manually constructed dictionaries. *Information*, 6(4):848–865.

Khai Mai, Thai-Hoang Pham, Minh Trung Nguyen, Tuan Duc Nguyen, Danushka Bollegala, Ryohei Sasano, and Satoshi Sekine. 2018. An empirical study on fine-grained named entity recognition. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 711–722.

Robert Malouf. 2002. Markov models for language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *ICML*, volume 17, pages 591–598.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics.

David Nadeau, Peter D Turney, and Stan Matwin. 2006. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 266–277. Springer.

Minlong Peng, Xiaoyu Xing, Qi Zhang, Jinlan Fu, and Xuan-Jing Huang. 2019. Distantly supervised named entity recognition using positive-unlabeled learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2409–2419.

Marthinus C du Plessis, Gang Niu, and Masashi Sugiyama. 2014. Analysis of learning from positive and unlabeled data. In *Advances in neural information processing systems*, pages 703–711.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84.

Tjong Kim Sang and F Erik. 2002. Introduction to the conll-2002 shared task: language-independent named entity recognition. *Computer Science*, pages 142–147.

Burr Settles. 2004. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*, pages 104–107. Association for Computational Linguistics.

Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. 2012. Linden: linking named entities with knowledge base via semantic knowledge. In *Proceedings of the 21st international conference on World Wide Web*, pages 449–458. ACM.

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 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. 2011. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *International Journal of Computational Linguistics & Chinese Language Processing, Volume 8, Number 1, February 2003: Special Issue on Word Formation and Chinese Language Processing*, 8(1):29–48.

Qi Zhang, Jinlan Fu, Xiaoyu Liu, and Xuanjing Huang. 2018. Adaptive co-attention network for named entity recognition in tweets. In *AAAI*.

Yue Zhang and Jie Yang. 2018. Chinese ner using lattice lstm. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL), 1554-1564.*

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an hmm-based chunk tagger. In *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 473–480. Association for Computational Linguistics.

Jianhan Zhu, Victoria Uren, and Enrico Motta. 2005. Espotter: Adaptive named entity recognition for web browsing. In *Biennial Conference on Professional Knowledge Management/Wissensmanagement*, pages 518–529. Springer.