Adversarial Transfer for Named Entity Boundary Detection with Pointer Networks

Jing Li[1] , **Deheng Ye**[2] and **Shuo Shang**[1*]
[1]Inception Institute of Artificial Intelligence, Abu Dhabi, United Arab Emirates
[2]Tencent AI Lab, Shenzhen, China
jingli.phd@hotmail.com, dericye@tencent.com, jedi.shang@gmail.com

Abstract

In this paper, we focus on *named entity boundary detection*, which aims to detect the start and end boundaries of an entity mention in text, without predicting its type. A more accurate and robust detection approach is desired to alleviate error propagation in downstream applications, such as entity linking and fine-grained typing systems. Here, we first develop a novel entity boundary labeling approach with pointer networks, where the output dictionary size depends on the input, which is variable. Furthermore, we propose AT-BDRY, which incorporates adversarial transfer learning into an end-to-end sequence labeling model to encourage domain-invariant representations. More importantly, AT-BDRY can reduce domain difference in data distributions between the source and target domains, via an unsupervised transfer learning approach (*i.e.,* no annotated target-domain data is necessary). We conduct Formal Text $\rightarrow$ Formal Text, Formal Text $\rightarrow$ Informal Text and ablation evaluations on five benchmark datasets. Experimental results show that AT-BDRY achieves state-of-the-art transferring performance against recent baselines.

1 Introduction

Named entity recognition (NER) is a fundamental task in natural language processing which aims to jointly resolve the boundaries and type of a named entity in text [Li *et al.*, 2018a]. In this paper, we ignore the entity typing and focus on the subtask of *named entity boundary detection*, which involves detecting the start and end boundaries of an entity mention in text.

There are several reasons for studying the subtask. *First*, most fine-grained entity typing systems, such as FIGER [Abhishek *et al.*, 2017], FINET [Corro *et al.*, 2015] and SANE [Lal *et al.*, 2017], either manually label entity mentions or assume that entity mentions have already been predetected [Corro *et al.*, 2015]. In this case, the typing task becomes a multi-label classification task. Errors made in entity boundary detection inevitably mislead and adversely affect subsequent entity-typing systems. This has created an overwhelming demand for more accurate and robust boundary detection approaches. *Second*, the availability of knowledge bases, *e.g.,* Wikipedia, FreeBase, ProBase, enables the study of entity linking, which aims to determine the identity of entities mentioned in text. Because of the overwhelming entity types defined in knowledge bases, the types predicted by NER systems become less necessary. Further, there could be conflicts between the types predicted by NER systems and the types of the entities disambiguated through the entity linking process [Phan *et al.*, 2018].

Some studies [Ren *et al.*, 2016; Corro *et al.*, 2015] utilize off-the-shelf NER systems (*e.g.,* StanfordNER and Spotlight3) to detect entity boundaries for downstream applications. However, off-the-shelf systems suffer from the shift in data distribution from the actual data encountered at test time, resulting in poor performance. Recently, a few studies [Yang *et al.*, 2017; von Däniken and Cieliebak, 2017; Zhao *et al.*, 2018; Lin and Lu, 2018] have investigated *transfer learning* in NER, aiming to transfer knowledge from source domain to target domain. However, these existing methods require an amount of annotated target-domain data to fine-tune models. In this paper, we assume no label annotations in the target domain.

Essentially, entity boundary detection is a sequence labeling problem, where the task is to predict a sequence of 'yes/no' boundary tags at word level in a sentence. Existing labeling techniques can be broadly categorized into two paradigms: recurrent neural networks with conditional random fields (RNN-CRF) [Ma and Hovy, 2016; Lample *et al.*, 2016; Huang *et al.*, 2015] and RNN as a language model to generate the output sequence (seq2seq) [Zheng *et al.*, 2017; Shen *et al.*, 2017]. However, studies [Vinyals *et al.*, 2015; Li *et al.*, 2018b] indicate that they 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). Here, we seek a new sequence labeling approach to alleviate these two issues.

In this paper, we propose AT-BDRY, an adversarial transfer approach for named entity boundary detection with pointer networks. First, we integrate the pointer mechanism [Vinyals *et al.*, 2015] into a sequence labeling framework, which can effectively handle variable size vocabulary in the output to

*Corresponding author.

produce entity boundaries depending on the input sequence. Second, we introduce adversarial learning with a domain discriminator (adversary), resulting in the emergence of extracted features that are (i) discriminative for sequence labeling on the source domain and (ii) indistinguishable with respect to the shift between source and target domains. It is worth noting that the source-domain data is annotated and target-domain data is unannotated (unsupervised transfer).

In summary, we make three contributions:

- We developed a novel boundary labeling approach based on pointer networks, which infers entity boundaries from a variable input sequence.
- We proposed AT-BDRY, which incorporates adversarial learning into an end-to-end sequence labeling model to encourage domain-invariant representations.
- We conducted Formal Text $\rightarrow$ Formal Text, Formal Text $\rightarrow$ Informal Text and ablation evaluations on five datasets. Experimental results show that AT-BDRY achieves state-of-the-art transferring performance against recent baselines.

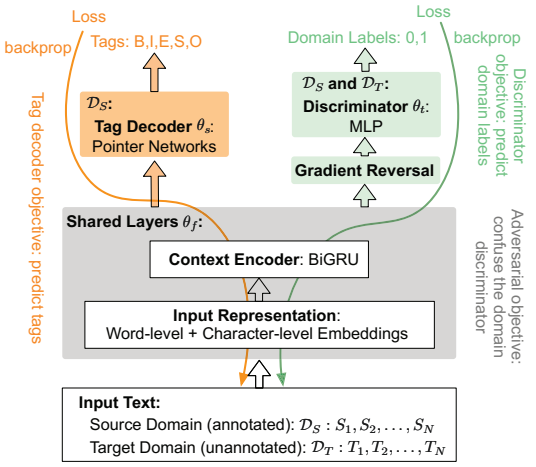


Figure 1: The architecture of AT-BDRY. The left part learns a tag decoder for entity boundary detection based on the source domain data, which is then used for the target domain at test time. The right part makes the distributions of source ($\mathcal{D}_S$) and target ($\mathcal{D}_T$) domains indistinguishable through a domain discriminator.

2 Related Work

2.1 Named Entity Recognition

There are three common paradigms for NER: *knowledge-based unsupervised* systems, *feature-based supervised* systems and *neural-based* systems. Nadeau and Sekine [2007] summarized knowledge-based unsupervised and feature-based supervised systems in detail. Here, we focus on describing neural-based systems.

Many neural architectures have been applied to NER and achieve state-of-the-art results. *Neural-based* systems have the advantage of inferring latent features and learning labels in an end-to-end fashion. The use of neural model for NER was pioneered by [Collobert *et al.*, 2011], where an architecture based on temporal convolutional neural networks (CNNs) over word sequence was proposed. Recent neural models for NER can be broadly classified into three dimensions by their representation of words in sentence and architectures of tag decoders [Li *et al.*, 2018a].

Compared with existing architectures, the main difference in our proposed AT-BDRY is that our tag decoder is a pointer network, not CRF. Our model effectively captures sequential dependencies when boundary tags are sparse, while alleviating variable size vocabulary in the output to produce entity boundaries depending on the input sequence. In addition, Zhai *et al.* [2017] proposed a model for sequence chunking based on pointer networks, which is most related to our work. Different from Zhai's model, our proposed model directly infers the start and end boundaries of an entity, which leads to a simpler architecture with fewer parameters.

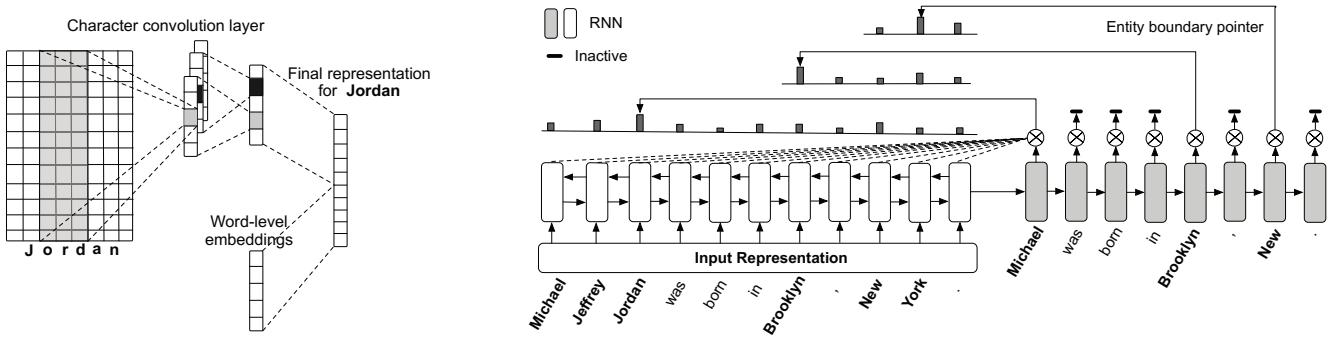
2.2 Transfer Learning in NER

Transfer learning aims to perform a machine learning task on a target domain by taking advantage of knowledge learned from a source domain [Pan *et al.*, 2013]. In the setting of transfer learning, different neural models commonly share different parts of model parameters between source task and target task. Yang *et al.* [2017] first investigated the transferability of different layers of representations. Then, they presented three different parameter-sharing architectures for cross-domain, cross-lingual, and cross-application scenarios. Pius and Mark [2017] extended Yang's approach to allow joint training on informal corpus and incorporate sentence-level feature representation. Zhao *et al.* [2018] proposed a multi-task model with domain adaption, where the fully-connected layers are adapted to different datasets, and the CRF features are computed separately. Different from these parameter-sharing architectures, Lee *et al.* [2017] applied transfer learning in NER by training a model on source task and using the trained model on target task for fine-tuning. Recently, Lin and Lu [2018] also proposed a fine-tuning approach for NER by introducing three neural adaptation layers: word adaptation layer, sentence adaptation layer, and output adaptation layer. Cao *et al.* [2018] proposed a model for adversarial multi-task learning with a cotraining manner.

Unlike these methods (require an amount of annotated data in target domain), our approach assumes no label annotations in the target domain. Moreover, our approach incorporates adversarial learning into a sequence labeling model to encourage domain-invariant representations.

3 AT-BDRY: Adversarial Transfer for Named Entity Boundary Detection

Our approach, termed AT-BDRY, consists of four key components as shown in Figure 1. The input representation layer represents each word with word-level and character-level embeddings. The context encoder aims to extract contextual representations for input sentences. The tag decoder uses a pointer mechanism to detect named entity boundaries based on intermediate representations by the context encoder. The domain discriminator is to judge whether a training instance belongs to source or target domains. In particular, we incorporate adversarial training in the shared space to ensure that



(a) Representation of an input word "*Jordan*". (b) The architecture of pointer networks for entity boundary detection.

Figure 2: 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*".

the intermediate representations by the context encoder contain no discriminative information about the origin of input (source or target). That is, we hope 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 judge the domain class of each example.

3.1 Input Representation

The input representation in our model consists of character-level and word-level representations. Previous studies [Chiu and Nichols, 2016; Huang *et al.*, 2015] have shown that character-level information (*e.g.,* prefix and suffix of a word) is an effective resource for NER task. Two kinds of network structures have been used to extract character-level representation, *i.e.,* CNN and BiLSTM. In our model, we use CNN because of its lower computational cost. Our design is similar to [Chiu and Nichols, 2016], and the main difference is that we use a sliding convolution layer (*i.e.,* variable window size of convolution filters) to capture character n-grams in a word, shown in Figure 2(a).

We assume that AT-BDRY works with the annotated data $\mathcal{D}_S : S_1, S_2, ..., S_N$ in the source domain , and the unannotated data $\mathcal{D}_T : T_1, T_2, ..., T_M$ in the target domain. Given an input sentence $\boldsymbol{W} = (W_1, W_2, \ldots, W_L)$ of length L, $\boldsymbol{W} \in \{\mathcal{D}_S, \mathcal{D}_T\}$, let W_l denote its l-th word. The character-level representation and word-level embedding (*e.g.,* pre-trained embedding) for W_l are concatenated as its final representation, $\boldsymbol{x}_l \in \mathbb{R}^K$, where K represents the dimension of $\boldsymbol{x}_l$, see Figure 2(a). Note that hand-crafted features can be easily integrated into this architecture. However, we do not use any hand-crafted features in this study.

3.2 Context Encoder

We encode the input sequence $\mathbf{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_L)$ using an RNN. RNNs capture sequential dependencies. With hidden cells like long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997] and gated recurrent unit (GRU) [Cho *et al.*, 2014], 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:

$$\boldsymbol{z}_l = \sigma(\boldsymbol{U}_z \boldsymbol{x}_l + \boldsymbol{R}_z \boldsymbol{h}_{l-1} + \boldsymbol{b}_z) \quad (1)$$

$$\boldsymbol{r}_l = \sigma(\boldsymbol{U}_r \boldsymbol{x}_l + \boldsymbol{R}_r \boldsymbol{h}_{l-1} + \boldsymbol{b}_r) \quad (2)$$

$$\boldsymbol{n}_l = \tanh(\boldsymbol{U}_h \boldsymbol{x}_l + \boldsymbol{R}_h (\boldsymbol{r}_l \odot \boldsymbol{h}_{l-1}) + \boldsymbol{b}_h) \quad (3)$$

$$\boldsymbol{h}_l = \boldsymbol{z}_l \odot \boldsymbol{h}_{l-1} + (1 - \boldsymbol{z}_l) \odot \boldsymbol{y}_l \quad (4)$$

where $\sigma()$ is the sigmoid function, $\tanh()$ 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 the parameters of the encoder that need to be learnt.

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:

$$\boldsymbol{h}_l = \overrightarrow{\boldsymbol{h}}_l \oplus \overleftarrow{\boldsymbol{h}}_l \quad (5)$$

where $\oplus$ indicates a concatenation operation, $\overrightarrow{\boldsymbol{h}}_l$ 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 $\boldsymbol{h} \in \mathbb{R}^{L \times 2H}$.

3.3 Tag Decoder

At each step of decoding, AT-BDRY 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 j is computed by:

$$\boldsymbol{d}_j = GRU(\boldsymbol{x}_j, \boldsymbol{\gamma}) \quad (6)$$

where $\boldsymbol{\gamma}$ 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 pass to the RNN. As shown in Figure 2(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 RNN. Supposing there are J time steps, the decoder RNN 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, AT-BDRY 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 as described above, we pad the hidden states of encoder with a sentinel word representing *inactive*. That is, the decoder should point to this sentinel word once the current input is not a start boundary of an entity. We also extend pointer networks [Vinyals *et al.*, 2015] with a direction-aware mechanism. 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 to compute the distribution of end boundary over all possible positions in the input sequence at decoding step j:

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

$$p(y_j|\boldsymbol{x}_j) = \text{softmax}(\boldsymbol{u}^j) \tag{8}$$

Here, $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ are learnable parameters, $i \in [j, J]$ indicates a possible position in the input sequence, and *softmax* normalizes u_i^j, indicating the probability that word W_i is an end boundary, given the start boundary W_j. When W_j 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, AT-BDRY points to "*inactive*" when given "*was*" as the decoder input in Figure 2(b).

3.4 Domain Discriminator

Recall that $\boldsymbol{h} \in \mathbb{R}^{(L+1)\times 2H}$ are the hidden states in the context encoder. We use an attention mechanism to form a context vector and apply a Multi-Layer Perceptron to predict domain label y_d (binary, *i.e.,* 0 and 1).

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

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

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

3.5 Adversarial Training

The tag prediction loss and the domain prediction loss can be written as

$$\mathcal{L}(\boldsymbol{\theta}_f, \boldsymbol{\theta}_s) = \sum_{\mathcal{D}_S}\sum_{j=1}^{J} -\log p(y_j|\boldsymbol{x}_j; \boldsymbol{\theta}_f, \boldsymbol{\theta}_s) \tag{12}$$

$$\mathcal{L}(\boldsymbol{\theta}_f, \boldsymbol{\theta}_t) = \sum_{\mathcal{D}_S, \mathcal{D}_T} -\log p(y_d|\boldsymbol{c}; \boldsymbol{\theta}_f, \boldsymbol{\theta}_t) \tag{13}$$

where $\boldsymbol{\theta}_f$ are the learnable parameters of the shared layers (input representation and context encoder), $\boldsymbol{\theta}_s$ are the parameters of the tag decoder, and $\boldsymbol{\theta}_t$ are the parameters of the discriminator. At learning time, in order to encourage domain-invariant features, we seek the parameters $\boldsymbol{\theta}_f$ that *maximize* the loss of the domain discriminator (by making the two feature distributions as indistinguishable as possible), while simultaneously seeking the parameters $\boldsymbol{\theta}_d$ and $\boldsymbol{\theta}_f$ that *minimize* the loss of the domain discriminator. In addition, we seek the parameters $\boldsymbol{\theta}_t$ 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 $E(\boldsymbol{\theta}_f, \boldsymbol{\theta}_s, \boldsymbol{\theta}_t) = \mathcal{L}(\boldsymbol{\theta}_f, \boldsymbol{\theta}_s) - \lambda\mathcal{L}(\boldsymbol{\theta}_f, \boldsymbol{\theta}_t)$. The parameter λ controls the trade-off between the two objectives. Then, we deliver a saddle point of $E(\boldsymbol{\theta}_f, \boldsymbol{\theta}_s, \boldsymbol{\theta}_t)$ as

$$(\hat{\boldsymbol{\theta}_f}, \hat{\boldsymbol{\theta}_s}) = \arg\min_{\boldsymbol{\theta}_f, \boldsymbol{\theta}_s} E(\boldsymbol{\theta}_f, \boldsymbol{\theta}_s, \hat{\boldsymbol{\theta}_t}) \tag{14}$$

$$\hat{\boldsymbol{\theta}_t} = \arg\max_{\boldsymbol{\theta}_t} E(\hat{\boldsymbol{\theta}_f}, \hat{\boldsymbol{\theta}_s}, \boldsymbol{\theta}_t) \tag{15}$$

Following [Ganin and Lempitsky, 2015], we add a special gradient reversal layer below the shared layer to address the minimax optimization problem.

Dataset	# Sentences			#Mentions
	Train	Dev	Test	
CoNLL03	14,987	3,466	3,684	34,841
OntoNotes5.0	59,917	8,528	8,262	71,031
WikiGold	144,342	-	1,696	298,961
WNUT16	2,394	1,000	3,856	5,630
WNUT17	3,394	1,009	1,287	3,890

Table 1: Statistics of datasets.

4 Experiments

4.1 Experimental Setup

Datasets. We use five popular benchmark datasets to ascertain the effectiveness of AT-BDRY. Because our task is boundary detection, we ignore entity types in all datasets. The statistics of the datasets are reported in Table 1. CoNLL03, OntoNotes5.0 and WikiGold are formal text. WNUT16 and WNUT17 are informal text.

Baselines. We evaluate AT-BDRY against 6 baselines. Note that INIT, TranT-B and CDMA-NER require annotated target-domain data for fine-tuning. We randomly leave out 20% of training set, and combine it with development set as annotated target-domain data for these three baselines. We measure Precision (P), Recall (R), and F-score (F1) to evaluate entity boundary detection accuracy.

- **SourceOnly** - It is trained only on source-domain data with pointer networks (Figure 2(b)) and directly applied to make predictions on target-domain data.
- **TJE** - This model projects both labels and features into a same low-dimensional space, where classification on the target-domain test data can be done with a simple nearest neighbor rule [Pan *et al.*, 2013].
- **INIT** - This model trains the parameters on a source dataset and transfers all parameters to initialize the model for training on a target dataset [Lee *et al.*, 2017].
- **TranT-B** - This model is trained with hierarchical recurrent networks and shares partial parameters in the source and target channels [Yang *et al.*, 2017].

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***

Table 2: The performance of entity boundary detection models on OntoNotes5.0. Significant improvement over baselines is marked with * (p-value < 0.01).

- **CDMA-NER** - This approach first pre-trains a source model and then uses three adaptation layers in the target model [Lin and Lu, 2018].
- **In-Domain** - This model is both trained and tested on target-domain data with pointer networks. Therefore, its performance can be regarded as the upper bound of transferring tasks.

Implementation Details. For all neural network models, we use GloVe 300-dimensional pre-trained word embeddings released by Stanford, which are fine-tuned during training. The dimension of 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. Each bidirectional encoder GRU has a depth of 3 and hidden size of 128. Each decoder GRU has a depth of 3 and hidden size of 256. Note that the encoder GRU is bidirectional and the decoder GRU is unidirectional in our model. Thus, the decoder has twice the hidden size of the encoder. The Adam optimizer was adopted with a learning rate of 0.001, selected from $\{0.01, 0.001, 0.0001\}$. We use a dropout of 0.5 after the convolution or recurrent layers. The decay rate is 0.09 and the gradient clip is 5.0. For neural baselines, we use exactly the same hyper-parameter grid and training procedure as our proposed model above. We report the results based on the best performance on the development set. All neural network models are implemented with PyTorch framework and evaluated on NVIDIA Tesla V100 GPU.

4.2 Results

We first present the performance of point networks for in-domain entity boundary detection. We then present the transferring performance on Formal Text → Formal Text and Formal Text → Informal Text. Finally, we verify the effect of adversarial transfer via an ablation study.

In-domain Entity Boundary Detection Performance

We train our boundary detection model (shown in Figure 2(b)) on OntoNotes5.0 and compare it with 5 methods. RegEx is created with regular expressions, based on word surface patterns, *e.g.,* letter case. Shallow-CRF trains a CRF using the commonly used token-level features [Liao and Veeramachaneni, 2009]. BiLSTM-MLP/CRF [Chiu and Nichols, 2016] utilizes BiLSTM to encode word sequence, and MLP/CRF to infer decoder tags. The results are summarized in Table 2.

First, our boundary detection model outperforms all baselines in terms of P, R and $F1$. More specifically, our approach 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. We attribute this to the fact that our approach resorts to point networks, which can effectively capture the dependencies of input sentence when the entity boundaries are sparse. Our solution also provides a new perspective to model sequence labeling task using pointer networks instead of the classic CRF-based approach.

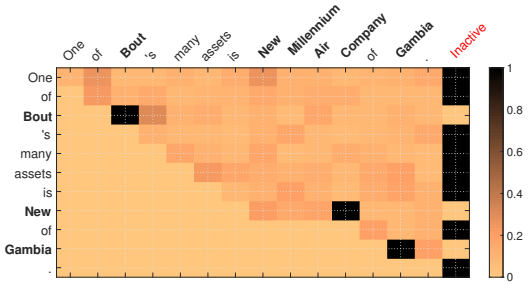


Figure 3: Visualization of pointer attention weights for sentence "*One of* ***Bout****'s many assets is* ***New Millennium Air Company*** *of* ***Gambia***.". Our model successfully detects three end boundaries at the positions: *Bout*, *Company* and *Gambia*.

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. Commonly, off-the-shelf NER systems cannot work well on cross-domain datasets because they do not take into account domain adaptation.

For better understanding, we visualize pointer attention weights with an example in Figure 3. 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. For example, for the input "*New*", our approach detects the end boundary of this entity at the position "*Company*". For a given input "*of*", our approach determines that it is not the start of an entity mention by the sentinel word "*Inactive*". Observe that the identified boundaries have dominant attention weights, which implies that our approach can successfully learn sentence features for entity boundary detection.

Formal Text → Formal Text

We consider one formal text data (*e.g.,* CoNLL03) as the source domain and another formal text (*e.g.,* Wikigold) as the target domain. Table 3 reports the performance on standard test sets of target domains.

From the results, we have the following observations. (1) AT-BDRY significantly outperforms all baseline methods to date. Compared with the second-best performance, AT-BDRY achieves relative F1 improvements of 3.98%, 1.95%, 3.02% and 4.22%, corresponding to each transferring task. (2) With the adversarial transfer mechanism, AT-BDRY significantly improves the performance of training only on source datasets. More specifically, AT-BDRY achieves relative F1 improvements of 13.69%, 3.25%, 9.52% and 12.13%, compared with the SourceOnly method. (3) For the case of CoNLL03→Wikigold, although AT-BDRY does not outperform the In-Domain method, the performance of AT-BDRY is close to the upper bound. It is worth noting that AT-BDRY

Methods	Source	CoNLL03			CoNLL03			OntoNotes5.0			OnotNotes5.0		
	Target	OntoNotes5.0			Wikigold			CoNLL03			Wikigold		
		$P\%$	$R\%$	$F1\%$	$P\%$	$R\%$	$F1\%$	$P\%$	$R\%$	$F1\%$	$P\%$	$R\%$	$F1\%$
SourceOnly		66.96	77.90	72.02	78.88	87.46	82.95	81.49	72.42	76.69	72.37	75.37	73.84
TJE [Pan *et al.*, 2013]		70.29	81.24	75.37	80.08	88.34	84.01	83.39	76.45	79.77	74.66	80.39	77.42
INIT [Lee *et al.*, 2017]		68.26	79.75	73.56	76.79	86.55	81.38	83.18	77.69	80.34	72.59	75.92	74.22
TranT-B [Yang *et al.*, 2017]		73.71	80.46	76.94	79.07	89.01	83.75	84.96	78.37	81.53	72.74	78.34	75.44
CDMA-NER [Lin and Lu, 2018]		74.40	82.67	78.32	78.12	88.78	83.11	83.80	78.23	80.92	77.88	81.09	79.45
AT-BDRY (ours)		**77.98***	**85.33***	**81.44***	**82.33***	**89.25***	**85.65***	**87.95***	**80.37***	**83.99***	**81.79***	**83.83***	**82.80***
In-Domain (upper bound)		93.22	92.67	92.94	85.64	90.82	88.15	94.80	95.93	95.36	85.64	90.82	88.15

Table 3: Transfer performance on Formal Text → Formal Text. Significant improvement over baselines is marked with * (p-value < 0.05).

Methods	Source	CoNLL03			CoNLL03			OntoNotes5.0			OnotNotes5.0		
	Target	WNUT16			WNUT17			WNUT16			WNUT17		
		$P\%$	$R\%$	$F1\%$	$P\%$	$R\%$	$F1\%$	$P\%$	$R\%$	$F1\%$	$P\%$	$R\%$	$F1\%$
SourceOnly		30.13	62.02	40.56	36.56	59.12	45.18	44.05	48.77	46.29	50.44	42.35	46.04
TJE [Pan *et al.*, 2013]		31.37	64.23	42.15	37.07	60.73	46.04	44.11	50.83	47.23	53.01	46.32	49.44
INIT [Lee *et al.*, 2017]		28.91	59.56	38.93	36.43	58.04	44.76	42.04	45.37	43.64	51.26	40.71	45.38
TranT-B [Yang *et al.*, 2017]		32.80	65.22	43.65	35.96	61.34	45.34	42.83	49.82	46.06	51.74	44.29	47.73
CDMA-NER [Lin and Lu, 2018]		31.98	64.73	42.81	36.69	62.13	46.14	**44.28**	52.31	47.96	49.45	46.78	48.08
AT-BDRY (ours)		**33.98***	**67.23***	**45.15***	**37.48**	**64.58***	**47.44***	43.36	**58.93***	**49.96***	**54.45***	**49.45***	**51.83***
In-Domain (upper bound)		63.98	51.45	57.03	63.36	50.32	56.09	63.98	51.45	57.03	63.36	50.32	56.09

Table 4: Transfer performance on Formal Text → Informal Text. Significant improvement over baselines is marked with * (p-value < 0.05).

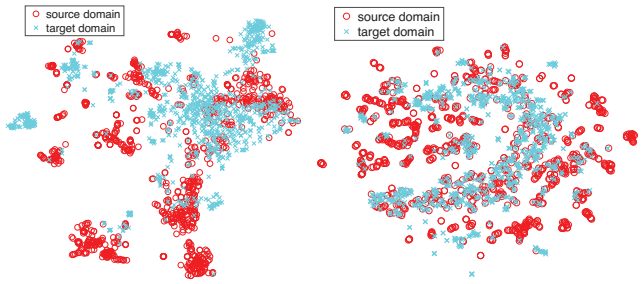
does not use any annotated target-domain data.

Formal Text → Informal Text

NER in formal text is well studied and the performance of many existing models drops dramatically in informal text. In this experiment, we investigate the effectiveness of AT-BDRY when the data distribution gap between the source domain and the target domain increases. Therefore, we choose formal text as the source domain and informal text as the target domain. Table 4 summarizes the results. Similar findings from Formal Text → Formal Text are observed in this experiment. In addition, we also observe that a better performance is achieved when OntoNotes5.0 (rather than CoNLL03) is taken as the source domain. In most cases, the recall score is much higher than the precision score, which implies that NER in formal text is a very challenging task because of the noisy nature of informal text as well as emerging entities with novel surfaces. Notably, AT-BDRY achieves an F1 of 51.83% on OntoNote5.0→WNUT17. This F1 score is much higher than the that of SourceOnly (46.04%) and is close to the upper bound (56.09%). We attribute this to the fact that AT-BDRY can learn domain-invariant representations, which can effectively reduce the distance in data distributions between the source and target domains.

4.3 Ablation Study

In our approach, the adversarial transfer component plays a role in encouraging domain-invariant representations. In this experiment, we investigate the effect of adversarial transfer by visualization. With t-SNE [Maaten and Hinton, 2008], Figure 4 demonstrates the effect of adversarial transfer on the distributions of the extracted features from the BiGRU encoder in the CoNLL03 → WNUT16 experiment. We can observe that our AT-BDRY approach possessing adversarial transfer learning, makes the two distributions of the extracted features much closer.



(a) Without adversarial transfer (b) With adversarial transfer

Figure 4: The effect of adversarial transfer on the distribution of the extracted features from the BiGRU encoder. The adversarial transfer in our approach makes the two distributions of features much closer.

5 Conclusion

In this paper, we study the task of named entity boundary detection. We propose AT-BDRY, an end-to-end neural model to detect entity boundaries from text without any handcrafted features or any prior linguistic knowledge. In particular, our approach incorporates adversarial transfer learning into a novel sequence labeling model to encourage domain-invariant representations. AT-BDRY ensures the extracted features are discriminative (for sequence labeling) and indistinguishable (for the discriminator) so that the knowledge from source domain can be transferred to the target domain, without requiring any annotated target-domain data. Through extensive experiments, we demonstrate the effectiveness of AT-BDRY against state-of-the-art solutions on different datasets.

References

[Abhishek *et al.*, 2017] Abhishek Abhishek, Ashish Anand, and Amit Awekar. Fine-grained entity type classification by jointly learning representations and label embeddings. In *EACL*, pages 797–807, 2017.

[Cao *et al.*, 2018] Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao, and Shengping Liu. Adversarial transfer learning for chinese named entity recognition with self-attention mechanism. In *EMNLP*, pages 182–192, 2018.

[Chiu and Nichols, 2016] Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. In *TACL*, volume 4, pages 357–370, 2016.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *SSST-8*, pages 103–111, 2014.

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

[Corro *et al.*, 2015] Luciano del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. Finet: Context-aware fine-grained named entity typing. In *EMNLP*, pages 868–878, 2015.

[Ganin and Lempitsky, 2015] Yaroslav Ganin and Victor S. Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, pages 1180–1189, 2015.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

[Lal *et al.*, 2017] Anurag Lal, Apoorve Tomer, and C Ravindranath Chowdary. Sane: System for fine grained named entity typing on textual data. In *WWW*, pages 227–230, 2017.

[Lample *et al.*, 2016] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *NAACL*, pages 260–270, 2016.

[Lee *et al.*, 2017] Ji Young Lee, Franck Dernoncourt, and Peter Szolovits. Transfer learning for named-entity recognition with neural networks. In *LREC*, pages 4470–4473, 2017.

[Li *et al.*, 2018a] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *arXiv preprint arXiv:1812.09449*, 2018.

[Li *et al.*, 2018b] Jing Li, Aixin Sun, and Shafiq Joty. Segbot: A generic neural text segmentation model with pointer network. In *IJCAI*, pages 4166–4172, 2018.

[Liao and Veeramachaneni, 2009] Wenhui Liao and Sriharsha Veeramachaneni. A simple semi-supervised algorithm for named entity recognition. In *NAACL-HLT*, pages 58–65, 2009.

[Lin and Lu, 2018] Bill Yuchen Lin and Wei Lu. Neural adaptation layers for cross-domain named entity recognition. In *EMNLP*, pages 2012–2022, 2018.

[Ma and Hovy, 2016] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL*, pages 1064–1074, 2016.

[Maaten and Hinton, 2008] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

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

[Pan *et al.*, 2013] Sinno Jialin Pan, Zhiqiang Toh, and Jian Su. Transfer joint embedding for cross-domain named entity recognition. *ACM TOIS*, 31(2):7, 2013.

[Phan *et al.*, 2018] Minh C Phan, Aixin Sun, Yi Tay, Jialong Han, and Chenliang Li. Pair-linking for collective entity disambiguation: Two could be better than all. *TKDE Early Access*, 2018.

[Ren *et al.*, 2016] Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *EMNLP*, pages 1369–1378, 2016.

[Shen *et al.*, 2017] Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, and Animashree Anandkumar. Deep active learning for named entity recognition. In *ICLR*, 2017.

[Vinyals *et al.*, 2015] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *NIPS*, pages 2692–2700, 2015.

[von Däniken and Cieliebak, 2017] Pius von Däniken and Mark Cieliebak. Transfer learning and sentence level features for named entity recognition on tweets. In *Proc. the 3rd Workshop on Noisy User-generated Text*, pages 166–171, 2017.

[Yang *et al.*, 2017] Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. Transfer learning for sequence tagging with hierarchical recurrent networks. In *ICLR*, 2017.

[Zhai *et al.*, 2017] Feifei Zhai, Saloni Potdar, Bing Xiang, and Bowen Zhou. Neural models for sequence chunking. In *AAAI*, pages 3365–3371, 2017.

[Zhao *et al.*, 2018] Huasha Zhao, Yi Yang, Qiong Zhang, and Luo Si. Improve neural entity recognition via multi-task data selection and constrained decoding. In *NAACL-HLT*, volume 2, pages 346–351, 2018.

[Zheng *et al.*, 2017] Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. Joint extraction of entities and relations based on a novel tagging scheme. In *ACL*, pages 1227–1236, 2017.