Domain Generalization for Named Entity Boundary Detection via Meta-Learning

Jing Li, Shuo Shang and Lisi Chen

***Abstract*—Named entity recognition (NER) aims to recognize mentions of rigid designators from text belonging to predefined semantic types such as person, location, organization etc. In this paper, we focus on a fundamental sub-task of NER, *Named Entity Boundary Detection* which aims at detecting the start and end boundaries of an entity mention in the text, without predicting its semantic type. The entity boundary detection is essentially a sequence labeling problem. Existing sequence labeling methods either suffer from sparse boundary tags (*i.e.,* entities are rare and non-entities are common) or they cannot well handle the issue of variable size output vocabulary (*i.e.,* need to retrain models with respect to different vocabularies). To address these two issues, we propose a novel entity boundary labeling model which leverages pointer networks to effectively infer boundaries depending on the input sequence. On the other hand, training models on source domains that generalize to new target domains at test time is a challenging problem because of the performance degradation. To alleviate this issue, we propose METABDRY, a novel domain generalization approach for entity boundary detection without requiring any access to target domain information. Specially, adversarial learning is adopted to encourage domain-invariant representations. Meanwhile, meta-learning is used to explicitly simulate a domain shift during training so that meta-knowledge from multiple resource domains can be effectively aggregated. As such, METABDRY explicitly optimizes the capability of "learning to generalize", resulting in a more general and robust model to reduce the domain discrepancy. We first conduct experiments to demonstrate the effectiveness of our novel boundary labeling model. We then extensively evaluate METABDRY on eight datasets under domain generalization settings. The experimental results show that METABDRY achieves state-of-the-art results against recent seven baselines.**

***Index Terms*—Named entity recognition (NER), sequence labeling, domain generalization, meta-learning**

Example: Results from Stanford NER

The CReW also offers employees and up to [**3**]$_{\text{NUMBER}}$ of their guests, free entry to the Zoo, [**River**]$_{\text{TITLE}}$ Safari, [**Bird Park**]$_{\text{LOCATION}}$ and our latest offering, the [**Night**]$_{\text{TIME}}$ Safari.

Example: Results from Our Approach

The **CReW** also offers employees and up to 3 of their guests, free entry to the **Zoo**, **River Safari**, **Bird Park** and our latest offering, the **Night Safari**.

Fig. 1. Motivating examples.

I. INTRODUCTION

NAMED Entity Recognition (NER) aims at jointly resolving the boundaries (*i.e.,* start and end positions) and type of a named entity (*e.g.,* person, location and organization) in text [1]. NER is a fundamental task in natural language processing (NLP) and has attracted increasing attention in the field of artificial intelligence [2]. In this paper, we ignore the entity typing and focus on the sub-task of *named entity boundary detection*, which involves detecting the start and end boundaries of an entity mention in text.

This sub-task is interesting and important for several reasons. ***First***, *fine-grained* entity typing systems, such as FIGER [3], FINET [4], AFET [5], and SANE [6], have recently gained significant research interest. Most studies on fine-grained typing either manually label entity boundaries or assume that entity boundaries have already been pre-detected [4], [7]. It then becomes a multi-label classification task. Fine-grained typing systems require type-ignored entity mentions as independent input, posing an overwhelming demand for more accurate and robust boundary detection approaches. ***Second***, an alternative solution is to utilize off-the-shelf NER systems to detect named entity boundaries [4], [5]. However, off-the-shelf systems are not specifically designed for entity boundary detection. As shown in Figure 1, Stanford CoreNLP is unable to correctly identify the entities *CReW*, *River Safari* and *Night Safari*. As a consequence, errors made in entity boundary detection inevitably mislead and adversely affect subsequent entity-typing systems. ***Third***, ignoring entity types results in a homogeneous label space for all entities from heterogeneous domains. This can be helpful for generalizing a robust cross-domain model to detect named entity boundaries.

Essentially, named entity boundary detection can be treated as a sequence labeling problem, where the task is to predict a sequence of 'yes/no' boundary tags at word level in a sentence to identify the start and end positions of an entity mention. Existing sequence labeling techniques can be broadly categorized into two paradigms: (i) recurrent neural networks with conditional random fields (RNN-CRF) [8]–[10]; (ii) one RNN to encode input sequences and another RNN as a language model to generate the output sequence (RNN-RNN) [11], [12]. However, due to the sparsity of 'yes' boundary tags (*i.e.,* entities are rare and non-entities are common), CRFs did not provide any additional gain over simple classifiers like MaxEnt [13], [14]. On the other hand, when applying RNNs for predictions, they cannot well handle the issue of variable size output vocabulary (*i.e.,* need to retrain models

J. Li is with the Inception Institute of Artificial Intelligence (IIAI), Abu Dhabi 54115, United Arab Emirates. (e-mail: jingli.phd@hotmail.com).

S. Shang and L. Chen are with University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: jedi.shang@gmail.com; chenlisi.cs@gmail.com).

Manuscript received Nov 20, 2019; revised xxx xx, xx.

with respect to different vocabularies) [15], [16]. Here, we seek a new sequence labeling approach to alleviate these two issues.

Off-the-shelf systems often suffer from a shift in data distribution from the test data, resulting in poor performance. Some studies [17]–[22] have investigated *domain adaptation* in NER, aiming to transfer knowledge from one source domain to another target domain, requiring some instances from the target domain to perform adaptation. Instead, we target on a more ambitious task, *domain generalization*, which is a particularly challenging problem setting as we assume no access to any target information. That is, the model should be general and robust enough to perform well in new unseen domains, without further parameter updates [23]. Just like recently proposed GPT-3 which is applied without any gradient updates or fine-tuning[1], the key idea of this paper is to utilize the abundant data from multiple resource domains to aggregate meta-knowledge for named entity boundary detection.

Recently, meta-learning [24]–[26] has received resurgence in the context of few-shot learning. However, meta-learning focuses on learning how to learn or to quickly adapt to new information with little data. Inspired by MAML [26] and feature-critic networks [27], we apply meta-learning in the domain generalization setting to distill meta-knowledge from multiple resource domains by explicitly simulating a domain shift during training. In this way, the validation error on unseen domains can serve as a type of feedback to guide the learning of a general model. Moreover, we also incorporate adversarial learning to further encourage domain-invariant representations during training. In summary, the main contributions of this work are four-fold:

- To the best of our knowledge, we are the first to study the problem of transferring meta-knowledge learned from multiple source domains for sequence labeling in a meta-learning manner.
- We cast named entity boundary detection as a sequence labeling problem, and then propose a novel boundary labeling model. It has the key advantages of inherently handling variable size output vocabulary and addressing the issue of boundary tag sparsity. Experimental results show that our model achieves state-of-the-art performance in sequence labeling.
- We propose METABDRY, a novel domain generalization approach for named entity boundary detection, which incorporates adversarial learning and meta-learning during training to encourage domain generalization.
- We extensively evaluate METABDRY on eight datasets under domain generalization settings. The experimental results show that METABDRY achieves state-of-the-art performance in reducing the domain discrepancy, against seven recent baselines.

II. Related Work

Our research is related to two research topics: named entity recognition (NER) and meta-learning.

[1]https://github.com/openai/gpt-3

1) Named Entity Recognition (NER): There are three common paradigms for NER [1]: *knowledge-based unsupervised* systems, *feature-based supervised* systems and *neural-based* systems. Recently, many neural architectures have been widely applied in NER because neural-based systems have the advantage of inferring latent features and learning sequence labels in an end-to-end fashion. Thus, we focus on describing neural-based systems. The use of neural models for NER was pioneered by [28], where an architecture based on temporal convolutional neural networks (CNNs) over word sequence was proposed. Since then, there has been a growing body of work on neural-based NER. Existing neural-based systems can be unified into a framework with three components: an input representation, context encoder and tag decoder. Commonly-used input representations include word-level and character-level representations [8], [29]–[31]. Widely-used context encoder architectures include CNNs [28], recurrent neural networks (RNNs) [10], recursive neural networks [32] and deep transformers [33]. At the top of context encoder, a conditional random field (CRF) layer [34] or RNN layer [12] is employed to make sequence label predictions.

A few works have already explored transfer learning in NER. Transfer learning aims to perform a machine learning task on a target domain by taking advantage of knowledge learned from a source domain [35]. In the setting of transfer learning, different neural models for NER commonly share certain parts of model parameters between the source task and target task. Yang *et al.* [17] first investigated the transferability of different layers of representations. Pius and Mark [18] extended Yang's approach to allow joint training on the informal corpus and incorporate sentence-level feature representations. Jia *et al.* [36] utilized the cross-domain language model as a bridge across domains to design a novel parameter generation network. Zhou *et al.* [21] proposed two adversarial transfer network to explore effective feature fusion between high and low resource domains. Different from these parameter-sharing architectures, some methods [20], [37] apply transfer learning in NER by first training a model on a source task and then using it on the target task for fine-tuning. Recently, Li *et al.* [38] proposed an adversarial approach to transfer knowledge between two domains, rather than multiple domains. In addition, Li's approach needs access to the target domain information (*i.e.,* some unlabeled target domain data) for performing adaptation.

Compared with these existing architectures, the main difference in our proposed sequence labeling model is that our tag decoder is a pointer network, not a CRF. It can effectively capture sequential dependencies when boundary tags are sparse. Moreover, our work is the first to aggregate meta-knowledge from multiple resource domains to increase the transferability rather than from one single domain.

2) Meta-learning: Meta-learning (a.k.a. learning to learn) [24], [25] aims to learn a general model that can quickly adapt to a new task given very few training samples without needing to be retrained from scratch. Most recent approaches to meta-learning focus on few-shot learning and can be broadly categorized as metric-based methods [39], memory-based methods [40], and optimization-based methods [26]. A few methods

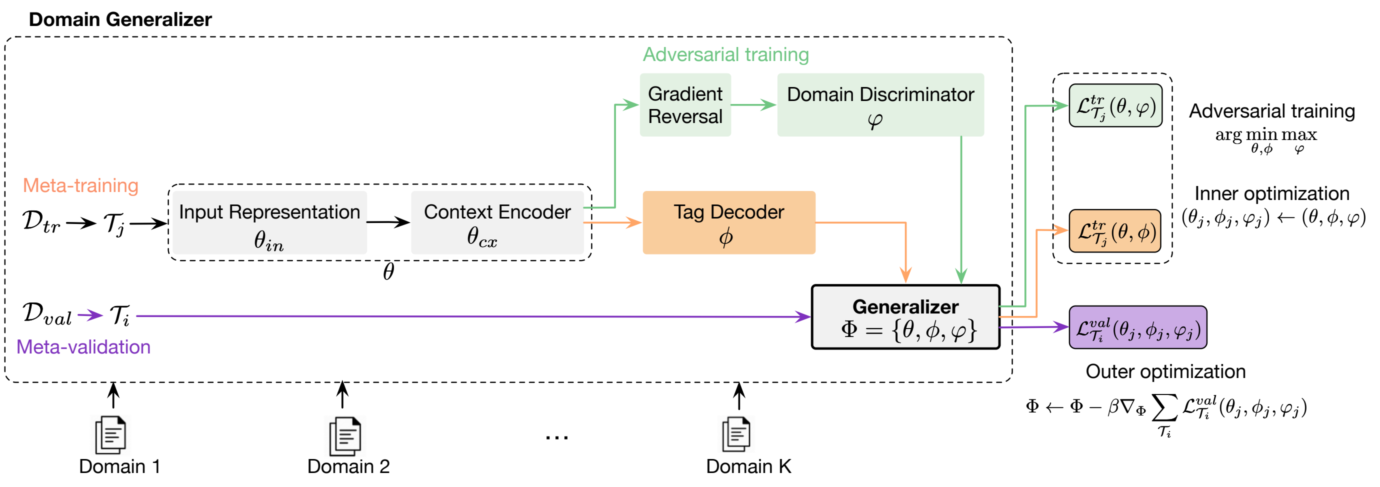

Domain Generalizer
Adversarial training
Gradient Reversal
Domain Discriminator φ
Meta-training
$\mathcal{D}_{tr} \rightarrow \mathcal{T}_j$
Input Representation θ_{in}
Context Encoder θ_{cx}
θ
Tag Decoder ϕ
$\mathcal{D}_{val} \rightarrow \mathcal{T}_i$
Meta-validation
Generalizer $\Phi = \{\theta, \phi, \varphi\}$
$\mathcal{L}^{tr}_{\mathcal{T}_j}(\theta, \varphi)$
Adversarial training $\arg\min_{\theta,\phi}\max_{\varphi}$
$\mathcal{L}^{tr}_{\mathcal{T}_j}(\theta, \phi)$
Inner optimization $(\theta_j, \phi_j, \varphi_j) \leftarrow (\theta, \phi, \varphi)$
$\mathcal{L}^{val}_{\mathcal{T}_i}(\theta_j, \phi_j, \varphi_j)$
Outer optimization $\Phi \leftarrow \Phi - \beta\nabla_\Phi \sum_{\mathcal{T}_i} \mathcal{L}^{val}_{\mathcal{T}_i}(\theta_j, \phi_j, \varphi_j)$
Domain 1
Domain 2
...
Domain K

Fig. 2. An overview of our proposed METABDRY, a domain generalization approach for named entity boundary detection. METABDRY explicitly simulates domain shift during the training process via meta-learning. In the meta-training phase, the adapted parameters $(\theta_j, \phi_j, \varphi_j)$ are learned from $\mathcal{D}_{tr}$. In the meta-validation phase, the base model is updated by gradient descent with respect to the parameters (θ, ϕ, φ) on $\mathcal{D}_{val}$.

[27], [41]–[43] have applied meta-learning strategies for image classification in zero-shot learning.

Meta-learning for NLP is less common than for computer vision. There are a few attempts that have been devoted to the application of meta-learning to NLP in the last two years. Gu *et al.* [44] extended the Model- Agnostic Meta-Learning (MAML) approach [26] for low-resource neural machine translation. Huang *et al.* [45] proposed a method for natural language to structured query generation based on MAML, by reducing a regular supervised learning problem to the few-shot meta-learning scenario. Qian and Zhou [46] trained a dialog system model using multiple rich-resource single-domain dialog data by applying the MAML algorithm to the dialog domain. Lin *et al.* [47] proposed to cast personalized dialogue learning as a meta-learning problem, which allows the model to generate personalized responses by efficiently leveraging only a few dialogue samples instead of human-designed persona descriptions. Obamuyide and Vlachos [48] framed relation classification with a meta-learning perspective. They proposed a MAML protocol for training relation classifiers that explicitly learn a model parameter initialization for enhanced predictive performance in limited supervision settings. Hu *et al.* [49] formulated the learning of out-of-vocabulary embedding as a few-shot regression problem. Furthermore, they proposed a simulated episode-based training schema to predict oracle embeddings, leveraging MAML for adapting the learned model to the new corpus fast and robustly.

Different from the above studies on few-shot learning, our work investigates a more challenging setting, *i.e.,* the domain generalization problem, where the target instances are not accessed during the final testing. To the best of our knowledge, our work is the first attempt in adapting meta-learning to sequence labeling.

III. METABDRY: NAMED ENTITY BOUNDARY DETECTION VIA META-LEARNING

In this section, we first introduce the problem setup of domain generalization for named entity boundary detection. Then we give a detailed description of the proposed METABDRY.

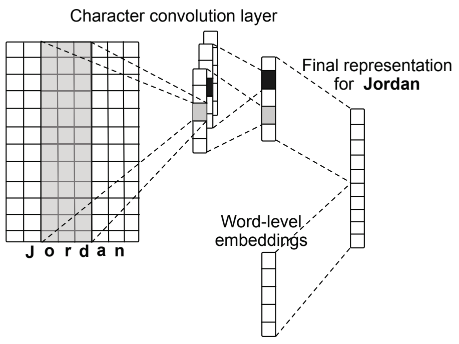
A. Problem Setup

Suppose that there are K source (training) domains $\mathcal{D} = \{\mathcal{D}_1, ..., \mathcal{D}_k, ..., \mathcal{D}_K\}$, where $\mathcal{D}_k$ is the k-th source domain containing annotated data pairs $(\mathcal{X}_k, \mathcal{Y}_k)$. For our named entity boundary detection problem, the input space $\mathcal{X}_k$ is the raw text (*e.g.,* words) and the label space $\mathcal{Y}_k$ is composed of entity boundary tags that indicate the start and end positions of a named entity. The ultimate goal is to use the source domain data set $\mathcal{D}$ to learn a parametric model $f_\Phi(\cdot) : \mathcal{X} \rightarrow \mathcal{Y}$, where the learnable parameters Φ are able to generalize well to a novel target (testing) domain $\mathcal{D}_{test}$ (an unseen domain in training), without requiring any knowledge of the target domain during training.

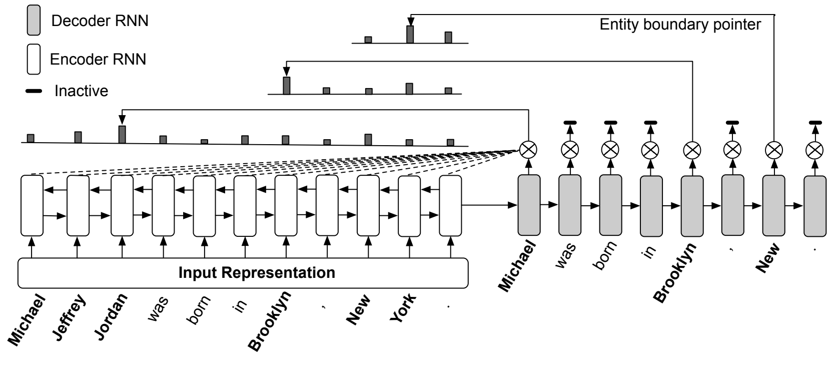
B. Model Overview

To solve the above domain generalization problem, we propose METABDRY, as shown in Figure 2. We design METABDRY based on three key considerations: (i) We expect that the extracted sequence features should be as general as possible so that they are robust enough to perform well on any unseen target domain without fine-tuning. (ii) We want to explicitly simulate a domain shift during the training process so that the validation error on an unseen domain can serve as a type of feedback to guide learning. (iii) We expect that the model should aggregate knowledge from a number of different domains so that it can be directly applied to new domains.

Specifically, for (i), we integrate an adversarial network to encourage internal domain-invariant representations. For (ii) and (iii), we adopt a meta-learning [26] approach to distill meta-knowledge from a number of domains. During each iteration, we randomly split $\mathcal{D}$ into a meta-training set $\mathcal{D}_{tr}$ and meta-validation set $\mathcal{D}_{val}$, where $\mathcal{D} = \mathcal{D}_{tr} \cup \mathcal{D}_{val}$ and $\mathcal{D}_{tr} \cap \mathcal{D}_{val} = \emptyset$. A meta-training task $\mathcal{T}_j$ is sampled from $\mathcal{D}_{tr}$, and is composed of n instances per domain. The tag



(a) Representation of an input word "*Jordan*".



(b) The architecture of pointer networks for entity boundary detection.

Fig. 3. An illustration of entity boundary detection with pointer networks. Input sentence: "*Michael Jeffrey Jordan was born in Brooklyn, New York.*". The identified entities by boundary detection are "*Michael Jeffrey Jordan*", "*Brooklyn*" and "*New York*".

decoder is used to detect named entity boundaries based on intermediate representations from the context encoder. The domain discriminator is used to judge which domain a training instance belongs to. The adversarial network ensures that the intermediate representations can mislead the domain discriminator and correctly guide the tag decoder prediction, while the domain discriminator tries its best to correctly determine the domain class of each training instance.

A meta-validation task $\mathcal{T}_i$ is sampled from $\mathcal{D}_{val}$, and is also composed of n instances per domain. Note that $\mathcal{T}_i$ is evaluated on the parameters $(\theta_j, \phi_j, \varphi_j)$ learned from $\mathcal{D}_{tr}$. Finally, METABDRY updates model parameters Φ using the loss for gradient descent with respect to the base parameters (θ, ϕ, φ). Our proposed METABDRY consists of three core components: the boundary labeling model, the adversarial training and the meta-learning strategies. Next, we will introduce each component in detail.

C. Our Boundary Labeling Model

In this work, we propose a novel sequence labeling model based on the pointer mechanism [15], [50]. Our sequence labeling model has the following advantages. (i) It effectively handles variable size vocabulary in the output to produce entity boundaries depending on the input sequence. (ii) It effectively handles sparse tags because entities are rare and non-entities are common. Figure 3 shows the architecture of our boundary labeling model, which has three key components: the input representation, context encoder and tag decoder.

1) Input Representation: As shown in Figure 3(a), the input representation in our model consists of character-level and word-level representations. Given an input sentence $\boldsymbol{W} = (W_1, W_2, \ldots, W_L)$ of length L, $\boldsymbol{W} \in \mathcal{D}$, let W_l denote its l-th word. The character-level representation (extracted by convolutional neural networks) and word-level embedding (*e.g.,* pretrained embedding) for W_l are concatenated as its final representation, $\boldsymbol{x}_l \in \mathbb{R}^D$, where D represents the dimension of $\boldsymbol{x}_l$. Note that hand-crafted features and other pretrained language embeddings can be easily integrated into this architecture. However, we do not use any hand-crafted features in this study.

2) Context Encoder: We encode the input sequence $\mathbf{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_L)$ using RNNs that are capable of capturing sequential dependencies. With hidden cells like long short-term memory (LSTM) [51] and gated recurrent unit (GRU) [52], an RNN captures long distance dependencies without running into the problems of gradient vanishing or explosion. In our implementation, we use the GRU which is computationally cheaper than LSTM. Specifically, GRU activations at time step l are computed as follows:

$$\boldsymbol{z}_l = \sigma(\boldsymbol{U}_z \boldsymbol{x}_l + \boldsymbol{R}_z \boldsymbol{h}_{l-1} + \boldsymbol{b}_z) \tag{1}$$
$$\boldsymbol{r}_l = \sigma(\boldsymbol{U}_r \boldsymbol{x}_l + \boldsymbol{R}_r \boldsymbol{h}_{l-1} + \boldsymbol{b}_r) \tag{2}$$
$$\boldsymbol{n}_l = \tanh(\boldsymbol{U}_h \boldsymbol{x}_l + \boldsymbol{R}_h (\boldsymbol{r}_l \odot \boldsymbol{h}_{l-1}) + \boldsymbol{b}_h) \tag{3}$$
$$\boldsymbol{h}_l = \boldsymbol{z}_l \odot \boldsymbol{h}_{l-1} + (1 - \boldsymbol{z}_l) \odot \boldsymbol{y}_l \tag{4}$$

where $\sigma(\cdot)$ is the sigmoid function, $\tanh(\cdot)$ is the hyperbolic tangent function, $\odot$ is an element-wise multiplication, $\boldsymbol{z}_l$ is the update gate vector, $\boldsymbol{r}_l$ is the reset gate vector, $\boldsymbol{n}_l$ is the new gate vector, and $\boldsymbol{h}_l$ is the hidden state at time step l. $\boldsymbol{U}$, $\boldsymbol{R}$, $\boldsymbol{b}$ are encoder parameters that need to be learned.

We use a bi-directional GRU (BiGRU) network to memorize past and future information in the input sequence. Each hidden state of the BiGRU is formalized as:

$$\boldsymbol{h}_l = \overrightarrow{\boldsymbol{h}}_l \oplus \overleftarrow{\boldsymbol{h}}_l \tag{5}$$

where $\oplus$ indicates a concatenation operation, and $\overrightarrow{\boldsymbol{h}}_l$ and $\overleftarrow{\boldsymbol{h}}_l$ are hidden states of the forward (left-to-right) and backward (right-to-left) GRUs, respectively. Assuming the size of the GRU layer is H, the encoder yields hidden states in $\boldsymbol{h} \in \mathbb{R}^{L \times 2H}$.

3) Tag Decoder: At each step of decoding, our model takes a word W_l from the input sequence as input, and transforms it to its distributed representation $\boldsymbol{x}_l$ by looking up the corresponding embedding matrix from the encoding phase. It then passes $\boldsymbol{x}_l$ through a GRU-based unidirectional hidden layer. The hidden state at time step m is computed by:

$$\boldsymbol{d}_m = GRU(\boldsymbol{x}_m, \gamma) \tag{6}$$

where γ are the parameters in the hidden layer of the GRU-based RNN, which have the same form as defined in Equations (1) – (4). Note that not every word from the input sentence needs to be passed to the decoder GRU. As shown in Figure 3(b), "*Jordan*" is the end boundary of the mention "*Michael Jeffrey Jordan*", so the two words "*Jeffrey*" and "*Jordan*" will not be passed to the GRU. Supposing there are J time steps, the decoder GRU produces hidden states in $\boldsymbol{d} \in \mathbb{R}^{J \times 2H}$ with $2H$ being the dimensions of the hidden layer of the decoder. Again, the encoder is bidirectional (hidden size H), and the decoder is unidirectional (hidden size $2H$).

In the pointing phase, our model detects entity boundaries only if the current input is a start boundary. Otherwise, it will switch the decoder status to *inactive* and no boundary will be detected. In order to achieve this mechanism, we pad the hidden states of the encoder with a sentinel word representing *inactive*. That is, the decoder should point to this sentinel word once the current input is no longer a start boundary of an entity. Recall that $\boldsymbol{h} \in \mathbb{R}^{L \times 2H}$ and $\boldsymbol{d} \in \mathbb{R}^{J \times H}$ are the hidden states in the encoder and decoder, respectively. We first pad $\boldsymbol{h}$ with a sentinel vector by $\boldsymbol{h} = [\boldsymbol{h}; 0]$, where $\boldsymbol{h} \in \mathbb{R}^{(L+1) \times 2H}$. Then, we use an attention mechanism [15] to compute the distribution of end boundary over all possible positions in the input sequence at decoding step m:

$$u_i^m = \boldsymbol{v}^T \tanh(\boldsymbol{G}_1 \boldsymbol{h}_i + \boldsymbol{G}_2 \boldsymbol{d}_m), \quad \text{for } i \in (m, \ldots, M) \tag{7}$$

$$p(y_m|\boldsymbol{x}_m) = \text{softmax}(\boldsymbol{u}^m) \tag{8}$$

Here, $\boldsymbol{v}$, $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ are learnable parameters, $i \in [m, M]$ indicates a possible position in the input sequence, and *softmax* normalizes u_i^m, indicating the probability that word W_i is an end boundary, given the start boundary W_m. When W_m is not a start boundary of any entity, the pointer is trained to point to the padded word W_{L+1}, *i.e., inactive*. For example, our model points to "*inactive*" when given "*was*" as the decoder input in Figure 3(b).

D. Adversarial Training Strategy

Recall that $\boldsymbol{h} \in \mathbb{R}^{2H}$ is the hidden state of the last step in the context encoder. We apply a Multi-Layer Perceptron (MLP) to predict domain labels y_d:

$$\boldsymbol{\omega} = \text{softmax}(\tanh(\boldsymbol{h} \cdot P + p)) \tag{9}$$

$$\boldsymbol{c} = \boldsymbol{h\omega} \tag{10}$$

$$p(y_d|\boldsymbol{c}) = \text{MLP}(\boldsymbol{c}) \tag{11}$$

The tag prediction loss and the domain prediction loss are calculated over the meta-training samples in task $\mathcal{T}_j$ from $\mathcal{D}_{tr}$. These two losses can be written as

$$\mathcal{L}_{\mathcal{T}_j}^{tr}(\theta, \phi) = \sum_{\mathcal{T}_j} \sum_{m=1}^{M} -\log p(y_m|\boldsymbol{x}_m; \theta, \phi) \tag{12}$$

$$\mathcal{L}_{\mathcal{T}_j}^{tr}(\theta, \varphi) = \sum_{\mathcal{T}_j} -\log p(y_d|\boldsymbol{c}; \theta, \varphi) \tag{13}$$

where θ are the learnable parameters of the shared layers (*i.e.,* input representation and context encoder), ϕ are the parameters of the tag decoder, and φ are the parameters of the discriminator. At learning time, in order to encourage domain-invariant features, we seek the parameters θ that *maximize* the loss of the domain discriminator (by making the two feature distributions as indistinguishable as possible), while simultaneously seeking the parameters θ and φ that *minimize* the loss of the domain discriminator. In addition, we seek the parameters ϕ that *minimize* the loss of the tag decoder. Thus, the optimization problem involves a minimization with respect to some parameters and a maximization with respect to others. Based on this idea, we define the whole objective:

$$\mathcal{L}_{\mathcal{T}_j}^{adv}(\theta, \phi, \varphi) = \mathcal{L}_{\mathcal{T}_j}^{tr}(\theta, \phi) - \lambda \mathcal{L}_{\mathcal{T}_j}^{tr}(\theta, \varphi) \tag{14}$$

The parameter λ controls the trade-off between the two objectives. Then, we deliver a saddle point of $\mathcal{L}_{\mathcal{T}_j}^{adv}(\theta, \phi, \varphi)$ as follows:

$$(\hat{\theta}, \hat{\phi}) = \arg \min_{\theta, \phi} \mathcal{L}_{\mathcal{T}_j}^{adv}(\theta, \phi, \hat{\varphi}) \tag{15}$$

$$\hat{\varphi} = \arg \max_{\varphi} \mathcal{L}_{\mathcal{T}_j}^{adv}(\hat{\theta}, \hat{\phi}, \varphi) \tag{16}$$

Note that the $-\lambda$ factor in Equation (14) is very important because the stochastic gradient descent would directly minimize the domain prediction loss without such factor, resulting in discriminative features across domains only. Following [53], we add a special gradient reversal layer (GRL) below the shared layer to address the minimax optimization problem. During the forward propagation, GRL acts as an identity transform (*i.e.,* multiplies it by 1). During the backpropagation, the GRL takes the gradient from the subsequent level and changes its sign, *i.e.,* $\lambda \mathcal{L}_{\mathcal{T}_j}^{tr}(\theta, \varphi)$ is is effectively replaced with $-\lambda \mathcal{L}_{\mathcal{T}_j}^{tr}(\theta, \varphi)$. Formally, we define the GRL as a function $R_\lambda(\boldsymbol{x})$ by two two equations describing the forward- and backpropagation behaviours:

$$R_\lambda(\boldsymbol{x}) = \boldsymbol{I} \tag{17}$$

$$\frac{dR_\lambda(\boldsymbol{x})}{d\boldsymbol{x}} = -\lambda \boldsymbol{I} \tag{18}$$

where $\boldsymbol{I}$ is an identity matrix. This adversarial training strategy will lead to the emergence of features that are domain-invariant and discriminative at the same time.

E. Meta-Learning Strategy

As shown in Figure 2, METABDRY consists of two core steps: a meta-training step and meta-validation step.

Meta-training (Inner Loop). In the meta-training step, METABDRY tries to learn adaptation parameters from the meta-training domains $\mathcal{D}_{tr}$, resulting in a temporary model. The parameters of the temporary model are adapted by gradient descent [27]:

$$\phi_j = \phi_{j-1} - \alpha \nabla_{\phi_{j-1}} \mathcal{L}_{\mathcal{T}_j}^{tr}(\theta_{j-1}, \phi_{j-1}) \tag{19}$$

$$\varphi_j = \varphi_{j-1} - \alpha \nabla_{\varphi_{j-1}} \mathcal{L}_{\mathcal{T}_j}^{tr}(\theta_{j-1}, \varphi_{j-1}) \tag{20}$$

$$\theta_j = \theta_{j-1} - \alpha \nabla_{\theta_{j-1}} \mathcal{L}_{\mathcal{T}_j}^{adv}(\theta_{j-1}, \phi_{j-1}, \varphi_{j-1}) \tag{21}$$

where j is the adaptation step in the inner loop, and α is the learning rate of the inner optimization. At each adaptation step, the gradients are calculated with respect to the parameters in previous step (*i.e.,* $\nabla_{\phi_{j-1}}, \nabla_{\varphi_{j-1}}, \nabla_{\theta_{j-1}}$). Note that the base model parameters $\theta_0, \phi_0, \varphi_0$ should not be changed in the inner loop (*i.e.,* when updating the temporary model).

Algorithm 1: Training and Testing METABDRY

```
1  Training Procedure()
       Input: $\mathcal{D} = \{\mathcal{D}_1, ..., \mathcal{D}_K\}$, and $\alpha, \beta$
2      Initialize $\theta, \phi, \varphi$;
3      while not converge do
4          Randomly split $\mathcal{D} = \mathcal{D}_{tr} \cup \mathcal{D}_{val}$ and $\mathcal{D}_{tr} \cap \mathcal{D}_{val} = \emptyset$;
5          Meta-training:
6          Let $\Phi = \{\theta_0, \phi_0, \varphi_0\}$ and $\theta_0 = \{\theta_{in}, \theta_{cx}\}$
7          for i in meta batch do                  // Outer loop
8              Sample a task $\mathcal{T}_i$ fom $\mathcal{D}_{val}$;
9              for j in adaptation steps do        // Inner loop
10                 Sample a task $\mathcal{T}_j$ from $\mathcal{D}_{tr}$;
11                 Compute meta-training loss $\mathcal{L}^{tr}_{\mathcal{T}_j}$ using Eq. (12) and (13);
12                 Compute adversarial loss $\mathcal{L}^{adv}_{\mathcal{T}_j}$ using Eq. (14);
13                 Compute adapted parameters with gradient descent for $\theta, \phi, \varphi$: ;    // $\mathcal{T}_j, \nabla_{\theta_{j-1}}$
14                 $\phi_j = \phi_{j-1} - \alpha \nabla_{\phi_{j-1}} \mathcal{L}^{tr}_{\mathcal{T}_j}(\theta_{j-1}, \phi_{j-1})$;
15                 $\varphi_j = \varphi_{j-1} - \alpha \nabla_{\varphi_{j-1}} \mathcal{L}^{tr}_{\mathcal{T}_j}(\theta_{j-1}, \varphi_{j-1})$;
16                 $\theta_j =$
                   $\theta_{j-1} - \alpha \nabla_{\theta_{j-1}} \mathcal{L}^{adv}_{\mathcal{T}_j}(\theta_{j-1}, \phi_{j-1}, \varphi_{j-1})$;
17             Compute meta-validation loss on $\mathcal{T}_i$:
18             $\mathcal{L}^{val}_{\mathcal{T}_i}(\theta_j, \phi_j, \varphi_j)$;
19         Meta-validation:
20         Perform gradient step w.r.t $\Phi$:
             $\Phi \leftarrow \Phi - \beta \nabla_\Phi \sum_{\mathcal{T}_i} \mathcal{L}^{val}_{\mathcal{T}_i}(\theta_j, \phi_j, \varphi_j)$ ;  // $\mathcal{T}_i, \nabla_\Phi$
21     return $\theta_{\text{Meta}}, \phi_{\text{Meta}}$
22 Testing Procedure()
       Input: Learned $\theta_{\text{Meta}}, \phi_{\text{Meta}}$, unseen domain data $\mathcal{D}_{test}$
23     while not end do
24         Serialize a task $\mathcal{T}_j$ from the hold-out domain $\mathcal{D}_{test}$;
25         Evaluate $\mathcal{D}^{\mathcal{T}_j}_{test}$ using the model ($\theta_{\text{Meta}}, \phi_{\text{Meta}}$);
26     return Entity boundary detection performance
```

Meta-validation (Outer Loop). After meta-training, METABDRY has already learned a temporary model $f_{\theta_j, \phi_j, \varphi_j}(\cdot) : \mathcal{X} \rightarrow \mathcal{Y}$ in the meta-training domains $\mathcal{D}_{tr}$. The meta-validation step tries to minimize the distribution divergence between the source domains $\mathcal{D}_{tr}$ and simulated target domains $\mathcal{D}_{val}$ using the learned temporary model. It mimics the process of the temporary model being adapted to unseen domains. More specifically, the outer meta-validation loss $\mathcal{L}^{val}_{\mathcal{T}_i}(\theta_j, \phi_j, \varphi_j)$ is computed on the task $\mathcal{T}_i$ from the meta-validation domains $\mathcal{D}_{val}$ with adapted parameters $(\theta_j, \phi_j, \varphi_j)$. The base model parameters are computed by gradient descent, in order to reduce errors on the validation domains $\mathcal{D}_{val}$:

$$(\theta_0, \phi_0, \varphi_0) \leftarrow (\theta_0, \phi_0, \varphi_0) - \beta \nabla_{\theta_0, \phi_0, \varphi_0} \sum_{\mathcal{T}_i} \mathcal{L}^{val}_{\mathcal{T}_i}(\theta_j, \phi_j, \varphi_j) \tag{22}$$

where β is the meta-learning rate. Note that the gradients are computed by differentiating the loss $\mathcal{L}^{val}_{\mathcal{T}_i}(\theta_j, \phi_j, \varphi_j)$ with respect to the parameters $\theta_0, \phi_0, \varphi_0$. Unlike the common gradient, the update mechanism of Equation (20) involves a gradient (*i.e.,* the base model ∇_{θ_0}) through a gradient (*i.e.,* the temporary model ∇_{θ_j}). This process requires second order optimization partial derivatives. METABDRY explicitly learns both the sequence labeling model and domain adaptation during training.

TABLE I
STATISTICS OF DATASETS.

Datasets	# Sentences			# Mentions
	Train	Dev	Test	
CON	14,041	3,250	3,453	34,841
ONT	59,917	8,528	8,262	71,031
WIK	144,342	500	1,696	300,069
WNU	3,394	1,009	1,287	3,850
RIT	1,000	240	254	1,487
CAD	6,077	760	760	2,057
RE3	687	77	199	3,388
SEC	1,047	117	303	1,479

Algorithm Flow. The full pseudocode for training and testing METABDRY is given in Algorithm 1. At each iteration during training, we randomly split $\mathcal{D}$ into $\mathcal{D}_{tr}$ and $\mathcal{D}_{val}$ for the inner loop and outer loop, respectively. In the inner loop, METABDRY takes a gradient step to get new adaptation parameters, and obtains the new meta-validation loss. In the outer loop, METABDRY uses the validation on $\mathcal{D}_{val}$ to differentiate through the inner loop and update the parameters of the base model: $\theta_0, \phi_0, \varphi_0$.

Our task (*i.e.,* entity boundary detection) ignores the entity typing, resulting in a homogeneous label space. Thus, for the final evaluation, we directly use the meta-learned sequence labeling model (*i.e.,* θ_{Meta} and ϕ_{Meta}) for a given target domain $\mathcal{D}_{test}$ without fine-tuning.

IV. EXPERIMENTS

In this section, we first detail the experimental setups. Then, we evaluate our proposed boundary labeling model on a benchmark and verify the domain generalization performance on eight datasets. Subsequently, we investigate the impact of different model settings and parameters. Finally, we present a case study of entity boundary detection results from different methods.

A. Experimental Setup

1) Datasets.: We use eight popular datasets to ascertain the effectiveness of METABDRY. **CoNLL03 (CON)** is a well known collection of Reuters newswire articles that contains a large portion of sports news [54]. **OntoNotes5.0 (ONT)** includes text from five different text genres: newswire, magazine, broadcast news, broadcast conversation, web data [55]. **WikiGold (WIK)** is a set of Wikipedia articles [56]. **WNUT2017 (WNU)** is a set of noisy user-generated text including Twitter, Reddit, YouTube comments, and StackExchange posts [57]. **Ritter11 (RIT)** is a randomly sampled set of tweets [58]. **Cadec (CAD)** is a richly annotated corpus of medical forum posts on patient reported Adverse Drug Events [59]. Annotations contain mentions of concepts such as drugs, adverse effects, symptoms and diseases. **Re3d (RE3)** specifically focuses on entity and relationship extraction relevant for somebody operating in the role of a defence and security intelligence analyst [60]. **Sec (SEC)** is a dataset of financial agreements made public through the U.S. Security and Exchange Commission (SEC) filings [61]. Because our task is boundary detection, we ignore entity types in all datasets. The statistics of the datasets are reported in Table I where "Dev" stands for "development set" for model selection. In our experiments, different datasets represent different domains.

For the final test, we perform leave-one-out experiments by picking one domain to hold out as the target domain $\mathcal{D}_{test}$. In each iteration of the meta-validation step, we randomly hold one domain out from $\mathcal{D}_{tr}$ as the meta-validation domain $\mathcal{D}_{val}$. As shown in Table I, $\mathcal{D}_{tr}$ and $\mathcal{D}_{val}$ are constructed from training sets of source domains, $\mathcal{D}_{test}$ is constructed from test sets of target domains. The hyperparameters are tuned on the development set of the target domain (*i.e.,* leave-one-out domain $\mathcal{D}_{test}$).

2) Metrics: Following the previous work [38], we measure entity boundary detection using Precision, Recall, and F1 Scores. Note that these measures are calculated under the "exact-match" rule which means an entity is correctly detected only if its start and end boundaries are both correctly identified. Let g be the total number of entity mentions in the human annotation, h be the total number of entity mentions in the model output, and c be the total number of correctly detected entities in the model output. Then, we measure Precision, Recall, and F1 scores for entity boundary detection performance as follows:

$$\text{Precision} = \frac{c}{h}, \text{Recall} = \frac{c}{g}, \text{and F1 score} = \frac{2c}{g+h} \quad (23)$$

3) Baseline Methods: We evaluate the proposed METABDRY against the following competitors:

- **AGG** - It simply aggregates training data across all source domains and no domain generalization technique is applied during training.
- **ReTrain** - It is first trained on the training sets of source domains, and then retrained on their development sets [37].
- **MTL** - It models different domains as different tasks, which are jointly trained in a multi-task learning manner [17].
- **DANN** - It uses an unsupervised domain adaptation approach with adversarial training [53].
- **DSN** - It is based on the idea of Domain Separation Networks, which explicitly learn to extract representations that are partitioned into two subspaces: one that is private to each domain (development set) and one that is shared across domains (training sets) [62].
- **MLDG** - It is a meta-learning based approach extending MAML [26] to the domain generalization problem [41].
- **AdvTrans** - It is an adversarial transfer learning approach without meta-learning [38]. The original version is for two-domain adaptation, and we re-purpose this method for multi-domain generalization.

4) Implementation Details: For all neural network models, we use GloVe 300-dimensional pre-trained word embeddings[2] released by Stanford, which are fine-tuned during training. The dimension of the character-level representation is 100 and the CNN sliding windows of filters are $[2, 3, 4, 5]$. The total number of CNN filters is 100. For the boundary labeling model, the bidirectional encoder GRU each has a depth of 3 and a hidden size 128. The decoder GRU is unidirectional in our model and is twice the hidden size of the encoder. The inner learning rate α, meta-learning rate β and dropout are tuned on the development set for each experiment. The decay rate is 0.09 and the gradient clip is 5.0. Our proposed METABDRY is implemented with the PyTorch framework[3] and evaluated on NVIDIA Tesla P100 GPUs. Note that METABDRY requires second order optimization partial derivatives. Unfortunately, the double backward for `_cudnn_rnn_backward` has not been implemented in PyTorch (1.0 Version) so far. Thus, we reimplement GRUs from scratch for the second order optimization.

TABLE II
THE PERFORMANCE OF ENTITY BOUNDARY LABELING MODELS ON ONTONOTES5.0. SIGNIFICANT IMPROVEMENT OVER BASELINES IS MARKED WITH * (p-VALUE < 0.05).

Methods	$P(\%)$	$R(\%)$	$F1(\%)$
RegEx	50.26	67.52	57.63
Shallow-CRF	90.75	85.84	88.23
StanfordNER	78.01	77.65	77.83
BiLSTM-MLP	91.12	92.73	91.92
BiLSTM-CRF	91.61	92.82	92.21
Ours	**93.22***	**94.67***	**93.94***

B. Boundary Labeling Performance

We train our boundary labeling model (see Figure 3) on OntoNotes5.0 and compare it with five methods. RegEx is created with regular expressions, based on word surface patterns, *e.g.,* letter cases. Shallow-CRF trains a conditional random field (CRF) using the commonly used token-level features [63]. BiLSTM-MLP/CRF [64] utilizes bidirectional long short-term memory (BiLSTM) to encode a word sequence, and MLP (multilayer perceptron)/CRF to infer decoder tags. The results are summarized in Table II.

First, our boundary labeling model outperforms all baseline methods in terms of P, R and $F1$ scores. More specifically, our model outperforms RegEx, Shallow-CRF, StanfordNER, BiLSTM-MLP, and BiLSTM-CRF by relative F1 improvements of 63.01%, 6.47%, 20.07%, 2.20%, and 1.88%, respectively. Our solution also provides a new perspective to model the sequence labeling task using pointer networks instead of the classic CRF-based approach. Second, the performance of StanfordNER is poor. This observation coincides with our previous claim that off-the-shelf NER systems are not specifically designed for entity boundary detection. Off-the-shelf tools are commonly trained on a single small dataset for boundary detection and typing. Typically, off-the-shelf NER systems do not work well on cross-domain datasets (*e.g.,*

[2]http://nlp.stanford.edu/projects/glove/

[3]https://pytorch.org/, the version 1.0 is used.

TABLE III
THE DOMAIN GENERALIZATION PERFORMANCE ($F1$ SCORE) OF ENTITY BOUNDARY DETECTION. SIGNIFICANT IMPROVEMENTS OVER THE BASELINES ARE MARKED WITH * (p-VALUE < 0.05).

Methods	Target Domains								
	CON	ONT	WIK	WNU	RIT	CAD	RE3	SEC	AVE.
AGG	80.38	77.2	78.55	52.68	69.84	56.37	33.71	20.86	58.69
ReTrain [37]	78.29	75.42	76.25	51.01	68.34	57.09	31.65	18.33	57.04
MTL [17]	81.92	77.31	79.45	53.87	70.97	57.34	32.74	21.79	59.42
DANN [53]	83.41	79.58	80.34	54.17	71.45	56.23	34.97	22.78	60.36
DSN [62]	82.55	78.92	80.77	52.93	69.03	57.21	34.26	21.84	59.68
MLDG [41]	84.94	79.32	81.43	54.62	71.84	58.09	35.04	23.96	61.15
AdvTrans [38]	84.14	80.25	82.12	53.37	72.06	58.62	34.71	24.34	61.20
METABDRY (ours)	**86.02***	**81.74***	**83.83***	**56.59***	**73.81***	**59.54***	**36.33***	**26.59***	**63.06***

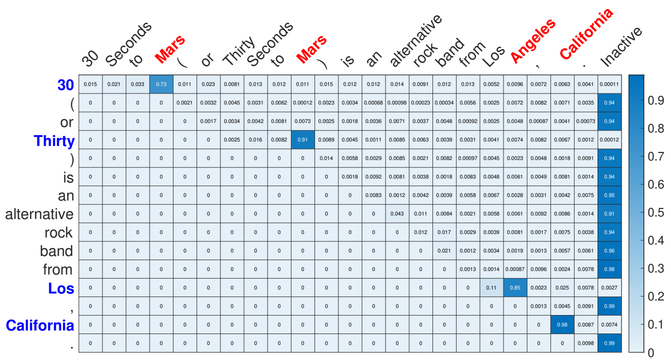


Fig. 4. Visualization of pointer attention weights. The identified start and end positions of entities are in blue and red, respectively. For example, given the input word "or", METABDRY judges it as the non-start (*i.e.,* Inactive) of an entity. Given the input word "Thirty", the end boundary "Mars" is detected based on the pointer attention weights.

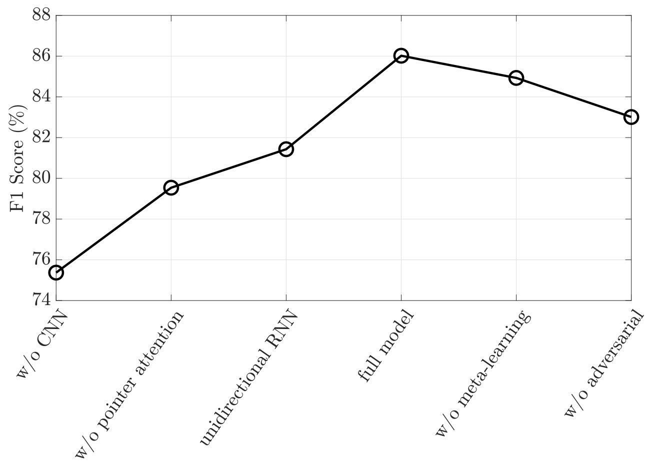


Fig. 5. Impact of architectural choices.

OntoNotes5.0 in this experiment) because they do not take into account domain generalization.

For better understanding, we visualize pointer attention weights with an example in Figure 4. The words on the y-axis are the input of the GRU decoder. The words on the x-axis are the input of the GRU encoder. METABDRY detects entity start and end boundaries in a greedy manner. For example, for the input "*30*", our approach detects the end boundary of this entity at the position "*Mars*". For a given input "*or*", our approach determines that it is not the start of an entity mention by the sentinel word "*Inactive*". Note that the middle words of a detected entity (*e.g.,* "*Seconds*" and "*to*" of "*Thirty Seconds to Mars*") are no longer passed into the decoder GRU. Meanwhile, the pointers in METABDRY are backward. This means that the weights before the current input are zeros. Thus, the attention weights are composed of an upper triangular matrix. Observe that the identified boundaries have dominant attention weights, which implies that our model can successfully learn syntactic features from sentences for entity boundary detection.

C. Domain Generalization Performance

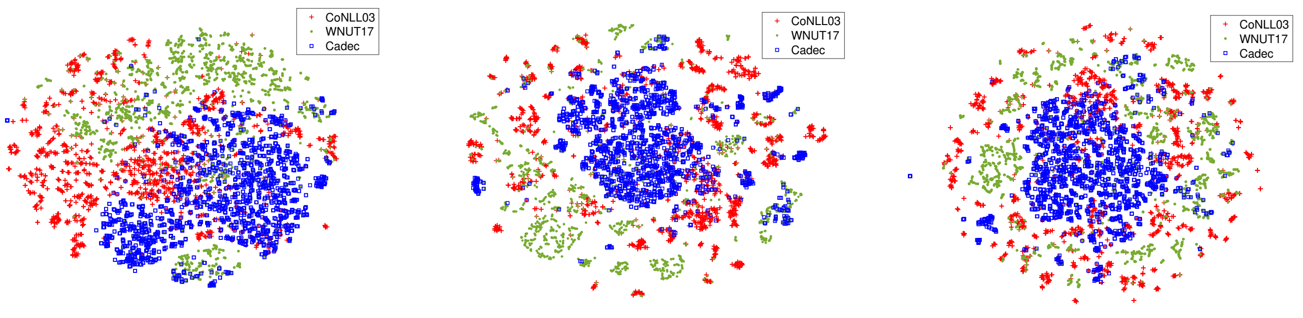
Table III reports the domain generalization performance of different methods under the leave-one-out settings (see IV-A1). We make the following observations:

First, our METABDRY method outperforms all baseline methods on all domain generalization tasks in terms of $F1$ scores, demonstrating that meta-learning is effective for augmenting model generalizability. More specifically, METABDRY outperforms AGG, ReTrain, MTL, DANN, DSN, MLDG and AdvTrans with relative improvements of 7.45%, 10.55%, 6.13%, 4.47%, 5.66%, 3.12% and 3.04% in terms of the averaged $F1$ scores across all domain generalization tasks.

Second, an interesting observation is that AGG, ReTrain, MTL and DSN obtain comparable performance. This reveals that both the naive strategies of aggregation and retraining cannot alleviate the difference in data distributions among multiple domains. This result also demonstrates the necessity of addressing the domain-shift issue when multiple-domain resources are available during training.

Third, the baseline methods using adversarial training (*i.e.,* DANN and AdvTrans) and meta-learning (*i.e.,* MLDG) outperform other baseline methods (AGG, ReTrain, MTL and DSN). This indirectly demonstrates the effectiveness of adversarial training and meta-learning strategies in domain generalization. METABDRY seamlessly incorporates these two strategies and significantly outperforms all baseline methods.

Fourth, named entity boundary detection on informal text (*i.e.,* WNU, RIT, CAD, RE3 and SEC) is much more difficult than on formal text (*i.e.,* CON, ONT and WIK). More specif-



(a) Without adversarial training (b) Without meta-learning (c) METABDRY

Fig. 6. The effect of adversarial training and meta-learning on the distribution of representations from BiGRU encoders.

ically, compared to the other six target domains, all methods obtain relatively low $F1$ scores on RE3 (defence and security intelligence text) and SEC (finance text). This observation shows that the heterogeneous domain generalization is a very challenging task when there is a large distribution gap between the source and target domains. However, METABDRY is still more effective than the baselines in reducing the distribution divergence. We attribute this to the fact that METABDRY explicitly simulates the domain shift during training so that it is conducive to generalize to a novel target domain.

D. Further Analysis

In this subsection, we first study various architectural choices on model performance and the impact of key parameter settings. Then we present a qualitative analysis to intuitively show results from different methods.

1) Ablation Study: Figure 5 reports an ablation analysis on the test set of CON. The full model is our proposed METABDRY. There are five variations: (1) we remove the CNN layer (w/o CNN); (2) remove the pointer attention mechanism and use the last hidden state of the encoder GRU (w/o pointer attention); (3) use unidirectional GRUs instead of bidirectional GRUs (unidirectional RNN); (4) remove the meta-learning strategy (w/o meta-learning); (5) remove the adversarial strategy (w/o adversarial). The ablation study clearly showcases the importance of each component of METABDRY. First, we observe that the character-level representations extracted from CNNs play an important role in domain generalization for entity boundary detection. This is because that the character-level representations can provide additional information to handle out-of-vocabulary (OOV) words in domain generalization where OOV is an extremely common phenomenon. Second, the pointer and bidirectional RNN mechanisms also significantly contribute to identify entity boundaries. This is because they provide more richer context information for seasoning boundaries. Third, the adversarial training and meta-learning strategies have positive effects on domain generalization for entity boundary detection.

We claim that METABDRY incorporates adversarial training and meta-learning strategies to encourage a domain generalizer. Furthermore, we visualize the effect of adversarial training and meta-learning. We randomly sample 2000 training instances for each domain (CoNLL03, WNUT17 and Cadec) and visualize the internal representations (extracted from BiGRU encoders) by t-SNE [65]. Figure 6(a) and 6(b) illustrate the distributions of the internal representations without adversarial training and meta-learning, respectively. Figure 6(c) provides a visualization of our full architecture (METABDRY). The domain discrepancies are large among these three domains because CoNLL03 is from the text genre of the newswire, WNUT17 is from the social media and Cadec is from the medical text. We observe that the three clusters in Figure 6(a) and 6(b) are much more discriminative than Figure 6(c). Our METABDRY approach possessing adversarial training and meta-learning mechanisms, brings the three distributions much closer and more indistinguishable. This ablation study visually demonstrates the effectiveness of our model design for encouraging domain-invariant representations.

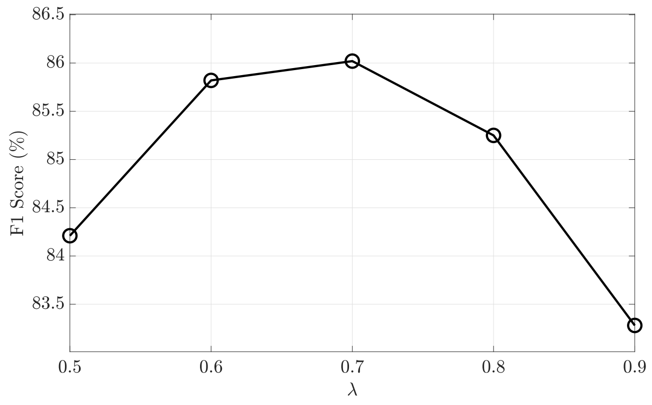


Fig. 7. Impact of parameter λ.

2) Parameter Sensitivity Study: As shown in Equation (14), the parameter λ is an important parameter to control the two objectives: the domain discriminator loss and tag decoder loss. Figure 7 shows an empirical study on parameter sensitivity for λ. We vary the λ as 0.5, 0.6, 0.7, 0.8 and 0.9. Figure 7 plots the F1 scores on the test set of CON across different values of λ. We observe that $\lambda = 0.7$ yields the best empirical performance. This parameter sensitivity analysis shows that we need to balance the two learning objectives with optimal λ for better domain generalization.

TABLE IV
CASE STUDY OF ENTITY BOUNDARY DETECTION RESULTS FROM DIFFERENT METHODS. THE GROUND-TRUTH ENTITIES IN ANNOTATED CORPORA ARE UNDERLINED. THE WRONGLY DETECTED ENTITIES ARE HIGHLIGHTED IN RED.

	Case 1 from CON: *Hosts UAE play Kuwait and South Korea take on Indonesia on Saturday in Group A matches.*	Case 2 from WNU: *Why were Olive and Emma's powers changed in Miss Peregrine's Home for Peculiar Children?*
AGG	Kuwait, South Korea, Indonesia	Olive, Emma
MTL	UAE, South Korea, Indonesia	Olive, Emma, Peregrine
MLDG	Kuwait, South Korea, Indonesia, Group	Olive, Emma, Miss Peregrine's Home
AdvTrans	Kuwait, South Korea, Indonesia, Group A	Olive, Emma, Miss Peregrine's Home, Peculiar Children
METABDRY	UAE, Kuwait, South Korea, Indonesia	Olive, Emma, Miss Peregrine's Home for Peculiar Children
	Case 3 from RIT: *OMG! Miley Cyrus Has A 14-Year-Old Stalker!: First Paris Hilton, now Miley Cyrus!*	Case 4 from CAD: *Family history of Heart desease, diabetes, and high blood pressure*
AGG	Miley Cyrus, Miley Cyrus	Heart
MTL	Miley Cyrus, 14-Year-Old Stalker, Miley Cyrus	Heart
MLDG	Miley Cyrus, 14-Year-Old Stalker, Paris Hilton, Miley Cyrus	Family history of Heart
AdvTrans	Miley Cyrus, First Paris Hilton, Miley Cyrus	Heart desease
METABDRY	Miley Cyrus, Paris Hilton, Miley Cyrus	Heart desease, high blood pressure
	Case 5 from RE3: *Designs were completed by the British firm Voganlei and Coode.*	Case 6 from SEC: *1] - Union Bank of California NA and Crocs Inc. [UNION BANK OF CALIFORNIA LOGO] AMENDMENT NO.*
AGG	British	California, CALIFORNIA
MTL	British, Voganlei, Coode	California, Crocs Inc., CALIFORNIA
MLDG	British, Voganlei and Coode	California, Crocs Inc., CALIFORNIA
AdvTrans	British, Voganlei and Coode	Union Bank of California, Crocs Inc., UNION BANK OF CALIFORNIA LOGO
METABDRY	British firm Voganlei and Coode	Union Bank of California, UNION BANK OF CALIFORNIA LOGO

3) Time Consuming Analysis: As shown in Algorithm 1, METABDRY consists of two core phases: training and testing phases. Now we empirically investigate the time consumption of these two phases on one single GPU. In particular, the outer step size is set to 8 and the inner step size is set to 1. During training, two samples per each domain are sampled to construct a mini-batch, resulting in the batch size of 12. Our experimental study shows that each inner loop (*i.e.,* "j" loop) needs 0.33 second, and the outer loops (*i.e.,* "i" loop) totally need 2.64 seconds. However, each iteration (*i.e.,* "while" loop) during training takes 6.2 seconds. Thus, the main time consumption (*i.e.,* $(6.2 - 2.64)$ seconds) lies in second order optimization partial derivatives (*i.e.,* line 18 in Algorithm 1) and the parameters update (*i.e.,* line 20 in Algorithm 1).

For the final test, the learned parameters (*i.e.,* θ_{Meta} and ϕ_{Meta}) are fixed and directly evaluated on target domains. The batch size of test is also set to 12. Each iteration with 12 samples during testing takes 0.064 second, which is much faster than training. Note that the time consumption is linearly proportional to the data size for both training and testing phases. For example, METABDRY totally takes 18.432 seconds when testing on CON domain (3454 samples).

4) Qualitative Study: For intuitive comparisons, we conduct a qualitative study to investigate the prediction results from different models. We only show six cases from CON, WNU, RIT, CAD, RE3, and SEC across five methods (AGG, MTL, MLDG, AdvTrans and METABDRY) in Table IV, because the text genres of ONT, WIK are similar to CON and the performance of ReTrain is comparable to AGG, the performance of DSN and MLDG is comparable to AdvTrans. For the case 1 - 3, only METABDRY can correctly detect all entity boundaries while other baseline methods cannot. For the case 4 - 6, all methods (including our METABDRY) have wrongly detected results.

From Table IV, we have the following observations: First, compared with the baseline methods, METABDRY misses fewer entities. Totally, METABDRY misses one entity in the case 4 ("*diabetes*"). In particular, the errors from the AGG and MTL methods mainly lie in missing some entities. For example, the AGG method misses one entity in the case 1, one entity in the case 2, one entity in the case 3 and two entities in the case 4.

Second, for all methods, the case capitalization may mislead boundary detection. For example, "*Group A*" and "*First Paris Hilton*" are wrongly detected by AdvTrans. "*Group*", "*14-Year-Old Stalker*" and "*Family history of Heart*" are wrongly detected by MLDG. "*UNION BANK OF CALIFORNIA LOGO*" is wrongly detected by METABDRY. Usually, the formal text (*e.g.,* CON and WIK) has strict criteria for case sensitivity while the informal text has not. For example, the case 4 from social media, and case 6 from the finance text are both written with capital letters. Thus, domain generalization for informal text is more challenging because the informal text is complicated and noisy. However, METABDRY significantly outperforms the baselines. For example, METABDRY correctly detects all entities in the cases 2 and 3 both from social media.

Third, METABDRY can detect boundaries of long entities while the baseline methods break a long entity into parts and lose the entity integrity. For example, none of baseline methods can correctly detect the long entity "*Miss Peregrine's home for Peculiar Children*" in the case 2 while our approach METABDRY can perfectly identify boundaries. We attribute this to the fact that our approach is more general and robust to reduce the domain discrepancy.

Fourth, the annotation criteria among different domains is the main issue for domain generalization. There may be an annotation conflict for the same type of entities. A typical example is that whether the definite article "*the*" should be included in entities. For example, "*the British firm Voganlei and Coode*" from the case 5 include the definite article while most of the source domains do not include it. In the case 6, "*Union Bank of California*" is annotated with two different entities: "*Union Bank*" and "*California*". This annotation also makes unsupervised domain generalization more challenging. An effective approach to alleviating this issue is to fine-tune METABDRY with a small amount of annotated target domain data. In this work, we assume no access to any target domain information and we leave the supervised domain generalization problem as a future work. Although METABDRY is designed for domain generalization, we do not claim that it can handle all cases in the real world.

V. CONCLUSION

In this paper, we studied a very ambitious problem in NLP, *i.e.,* the task of named entity boundary detection under domain generalization settings. We first proposed a novel boundary labeling model, utilizing pointer networks to effectively address the issue of boundary tag sparsity and the issue of variable size output vocabulary. We then proposed a domain generalization approach for named entity boundary detection, which incorporates adversarial learning and meta-learning to encourage a general and robust model. Our work is the first attempt in adapting meta-learning to aggregate meta-knowledge from multiple resource domains for sequence labeling. We first conducted experiments on a benchmark dataset to evaluate the effectiveness of our novel boundary labeling model. We then conducted extensive experiments on eight popular datasets, verifying the correctness and effectiveness of our proposed approach.

We believe that named entity boundary detection is a fundamental problem in NLP. Our robust boundary detection model will benefit a wide variety of downstream applications. In the future, we would like to explore the applications of the detected entity mentions in entity linking, fine-grained entity typing, sentiment analysis, question answering and text summarization. In addition, although this study assumes no access to any target information, our meta-learned domain generalizer can be further fine-tuned (either fast weights or base weights) in a supervised manner with few target samples. We leave this as the future work for performance improvements. Lastly, we are also interested in boosting our approach with recent pre-trained language models such as ELMo and BERT.

REFERENCES

[1] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[2] B. Babych and A. Hartley, "Improving machine translation quality with automatic named entity recognition," in *EAMT*, 2003, pp. 1–8.

[3] A. Abhishek, A. Anand, and A. Awekar, "Fine-grained entity type classification by jointly learning representations and label embeddings," in *EACL*, 2017, pp. 797–807.

[4] L. d. Corro, A. Abujabal, R. Gemulla, and G. Weikum, "Finet: Context-aware fine-grained named entity typing," in *EMNLP*, 2015, pp. 868–878.

[5] X. Ren, W. He, M. Qu, L. Huang, H. Ji, and J. Han, "Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding," in *EMNLP*, 2016, pp. 1369–1378.

[6] A. Lal, A. Tomer, and C. R. Chowdary, "Sane: System for fine grained named entity typing on textual data," in *WWW*, 2017, pp. 227–230.

[7] E. Choi, O. Levy, Y. Choi, and L. Zettlemoyer, "Ultra-fine entity typing," in *ACL*, 2018, pp. 87–96.

[8] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," in *ACL*, 2016, pp. 1064–1074.

[9] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *NAACL*, 2016, pp. 260–270.

[10] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[11] S. Zheng, F. Wang, H. Bao, Y. Hao, P. Zhou, and B. Xu, "Joint extraction of entities and relations based on a novel tagging scheme," in *ACL*, 2017, pp. 1227–1236.

[12] Y. Shen, H. Yun, Z. C. Lipton, Y. Kronrod, and A. Anandkumar, "Deep active learning for named entity recognition," in *ICLR*, 2017.

[13] S. Fisher and B. Roark, "The utility of parse-derived features for automatic discourse segmentation," in *ACL*, 2007.

[14] S. Joty, G. Carenini, and R. T. Ng, "Codra: A novel discriminative framework for rhetorical analysis," *Comput. Linguist.*, vol. 41, no. 3, pp. 385–435, 2015.

[15] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *NIPS*, 2015, pp. 2692–2700.

[16] J. Li, A. Sun, and S. Joty, "Segbot: A generic neural text segmentation model with pointer network," in *IJCAI*, 2018, pp. 140–147.

[17] Z. Yang, R. Salakhutdinov, and W. W. Cohen, "Transfer learning for sequence tagging with hierarchical recurrent networks," in *ICLR*, 2017.

[18] P. von Däniken and M. Cieliebak, "Transfer learning and sentence level features for named entity recognition on tweets," in *W-NUT*, 2017, pp. 166–171.

[19] H. Zhao, Y. Yang, Q. Zhang, and L. Si, "Improve neural entity recognition via multi-task data selection and constrained decoding," in *NAACL-HLT*, vol. 2, 2018, pp. 346–351.

[20] B. Y. Lin and W. Lu, "Neural adaptation layers for cross-domain named entity recognition," in *EMNLP*, 2018, pp. 2012–2022.

[21] J. T. Zhou, H. Zhang, D. Jin, H. Zhu, M. Fang, R. S. M. Goh, and K. Kwok, "Dual adversarial neural transfer for low-resource named entity recognition," in *ACL*, 2019, pp. 3461–3471.

[22] S. Francis, J. V. Landeghem, and M. Moens, "Transfer learning for named entity recognition in financial and biomedical documents," *Information*, vol. 10, no. 8, p. 248, 2019.

[23] K. Akuzawa, Y. Iwasawa, and Y. Matsuo, "Adversarial invariant feature learning with accuracy constraint for domain generalization," *CoRR*, vol. abs/1904.12543, 2019.

[24] J. Schmidhuber, "On learning how to learn learning strategies," 1995.

[25] S. Thrun and L. Y. Pratt, Eds., *Learning to Learn*. Springer, 1998. [Online]. Available: https://doi.org/10.1007/978-1-4615-5529-2

[26] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017, pp. 1126–1135.

[27] Y. Li, Y. Yang, W. Zhou, and T. M. Hospedales, "Feature-critic networks for heterogeneous domain generalization," in *ICML*, 2019, pp. 3915–3924.

[28] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[29] E. Strubell, P. Verga, D. Belanger, and A. McCallum, "Fast and accurate entity recognition with iterated dilated convolutions," in *ACL*, 2017, pp. 2670–2680.

[30] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *NAACL-HLT*, 2018, pp. 2227–2237.

[31] J. Li, Z. Xing, and A. Kabir, "Leveraging official content and social context to recommend software documentation," *IEEE Transactions on Services Computing*, 2018.

[32] P.-H. Li, R.-P. Dong, Y.-S. Wang, J.-C. Chou, and W.-Y. Ma, "Leveraging linguistic structures for named entity recognition with bidirectional recursive neural networks," in *EMNLP*, 2017, pp. 2664–2669.

[33] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
[34] J. Zhuo, Y. Cao, J. Zhu, B. Zhang, and Z. Nie, “Segment-level sequence modeling using gated recursive semi-markov conditional random fields,” in *ACL*, vol. 1, 2016, pp. 1413–1423.
[35] S. J. Pan, Z. Toh, and J. Su, “Transfer joint embedding for cross-domain named entity recognition,” *ACM Trans. Inf. Syst.*, vol. 31, no. 2, p. 7, 2013.
[36] C. Jia, L. Xiao, and Y. Zhang, “Cross-domain NER using cross-domain language modeling,” in *ACL*, 2019, pp. 2464–2474.
[37] J. Y. Lee, F. Dernoncourt, and P. Szolovits, “Transfer learning for named-entity recognition with neural networks,” in *LREC*, 2017, pp. 4470–4473.
[38] J. Li, D. Ye, and S. Shang, “Adversarial transfer for named entity boundary detection with pointer networks,” in *IJCAI*, 2019, pp. 5053–5059.
[39] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, “Matching networks for one shot learning,” in *NIPS*, 2016, pp. 3630–3638.
[40] S. Ravi and H. Larochelle, “Optimization as a model for few-shot learning,” in *ICLR*, 2017. [Online]. Available: https://openreview.net/forum?id=rJY0-Kcll
[41] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales, “Learning to generalize: Meta-learning for domain generalization,” in *AAAI*, 2018, pp. 3490–3497.
[42] Y. Balaji, S. Sankaranarayanan, and R. Chellappa, “Metareg: Towards domain generalization using meta-regularization,” in *NIPS*, 2018, pp. 1006–1016.
[43] D. Li, J. Zhang, Y. Yang, C. Liu, Y.-Z. Song, and T. M. Hospedales, “Episodic training for domain generalization,” in *ICCV*, 2019, pp. 1446–1455.
[44] J. Gu, Y. Wang, Y. Chen, V. O. K. Li, and K. Cho, “Meta-learning for low-resource neural machine translation,” in *EMNLP*, 2018, pp. 3622–3631.
[45] P. Huang, C. Wang, R. Singh, W. Yih, and X. He, “Natural language to structured query generation via meta-learning,” in *NAACL-HLT*, 2018, pp. 732–738.
[46] K. Qian and Z. Yu, “Domain adaptive dialog generation via meta learning,” in *ACL*, 2019, pp. 2639–2649.
[47] Z. Lin, A. Madotto, C.-S. Wu, and P. Fung, “Personalizing dialogue agents via meta-learning,” in *ACL*, 2019, pp. 5454–5459.
[48] A. Obamuyide and A. Vlachos, “Model-agnostic meta-learning for relation classification with limited supervision,” in *ACL*, 2019, pp. 5873–5879.
[49] Z. Hu, T. Chen, K. Chang, and Y. Sun, “Few-shot representation learning for out-of-vocabulary words,” in *ACL*, 2019, pp. 4102–4112.
[50] J. Li, B. Chiu, S. Shang, and L. Shao, “Neural text segmentation and its application to sentiment analysis,” *IEEE Transactions on Knowledge and Data Engineering*, 2020.
[51] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[52] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches,” in *SSST-8*, 2014, pp. 103–111.
[53] Y. Ganin and V. S. Lempitsky, “Unsupervised domain adaptation by backpropagation,” in *ICML*, 2015, pp. 1180–1189.
[54] E. F. Tjong Kim Sang and F. De Meulder, “Introduction to the conll-2003 shared task: Language-independent named entity recognition,” in *NAACL-HLT*, 2003, pp. 142–147.
[55] S. Pradhan, A. Moschitti, N. Xue, O. Uryupina, and Y. Zhang, “Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes,” in *EMNLP*, 2012, pp. 1–40.
[56] D. Balasuriya, N. Ringland, J. Nothman, T. Murphy, and J. R. Curran, “Named entity recognition in wikipedia,” in *ACL-IJCNLP*, 2009, pp. 10–18.
[57] L. Derczynski, E. Nichols, M. van Erp, and N. Limsopatham, “Results of the wnut2017 shared task on novel and emerging entity recognition,” in *W-NUT*, 2017, pp. 140–147.
[58] A. Ritter, S. Clark, O. Etzioni *et al.*, “Named entity recognition in tweets: an experimental study,” in *EMNLP*, 2011, pp. 1524–1534.
[59] S. Karimi, A. Metke-Jimenez, M. Kemp, and C. Wang, “Cadec: A corpus of adverse drug event annotations,” *J. Biomed. Inform.*, vol. 55, pp. 73–81, 2015.
[60] Re3d, “https://github.com/dstl/re3d.”
[61] J. C. S. Alvarado, K. Verspoor, and T. Baldwin, “Domain adaption of named entity recognition to support credit risk assessment,” in *ALTA*, 2015, pp. 84–90.
[62] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, “Domain separation networks,” in *NIPS*, 2016, pp. 343–351.
[63] W. Liao and S. Veeramachaneni, “A simple semi-supervised algorithm for named entity recognition,” in *NAACL-HLT*, 2009, pp. 58–65.
[64] J. P. Chiu and E. Nichols, “Named entity recognition with bidirectional lstm-cnns,” in *TACL*, vol. 4, 2016, pp. 357–370.
[65] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne,” *J. Mach. Learn. Res.*

```
@article{li21domaingen,
author    = {Jing Li and Shuo Shang and Lisi Chen},
title     = {Domain Generalization for Named Entity Boundary Detection via Metalearning},
journal   = {IEEE Transactions on Neural Networks and Learning Systems (TNNLS)},
volume    = {32},
number    = {9},
pages     = {3819--3830},
year      = {2021},
url       = {https://doi.org/10.1109/TNNLS.2020.3015912},
doi       = {10.1109/TNNLS.2020.3015912},
}
```