BOND: BERT-Assisted Open-Domain Named Entity Recognition with Distant Supervision

Chen Liang*, Yue Yu*, Haoming Jiang*, Siawpeng Er, Ruijia Wang, Tuo Zhao, Chao Zhang
Georgia Institute of Technology, Atlanta, GA, USA
{cliang73,yueyu,jianghm,ser8,rwang,tourzhao,chaozhang}@gatech.edu

ABSTRACT

We study the open-domain named entity recognition (NER) problem under distant supervision. The distant supervision, though does not require large amounts of manual annotations, yields highly incomplete and noisy distant labels via external knowledge bases. To address this challenge, we propose a new computational framework – BOND, which leverages the power of pre-trained language models (e.g., BERT and RoBERTa) to improve the prediction performance of NER models. Specifically, we propose a two-stage training algorithm: In the first stage, we adapt the pre-trained language model to the NER tasks using the distant labels, which can significantly improve the recall and precision; In the second stage, we drop the distant labels, and propose a self-training approach to further improve the model performance. Thorough experiments on 5 benchmark datasets demonstrate the superiority of BOND over existing distantly supervised NER methods. The code and distantly labeled data have been released in https://github.com/cliang1453/BOND.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**.

KEYWORDS

Named Entity Recognition, Open-Domain, Text Mining, Pre-trained Language Models, Distant Supervision, Self-Training

ACM Reference Format:
Chen Liang*, Yue Yu*, Haoming Jiang*, Siawpeng Er, Ruijia Wang, Tuo Zhao, Chao Zhang. 2020. BOND: BERT-Assisted Open-Domain Named Entity Recognition with Distant Supervision. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3394486.3403149

1 INTRODUCTION

Named Entity Recognition (NER) is the task of detecting mentions of real-world entities from text and classifying them into predefined types (e.g., locations, persons, organizations). It is a core task in knowledge extraction and is important to various downstream applications such as user interest modeling [13], question answering [14] and dialogue systems [2]. Traditional approaches to NER mainly train statistical sequential models, such as Hidden Markov Model (HMM) [47] and Conditional Random Field (CRF) [16] based on hand-crafted features. To alleviate the burden of designing hand-crafted features, deep learning models [11, 25] have been proposed for NER and shown strong performance. However, most deep learning methods rely on large amounts of labeled training data. As NER tasks require token-level labels, annotating a large number of documents can be expensive, time-consuming, and prone to human errors. In many real-life scenarios, the lack of labeled data has become the biggest bottleneck that prevents deep learning models from being adopted for NER tasks.

To tackle the label scarcity issue, one approach is to use distant supervision to generate labels automatically. In distant supervision, the labeling procedure is to match the tokens in the target corpus with concepts in knowledge bases (e.g. Wikipedia[1] and YAGO[2]), which are usually easy and cheap to access. Nevertheless, the labels generated by the matching procedure suffer from two major challenges. The first challenge is *incomplete annotation*, which is caused by the limited coverage of existing knowledge bases. Take two common open-domain NER datasets as examples. From Table 1, we find that the coverage of tokens on both datasets is very low (less than 60%). This issue renders many entities mentions unmatched and produces many false-positive labels, which can hurt subsequent NER model training significantly. The second challenge is *noisy annotation*. The annotation is often noisy due to the labeling ambiguity – the same entity mention can be mapped to multiple entity types in the knowledge bases. For instance, the entity mention '*Liverpool*' can be mapped to both '*Liverpool City*' (type: `LOC`) and '*Liverpool Football Club*' (type: `ORG`) in the knowledge base. While existing methods adopt label induction methods based on type popularity, they will potentially lead to a matching bias toward popular types. Consequently, it can lead to many false-positive samples and hurt the performance of NER models. What's worse, there is often a trade-off between the label accuracy and coverage: generating the high-quality label requires setting strict matching rules which may not generalize well for all the tokens and thus reduce the coverage and introduce false-negative labels. On the other hand, increasing the coverage of annotation suffers from the increasing number of incorrect labels due to label ambiguity. From the above, it is still very challenging to generate high-quality labels with high coverage to the target corpus.

Several studies have attempted to address the above challenges in distantly-supervised NER. To address the label incompleteness issue, some works adopt the partial annotation CRFs to consider all possible labels for unlabeled tokens [36, 45], but they still require

* These three authors contributed equally to this research.

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08.
https://doi.org/10.1145/3394486.3403149

[1] https://www.wikipedia.org/
[2] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/

a considerable amount of annotated tokens or external tools. To address the label noise issue, Ni et al. [28] use heuristic rules to filter out sentences with low matching quality. However, this filtering strategy improves the precision at the expense of lowering the recall. Cao et al. [3] attempt to induce labels for entity mentions based on their occurrence popularity in the concept taxonomy, which can suffer from labeling bias and produce mislabeled data. Moreover, most of the methods mainly focus on NER tasks in specific domains (e.g. biomedical, chemistry, etc.) where the ambiguity of the named entity is very low. When the matching ambiguity issue is more severe, such methods will be less effective especially under open-domain scenarios. Till now, training *open-domain* NER models with distant supervision remains a challenging problem.

We propose our model BOND, short for **B**ert-Assisted **O**pen-Domain **N**amed entity recognition with **D**istant Supervision, which learns accurate named entity taggers from distant supervision without any restriction on the domain or the content of the corpora. To address the challenges in learning from distant supervision, our approach leverages the power of pre-trained language models (e.g., ELMo [30], BERT [6], XLnet [46]) which are particularly attractive to this task due to the following merits: *First*, they are very large neural networks trained with huge amounts of unlabeled data in a *completely unsupervised manner*, which can be cheaply obtained; *Second*, due to their massive sizes (usually having hundreds of millions or billions of parameters), they have *strong expressive power* to capture general semantics and syntactic information effectively. These language models have achieved state-of-the-art performance in many popular NLP benchmarks with appropriate fine-tuning [6, 18, 23, 31, 46], which demonstrates their strong ability in modeling the text data.

To fully harness the power of pre-trained language models for tackling the two challenges, we propose a two-stage training framework. In the first stage, we fine-tune the RoBERTa model [23] with distantly-matched labels to essentially transfer the semantic knowledge in RoBERTa, which will improve the quality of prediction induced from distant supervision. It is worth noting that we adopt early stopping to prevent the model from overfitting to the incomplete annotated labels[3] and significantly improve the recall. Then we use the RoBERTa model to predict a set of pseudo soft-labels for all data. In the second stage, we replace the distantly-matched labels with the pseudo soft-labels and design a *teacher-student* framework to further improve the recall. The *student* model is first initialized by the model learned in the first stage and trained using pseudo soft-labels. Then, we update the *teacher* model from the *student* model in the previous iteration to generate a new set of pseudo-labels for the next iteration to continue the training of the *student* model. This *teacher-student* framework enjoys the merit that it progressively improves the model confidence over data. In addition, we select samples based on the prediction confidence of the *student* model to further improve the quality of soft labels. In this way, we can better exploit both the knowledge base information and the language models and improve the model fitting.

Our proposed method is closely related to low-resource NER and semi-supervised learning. We discuss more details in Section 5. We summarize the key contributions of our work as follows:

[3]Here the incomplete annotated labels refer to tokens wrongly labeled as type 'O'.

• We demonstrate that the pre-trained language model can also provide additional semantic information during the training process and reduce the label noise for distantly-supervised named entity recognition. To the best of our knowledge, this is the first work that leverages the power of pre-trained language model for open-domain NER tasks with distant supervision.

• We design a two-stage framework to fully exploit the power of language models in our task. Specifically, we refine the distant label iteratively with the language model in the first stage and improve the model fitting under the teacher-student framework in the second stage, which is able to address the challenge of noisy and incomplete annotation.

• We conduct comprehensive experiments on 5 datasets for named entity recognition tasks with distant supervision. Our proposed method significantly outperforms state-of-the-art distantly supervised NER competitors in all 5 datasets (4 of which by significant margins).

2 PRELIMINARIES

We briefly introduce the distantly-supervised NER problem and the pre-trained language models.

2.1 Distantly Supervised NER

NER is the process of locating and classifying named entities in text into predefined entity categories, such as person names, organizations, locations, etc. Formally, given a sentence with N tokens $\boldsymbol{X} = [x_1, ..., x_N]$, an entity is a span of tokens $\boldsymbol{s} = [x_i, ..., x_j]$ $(0 \le i \le j \le N)$ associated with an entity type. Based on the `BIO` schema [19], NER is typically formulated as a sequence labeling task of assigning a sequence of labels $\boldsymbol{Y} = [y_1, ..., y_N]$ to the sentence $\boldsymbol{X}$. Specifically, the first token of an entity mention with type `X` is labeled as `B-X`; the other tokens inside that entity mention are labeled as `I-X`; and the non-entity tokens are labeled as `O`.

For (fully) supervised NER, we are given M sentences that are already annotated at token level, denoted as $\{(\boldsymbol{X}_m, \boldsymbol{Y}_m)\}_{m=1}^M$. Let $f(\boldsymbol{X}; \theta)$ denote an NER model, which can compute N probability simplexes for predicting the entity labels of any new sentence $\boldsymbol{X}$, where θ is the parameter of the NER model. We train such a model by minimizing the following loss over $\{(\boldsymbol{X}_m, \boldsymbol{Y}_m)\}_{m=1}^M$:

$$\widehat{\theta} = \operatorname*{argmin}_{\theta} \frac{1}{M} \sum_{m=1}^{\mathrm{M}} \ell(\boldsymbol{Y}_m, f(\boldsymbol{X}_m; \theta)), \tag{1}$$

where $\ell(\cdot, \cdot)$ is the cross-entropy loss.

For distantly-supervised NER, we do not have access to well-annotated true labels, but only *distant labels* generated by matching unlabeled sentences with external gazetteers or knowledge bases (KBs). The matching can be achieved by string matching [9], regular expressions [8] or heuristic rules (e.g., POS tag constraints). Accordingly, we learn an NER model by minimizing Eq. (1) with $\{\boldsymbol{Y}_m\}_{m=1}^M$ replaced by their distantly labeled counterparts.

Challenges. The labels generated by distant supervision are often noisy and incomplete. This is particularly true for open-domain NER where there is no restriction on the domain or the content of the corpora. Fries et al. [8] and Giannakopoulos et al. [9] have proposed distantly-supervised NER methods for specific domains (*e.g.*,

Table 1: Existing Gazetteer Matching Performance on Open-Domain [35, 37] and Biomedical Domain NER Datasets [36]

Metric	Open-Domain		Biomedical Domains	
	CoNLL03	Tweet	BC5CDR	NCBI-Disease
Entity Types	4	10	2	1
F-1	59.61	35.83	71.98	69.32
Precision	71.91	40.34	93.93	90.59
Recall	50.90	32.22	58.35	56.15

biomedical domain), where the adopted domain-specific gazetteers or KBs are often of high matching quality and yield high precision and high recall distant labels. For the open domain, however, the quality of the distant labels is much worse, as there is more ambiguity and limited coverage over entity types in open-domain KBs. Table 1 illustrates the matching quality of distant labels on the open-domain and the biomedical-domain datasets. As can be seen, the distant labels for the open-domain datasets suffer from much lower precision and recall. This imposes great challenges to training accurate NER models.

2.2 Pre-trained Language Model

Pre-trained language models, such as BERT and its variants (*e.g.*, RoBERTa [23], ALBERT [18] and T5 [31]), have achieved state-of-the-art performance in many natural language understanding tasks [12]. These models are essentially massive neural networks based on bi-directional transformer architectures, and are trained using open-domain data in a completely unsupervised manner. The stacked self-attention modules of the transformer architectures can capture deep contextual information, and their non-recurrent structures enable the training to scale to large amounts of open-domain data. For example, the popular BERT-base model contains 110 million parameters, and is trained using the BooksCorpus [48] (800 million words) and English Wikipedia (2500 million words). More importantly, many pre-trained language models have been publicly available online. One does not need to train them from scratch. When applying pre-trained language models to downstream tasks, one only needs to slightly modify the model and adapt the model through efficient and scalable stochastic gradient-type algorithms.

3 TWO-STAGE FRAMEWORK: BOND

We introduce our proposed two-stage framework–BOND. In the first stage of BOND, we adapt the BERT model to the distantly supervised NER task. In the second stage, we use a self-training approach to improve the model fitting to the training data. We summarize the BOND framework in Figure 1.

3.1 Stage I: BERT-Assisted Distantly Supervised Learning with Early Stopping

Before proceeding with our proposed method, we briefly introduce how we generate distant labels for open-domain NER tasks. Our label generation scheme contains two steps: We first identify potential entities by POS tagging and hand-crafted rules. We then query from Wikidata to identify the types of these entities using SPARQL [40] as illustrated in Figure 2. We next collect gazetteers from multiple online resources to match more entities in the data [35]. Please refer to the appendix for more technical details.

We then proceed with our proposed method. We use $f(\cdot;\theta)$ to denote the NER model parameterized by θ, $f_{n,c}(\cdot;\cdot)$ to denote the probability of the n-th token belonging to the c-th class, and $\{(\boldsymbol{X}_m, \boldsymbol{D}_m)\}_{m=1}^M$ to denote the distantly labeled data, where $\boldsymbol{D}_m = [d_{m,1}, ..., d_{m,N}]$ and $\boldsymbol{X}_m = [x_{m,1}, ..., x_{m,N}]$. The NER model $f(\cdot;\theta)$ is learned by minimizing the loss over $\{(\boldsymbol{X}_m, \boldsymbol{D}_m)\}_{m=1}^M$:

$$\widehat{\theta} = \operatorname*{argmin}_{\theta} \frac{1}{M} \sum_{m=1}^{M} \ell(\boldsymbol{D}_m, f(\boldsymbol{X}_m;\theta)), \tag{2}$$

where $\ell(\boldsymbol{D}_m, f(\boldsymbol{X}_m;\theta)) = \frac{1}{N}\sum_{n=1}^{N} -\log f_{n,d_{m,n}}(\boldsymbol{X}_m;\theta)$.

The architecture of the NER model $f(\cdot,\cdot)$ is a token-wise NER classifier on top of a pre-trained BERT, as shown in Figure 3. The NER classifier takes in the token-wise output embeddings from the pre-trained BERT layers, and gives the prediction on the type for each token. The pre-trained BERT contains rich semantic and syntax knowledge, and yields high quality output embeddings. Using such embeddings as the initialization, we can efficiently adapt the pre-trained BERT to the target NER task using stochastic gradient-type algorithms, e.g., ADAM [15, 22]. Following [31], our adaptation process updates the entire model including both the NER classification layer and the pre-trained BERT layers.

Algorithm 1: Stage I: BERT-Assisted Distantly Supervised Learning with Early Stopping

Input: M unlabeled sentences, $\{\boldsymbol{X}_m\}_{m=1}^M$; External KBs including Wikidata and multi-source gazetteers; The NER model with pre-trained BERT layers $f(\cdot;\theta^{(0)})$; The early stopping time T_1; The updating formula of ADAM $\mathcal{T}$.

// **Distant Label Generation (DLG)**

$\{\boldsymbol{D}_m\}_{m=1}^M = \text{Matching}(\{\boldsymbol{X}_m, \boldsymbol{D}_m\}_{m=1}^M; \text{External KBs})$

// **Model Adaptation**

for $t = 1, 2, ..., T_1$ **do**

Sample a minibatch $\mathcal{B}_t$ from $\{(\boldsymbol{X}_m, \boldsymbol{D}_m)\}_{m=1}^M$.

Update the model using ADAM:

$$\theta^{(t)} = \mathcal{T}(\theta^{(t-1)}, \mathcal{B}_t).$$

Output: The early stopped model: $\widehat{\theta} = \theta^{(T_1)}$

Figure 4 illustrates how the pre-trained BERT embeddings help the model adapt to distantly supervised NER tasks. We highlight that BERT is pre-trained through a masked language model (MLM) task, and is capable of predicting the missing words using the contextual information. Such a MLM task shares a lot of similarity with the NER task. Both of them are token-wise classification problems and heavily rely on the contextual information (see Figure 3). This naturally enables the semantic knowledge of the pre-trained BERT to be transferred to the NER task. Therefore, the resulting model can better predict the entity types than those trained from scratch using only the distantly labeled data.

Early Stopping. One important strategy we use in the adaptation process is early stopping. Due to the large model capacity as well as the limited and noisy supervision (distant labels), our NER model can overfit the noise in distant labels and forget the knowledge

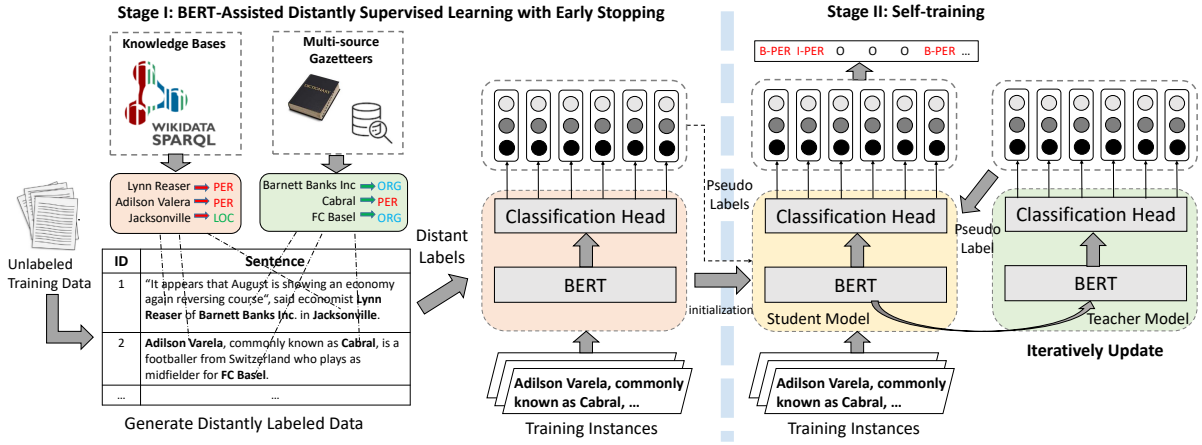


Figure 1: The two-stage BOND framework. In Stage I, the pre-trained BERT is adapted to the distantly supervised NER task with early stopping. In Stage II, a student model and a teacher model are first initialized from the model learned in Stage I. Then the student model is trained using pseudo-labels generated by the teacher model. Meanwhile, the teacher model is iteratively updated by the early-stopped student.

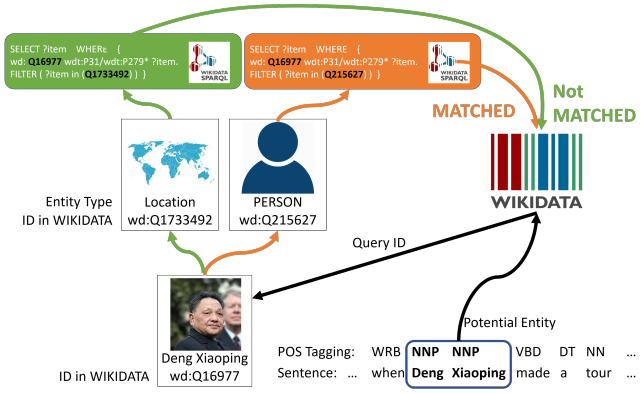


Figure 2: Illustration of matching entities from Wikidata

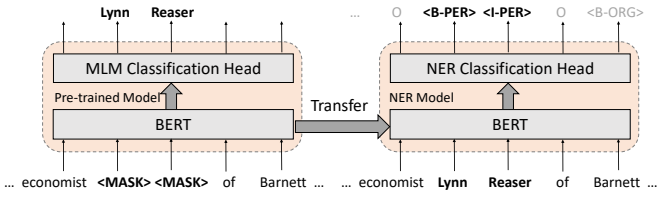


Figure 3: Pre-trained Mask Language Model vs. NER Model

of the pre-trained BERT if without any intervention. Early stopping essentially serves as a strong regularization to prevent such overfitting and improves generalization ability to unseen data.

Remark 1. Stage I addresses both of the two major challenges in distantly supervised NER tasks: noisy annotation and incomplete annotation. As the semantic knowledge in the pre-trained BERT is transferred to the NER model, the noise is suppressed such that the prediction precision is improved. Moreover, early stopping prevents the model from overfitting the incomplete annotated labels and further improves the recall.

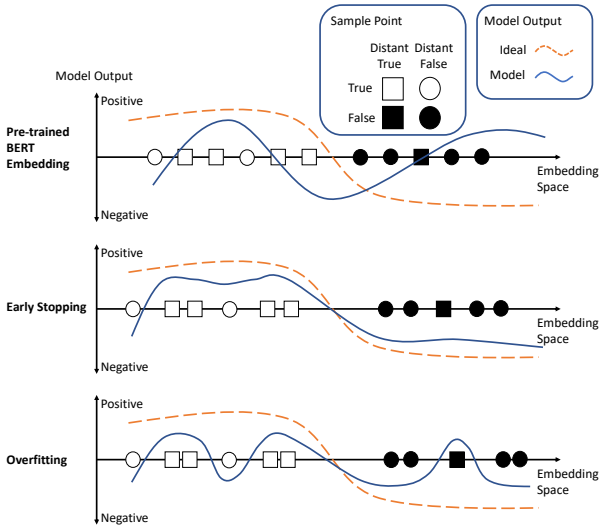


Figure 4: Illustration of Stage I. Top) The pre-trained semantic knowledge is transferred to the NER task; Middle) Early stopping leverages the pre-trained knowledge and yields better prediction; Bottom) Without early stopping, the model overfits the noise. The token embeddings are evolving, as we update the pre-trained BERT layers.

3.2 Stage II: Self-Training

We first describe a teacher-student framework of self-training to improve the model fitting, and then we propose to use high-confidence soft labels to further improve the self-training.

3.2.1 The Teacher-student Framework. We use $f(\cdot;\theta_{\text{tea}})$ and $f(\cdot;\theta_{\text{stu}})$ to denote teacher and student models, respectively. Given the model learned in Stage I, $f(\cdot;\widehat{\theta})$, one option is to initialize the teacher model and the student model as:

$$\theta_{\text{tea}}^{(0)} = \theta_{\text{stu}}^{(0)} = \widehat{\theta},$$

and another option is

$$\theta_{\text{tea}}^{(0)} = \widehat{\theta} \quad \text{and} \quad \theta_{\text{stu}}^{(0)} = \theta_{\text{BERT}}, \tag{3}$$

where θ_{BERT} denotes the initial model with the pre-trained BERT layers used in Stage I. For simplicity, we refer the second option to "re-initialization".

At the t-th iteration, the teacher model generates pseudo labels $\{\widetilde{Y}_m^{(t)} = [\widetilde{y}_{m,1}^{(t)}, ..., \widetilde{y}_{m,N}^{(t)}]\}_{m=1}^M$ by

$$\widetilde{y}_{m,n}^{(t)} = \underset{c}{\text{argmax}}\, f_{n,c}(\boldsymbol{X}_m; \theta_{\text{tea}}^{(t)}). \tag{4}$$

Then the student model fits these pseudo-labels. Specifically, given the teacher model $f(\cdot; \theta_{\text{tea}}^{(t)})$, the student model is learned by solving

$$\widehat{\theta}_{\text{stu}}^{(t)} = \underset{\theta}{\text{argmin}} \frac{1}{M} \sum_{m=1}^{M} \ell(\widetilde{Y}_m^{(t)}, f(\boldsymbol{X}_m; \theta)). \tag{5}$$

We then use ADAM to optimize Eq. (5) with early stopping. At the end of t-th iteration, we update the teacher model and the student model by:

$$\theta_{\text{tea}}^{(t+1)} = \theta_{\text{stu}}^{(t+1)} = \widehat{\theta}_{\text{stu}}^{(t)}.$$

The algorithm is summarized in Algorithm 2.

Algorithm 2: Stage II: Self-Training

Input: M training sentences, $\{\boldsymbol{X}_m\}_{m=1}^M$; The early stopped model obtained in Stage I, $f(\cdot; \widehat{\theta})$; The number of self-training iterations T_2; The early stopping time T_3; The updating formula of ADAM $\mathcal{T}$.

Initialize the teacher model and the student model:

$$\theta_{\text{tea}}^{(0)} = \theta_{\text{stu}}^{(0)} = \widehat{\theta}.$$

for $t = 1, 2, ...T_2$ **do**

  $\theta_{\text{stu}}^{(t,0)} = \theta_{\text{stu}}^{(t)}$.

  for $k = 1, 2, ..., T_3$ **do**

    Sample a minibatch $\mathcal{B}_k$ from $\{\boldsymbol{X}_m\}_{m=1}^M$.

    Generate pseudo-labels $\{\widetilde{Y}_m\}_{m\in\mathcal{B}_k}$ by Eq. (4).

    Update the student model:

$$\theta_{\text{stu}}^{(t,k)} = \mathcal{T}(\theta_{\text{stu}}^{(t,k-1)}, \{(\boldsymbol{X}_m, \widetilde{Y}_m)\}_{m\in\mathcal{B}_k}).$$

  Update the teacher and student:

$$\theta_{\text{tea}}^{(t)} = \theta_{\text{stu}}^{(t)} = \theta_{\text{stu}}^{(t,T_3)}.$$

Output: The final student model: $\theta^{(T_2)}$

Remark 2. Note that we discard all pseudo-labels from the $(t$-1)-th iteration, and only train the student model using pseudo-labels generated by the teacher model at the t-th iteration. Combined with early stopping, such a self-training approach can improve the model fitting and reduce the noise of the pseudo-labels as illustrated in Figure 5. With progressive refinement of the pseudo-labels, the student model can gradually exploit knowledge in the pseudo-labels and avoid overfitting.

Remark 3. Our teacher-student framework is quite general, and can be naturally combined with other training techniques, e.g., mean teacher [38] and virtual adversarial training [27]. Please refer to Section 5 for more detailed discussions.

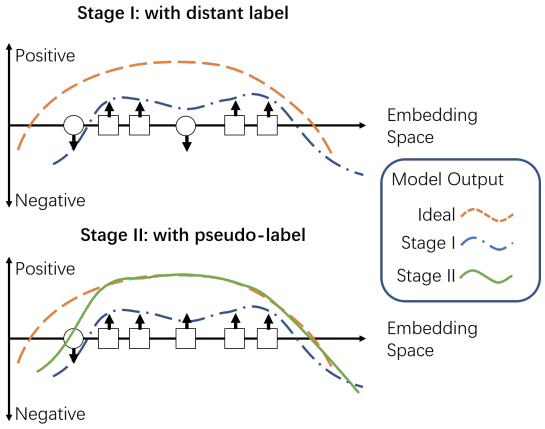


Figure 5: Illustration of self-training. The self-training can gradually reduce the noise of the pseudo-labels and improve model fitting.

3.2.2 Re-weighted High-Confidence Soft Labels. The hard pseudo-labels generated by Eq. (4) only keeps the most confident class for each token. To avoid losing too much information of other classes, we propose to use soft labels with confidence re-weighting.

Recall that for the n-th token in the m-th sentence, the output probability simplex over C classes is denoted as

$$[f_{n,1}(\boldsymbol{X}_m; \theta), ..., f_{n,C}(\boldsymbol{X}_m; \theta)].$$

At the t-th iteration, the teacher model generates soft pseudo-labels $\{\boldsymbol{S}_m^{(t)} = [\boldsymbol{s}_{m,n}^{(t)}]_{n=1}^N\}_{m=1}^M$ following [43]:

$$\boldsymbol{s}_{m,n}^{(t)} = [s_{m,n,c}^{(t)}]_{c=1}^C = \left[\frac{f_{n,c}^2(\boldsymbol{X}_m; \theta_{\text{tea}}^{(t)})/p_c}{\sum_{c'=1}^C f_{n,c'}^2(\boldsymbol{X}_m; \theta_{\text{tea}}^{(t)})/p_{c'}} \right]_{c=1}^C \tag{6}$$

where $p_c = \sum_{m=1}^M \sum_{n=1}^N f_{n,c}(\boldsymbol{X}_m; \theta_{\text{tea}}^{(t)})$ calculates the unnormalized frequency of the tokens belonging to the c-th class. As can be seen, such a squared re-weighting step in Eq. (6) essentially favors the classes with higher confidence. The student model $f(\cdot; \theta_{\text{stu}}^{(t)})$ is then optimized by minimizing

$$\theta_{\text{stu}}^{(t)} = \underset{\theta}{\text{argmin}} \frac{1}{M} \sum_{m=1}^{M} \ell_{\text{KL}}(\boldsymbol{S}_m^{(t)}, f(\boldsymbol{X}_m; \theta)),$$

where $\ell_{\text{KL}}(\cdot, \cdot)$ denotes the KL-divergence-based loss:

$$\ell_{\text{KL}}(\boldsymbol{S}_m^{(t)}, f(\boldsymbol{X}_m; \theta)) = \frac{1}{N} \sum_{n=1}^{N} \sum_{c=1}^{C} -s_{m,n,c}^{(t)} \log f_{n,c}(\boldsymbol{X}_m; \theta). \tag{7}$$

High-Confidence Selection. To further address the uncertainty in the data, we propose to select tokens based on the prediction confidence. Specifically, at the t-th iteration, we select a set of high confidence tokens from the m-th sentence by

$$H_m^{(t)} = \{n : \max_c s_{m,n,c}^{(t)} > \epsilon\}, \tag{8}$$

where $\epsilon \in (0, 1)$ is a tuning threshold. Accordingly, the student model $f(\cdot; \theta_{\text{stu}}^{(t)})$ can be optimized by minimizing the loss only over the selected tokens:

$$\theta_{\text{stu}}^{(t)} = \underset{\theta}{\text{argmin}} \frac{1}{M|H_m^{(t)}|} \sum_{m=1}^{M} \sum_{n\in H_m^{(t)}} \sum_{c=1}^{C} -s_{m,n,c}^{(t)} \log f_{n,c}(\boldsymbol{X}_m; \theta).$$

The high confidence selection essentially enforces the student model to better fit tokens with high confidence, and therefore is able to improve the model robustness against low-confidence tokens.

4 EXPERIMENTS

We conduct a series of experiments to demonstrate the superiority of our proposed method.

4.1 Experimental Setup

4.1.1 Datasets. We consider the following NER benchmark datasets: (i) **CoNLL03** [39] is a well-known open-domain NER dataset from the CoNLL 2003 Shared Task. It consists of 1393 English news articles and is annotated with four entity types: person, location, organization, and miscellaneous. (ii) **Twitter** [10] is from the WNUT 2016 NER shared task. This is an open-domain NER dataset that consists of 2400 tweets (comprising 34k tokens) with 10 entity types. (iii) **OntoNotes5.0** [41] contains text documents from multiple domains, including broadcast conversation, P2.5 data and Web data. It consists of around 1.6 millions words and is annotated with 18 entity types. (iv) **Wikigold** [1] is a set of Wikipedia articles (40k tokens) randomly selected from a 2008 English dump and manually annotated with the four CoNLL03 entity types. (v) **Webpage** [33] is an NER dataset that contains personal, academic, and computer science conference webpages. It consists of 20 webpages that cover 783 entities belonging to the four types the same as CoNLL03.

For distant labels generation, we match entity types in external KBs including Wikidata corpus and gazetteers collected from multiple online sources. The data sources and matching details are described in the appendix.

4.1.2 Baselines. We compare our model with different groups of baseline methods.

• **KB Matching.** The first baseline performs string matching with external KBs using the mechanism described in the appendix.

• **Fully-supervised Methods.** We also include fully-supervised NER methods for comparison, including: (i) **RoBERTa-base** [23]—it adopts RoBERTa model with linear layers to perform token-level prediction; (ii) **BiLSTM-CRF** [25] adopts bi-directional LSTM with character-level CNN to produce token embeddings, which are fed into a CRF layer to predict token labels.

• **Distantly-supervised Methods.** The third group of baselines are recent deep learning models for distantly-supervised NER, including: (i) **BiLSTM-CRF** [25] is trained using the distant labels matched from KBs; (ii) **AutoNER** [36] trains the model by assigning ambiguous tokens with all possible labels and then maximizing the overall likelihood using a fuzzy LSTM-CRF model; (iii) **LRNT** [3] is the state-of-the-art model for low-resource named tagging, which applies partial-CRFs on high-quality data with non-entity sampling. When comparing with these distantly supervised methods, we use the same distant labels as the training data for fair comparison.

• **Baselines with Different Settings**. The following methods also conduct open-domain NER under distant supervision. We remark that they use different KBs and extra training data. Therefore, we only compare with the results reported in their papers. (i) KALM [21] augments a traditional language model with a KB and use entity type information to enhance the model. (ii) ConNET [17] leverages multiple crowd annotation and dynamically aggregates them by attention mechanism. It learn from imperfect annotations from multiple sources.[4]

• For **Ablation Study**, we consider the following methods/tricks. (i) **MT** [38] uses Mean Teacher method to average model weights and forms a target-generating teacher model. (ii) **VAT** [27] adopts virtual adversarial training to smooth the output distribution to make the model robust to noise. (iii) **Hard Label** generates pseudo-labels using Eq. (4). (iv) **Soft Label** generates pseudo-labels using Eq. (6). (v) **Reinitialization** initializes the student and teacher models using Eq. (3). (vi) **High-Confidence Selection** selects tokens using Eq. (8).

4.2 Experimental Results

Our NER model use RoBERTa-base as the backbone. A linear classification layer is build up on the RoBERTa-base model. Please refer to the appendix for implementation details.

4.2.1 Main Results. Table 2 presents the F_1 scores, precision and recall for all methods. Note that our implementations of the fully supervised NER methods attain very close to the state-of-the-art performance [6, 20]. Our results are summarized as follows:

• For all five datasets, our method consistently achieves the best performance under the distant supervision scenarios, in F_1 score, precision and recall. In particular, our method outperforms the strongest distantly supervised NER baselines by $\{11.74, 21.91, 0.66, 14.35, 12.53\}$ in terms of F_1 score. These results demonstrate the significant superiority of our proposed method.

• The standard adaptation of pre-trained language models have already demonstrated remarkable performance. The models obtained by the Stage I of our methods outperform the strongest distantly supervised NER baselines by $\{5.87, 20.51, 0.42, 7.72, 4.01\}$ in terms of F_1 score. The Stage II of our methods further improves the performance of the Stage I by $\{5.87, 1.4, 0.24, 6.63, 8.52\}$.

• On CoNLL03 dataset, compared with baselines which use different sources – KALM and ConNET, our model also outperforms them by significant margins. More detailed technical comparisons between our method and them are provided in Section 5.

4.2.2 Ablation Study. To gain insights of our two-stage framework, we investigate the effectiveness of several components of our method via ablation study. The table 3 shows the results on both CoNLL03 and Wikigold datasets. Our results can be summarized as follows:

• For Stage I, **Pre-trained Language Models** significantly improve both precision and recall for both datasets. Specifically, when training the NER model from scratch, the F1 scores of the output model of Stage I drop from 75.61 to 36.66 on CoNLL03, and from 51.55 to 18.31 on Wikigold. This verifies that the rich semantic and contextual information in pre-trained RoBERTa has been successfully transferred to our NER model in Stage I.

• For Stage I, **Early stopping** improves both precision and recall for both datasets. We increase the training iterations from 900 to 18000 on CoNLL03 and from 350 to 7000 on Wikigold, and the F1 scores of the output model of Stage I drop from 75.61 to 72.11 on

[4]For KALM and ConNET model, the KB and crowd annotation are not public available, and thus we are unable to reproduce the results.

Table 2: Main Results on Testing Set: F_1 Score (Precision/Recall) (in %)

Method	CoNLL03	Tweet	OntoNote5.0	Webpage	Wikigold
Entity Types	4	10	18	4	4
KB Matching	71.40(81.13/63.75)	35.83(40.34/32.22)	59.51(63.86/55.71)	52.45(62.59/45.14)	47.76(47.90/47.63)
Fully-Supervised (Our implementation)					
RoBERTa	90.11(89.14/91.10)	52.19(51.76/52.63)	86.20(84.59/87.88)	72.39(66.29/79.73)	86.43(85.33/87.56)
BiLSTM-CRF	91.21(91.35/91.06)	52.18(60.01/46.16)	86.17(85.99/86.36)	52.34(50.07/54.76)	54.90(55.40/54.30)
Baseline (Our implementation)					
BiLSTM-CRF	59.50(75.50/49.10)	21.77(46.91/14.18)	66.41(68.44/64.50)	43.34(58.05/34.59)	42.92(47.55/39.11)
AutoNER	67.00(75.21/60.40)	26.10(43.26/18.69)	67.18(64.63/69.95)	51.39(48.82/54.23)	47.54(43.54/52.35)
LRNT	69.74(79.91/61.87)	23.84(46.94/15.98)	67.69(67.36/68.02)	47.74(46.70/48.83)	46.21(45.60/46.84)
Other Baseline (Reported Results)					
KALM †	76.00(– / –)	–	–	–	–
ConNET ◇	75.57(84.11/68.61)	–	–	–	–
Our BOND **Framework**					
Stage I	75.61(83.76/68.90)	46.61(53.11/41.52)	68.11(66.71/69.56)	59.11(60.14/58.11)	51.55(49.17/54.50)
BOND	81.48(82.05/80.92)	48.01(53.16/43.76)	68.35(67.14/69.61)	65.74(67.37/64.19)	60.07(53.44/68.58)

Note: †: KALM achieves better performance when using extra data. ◇: ConNET studies NER under a crowd sourcing setting, where the best human annotator achieves F_1 score at 89.51.

CoNLL03, and from 51.55 to 49.68 on Wikigold. This verifies that Early Stopping eases the overfitting and improves the generalization ability of our NER model.

• For Stage II, **Soft labels** improve the F_1 score and recall on both datasets. Specifically, the F_1 scores and recall increase from 77.28/71.98 to 80.18/78.84 on CoNLL03, and from 56.90/59.74 to 58.64/65.79 on Wikigold. Moreover, the precision on Wikigold is also improved. This verifies that the soft labels preserve more information and yield better fitted models than those of the hard labels.

• For stage II, **High-Confidence Selection** improves the F_1 scores on both datasets. Specifically, compared with using soft labels, the F_1 scores and recall increase from 81.56/78.84 to 80.18/72.31 on CoNLL03, and from 58.64/59.74 to 60.07/68.58 on Wikigold. Besides, the precision on CoNLL03 is also improved. This verifies that the high-confidence labels help select data and yield more robust performance.

• For Stage II, **Re-initialization** improves both precision and recall, only when the hard labels are adopted. We believe that this is because the hard labels lose too much information about data uncertainty, re-initializing the RoBERTa layers restores semantic and contextual information, and can compensate such loss.

In contrast, when soft labels are adopted, **Re-initialization** deteriorates both precision and recall. We believe that this is because the soft label retains sufficient information (i.e., the knowledge transferred from RoBERTa and learned from the distant labels). As a result, re-initialization only leads to underfitting on the data.

Moreover, we also consider **Multiple Re-initialization**, and observe similar results.

• **Mean Teacher** and **Virtual Adversarial Training** can be naturally integrated into our versatile teacher-student framework by adding an additional MT teacher or a VAT teacher. **VAT** marginally improves the F1 scores on both datasets. **MT** marginally improves the F1 scores on Wikigold, and deteriorates the F_1 scores on CoNLL03.

Table 3: Ablation Study: F_1 Score (Precision/Recall) (in %)

Method	CoNLL03	Wikigold
Stage I		
Stage I	75.61(83.76/68.90)	51.55(49.17/54.50)
Stage I w/o pre-train	36.66(37.49/35.75)	18.31(18.14/18.50)
Stage I w/o early stop	72.11(81.65/64.57)	49.68(48.67/50.74)
Stage I w/ MT	76.30(82.92/70.67)	46.68(49.82/43.91)
Stage I w/ VAT	76.38(82.58/71.04)	47.54(50.02/45.30)
Stage I + Stage II		
BOND †	77.28(83.42/71.98)	56.90(54.32/59.74)
BOND w/ soft	80.18(81.56/78.84)	58.64(58.29/65.79)
BOND w/ soft+high conf	81.48(82.05/80.92)	60.07(53.44/68.58)
BOND w/ reinit	78.17(85.05/72.31)	58.55(55.31/62.19)
BOND w/ soft+reinit	76.92(83.39/71.38)	54.09(50.72/57.94)
BOND w/ MT	77.16(82.79/72.25)	57.93(55.66/60.39)
BOND w/ VAT	77.64(85.62/70.69)	57.39(55.05/59.41)

Note†*:* We use BOND to denote our two-stage framework using hard pseudo-labels in this table for clarity.

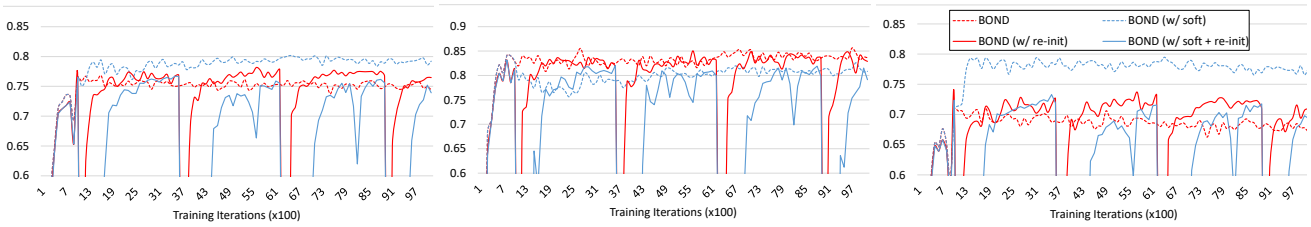
We believe that this is because **MT** and **VAT** perform well with high quality labels, however, the labels in our NER tasks are not very precise.

4.2.3 Parameter Study. We investigate the effects of the early stopping time of Stage I – T_1, the early stopping time of Stage II– T_3, and confidence threshold ϵ for selecting tokens using CoNLL03 data. The default values are $T_1 = 900, T_3 = 1800, \epsilon = 0.9$. The learning curves are summarized in Figure 6:

• Both T_1 and T_3 reflect trade-offs between precision and recall of the Stage I and Stage II, respectively. This verifies the importance of early stopping. The model performance is sensitive to T_1, and less sensitive to T_3.

• The recall increases along with ϵ. The precision shows a different behavior: it first decreases and then increases.

• We also consider a scenario, where T_3 is allowed to tune for each iteration of the Stage II. This requires more computational resource



(a) F_1 score (b) Precision (c) Recall

Figure 6: Learning Curves of BOND**,** BOND **(w/ reinit),** BOND **(w/ soft) and** BOND **(w/ soft + reinit)**

than the setting where T_3 remains the same for all iterations. This can further improve the model performance to 83.49, 84.09, 82.89 in terms of F_1 scores, precision and recall, respectively.

4.2.4 Case Study and Error Analysis. To demonstrate how BOND improves the recall, we compare the prediction performance of KB matching with the output models of Stage I and Stage II using Wikigold data. Figure 8 presents the bar plots of four entity types – "LOC", "PER", "ORG" and "MISC". As can be seen, the KB matching yields a large amount of "O" (non-entity) due to its limited coverage. As a result, the recall is very low 47.63%. In contrast, our model of the Stage I benefits from the transferred knowledge of pre-trained RoBERTa and is able to correct some wrongly matched O's to their corresponding entity types. Therefore, it enjoys a better recall 54.50%. Moreover, the self-training in the Stage II further improves the recall to 68.48%.

5 RELATED WORK AND DISCUSSION

Our work is related to **low-resource NER**. This line of research focuses on leveraging cross lingual information to improve the model performance. For examples, [5, 7] consider NER for a low resource target language. They propose to train an NER model with annotated language that are closely related to the target language. [44] proposes to use the bilingual dictionaries to tackle this challenge. More recently, [32] proposes a Bayesian graphical model approach to further improve the low resource NER performance.

Our work is also relevant to **semi-supervised learning**, where the training data is only partially labeled. There have been many semi-supervised learning methods, including the popular Mean Teacher and Virtual Adversarial Training methods used in our experiments for comparison [4, 26, 27, 34, 38]. Different from distant supervision, these semi-supervised learning methods usually has a partial set of labeled data. They rely on the labeled data to train an sufficiently accurate model. The unlabeled data are usually used for inducing certain regularization to further improve the generalization performance. The distant supervision, however, considers the setting with only noisy labels. Existing semi-supervised learning methods such as Mean Teacher and Virtual Adversarial Training can only marginally improve the performance, as shown in the ablation study in our experiments.

Other related works: [21] proposes a language model-based method – KALM for NER tasks. However, their approach has two drawbacks: (i) Since they design a language model designated for NER tasks, they need to first train the language models from scratch. However, this often requires a large amount of training corpus and enormous computational resources. In contrast, BOND uses general-purpose pre-trained language models, which are publicly available online. (ii) The training of their language model is not fully unsupervised and requires token-level annotations. To address this issue, they resort to distant supervision, which yields incomplete and noisy annotations. Therefore, their language model does not necessarily achieve the desired performance.

Larger Pre-trained Language Models: To further improve the performance of BOND, we can use larger pre-trained language models such as RoBERTa-large [23] (Three times as big as RoBERT-base in our experiments) and T5 [31] (Thirty times larger than RoBERTa-base). These larger models contain more general semantics and syntax information, and have the potentials to achieve even better performance for NER Tasks. Unfortunately, due to the limitation of our computational resources, we are unable to use them in our experiments.

REFERENCES

[1] Dominic Balasuriya, Nicky Ringland, Joel Nothman, Tara Murphy, and James R Curran. 2009. Named entity recognition in wikipedia. In *the 2009 Workshop on The People's Web Meets NLP*. 10–18.

[2] Kevin Bowden, Jiaqi Wu, Shereen Oraby, Amita Misra, and Marilyn Walker. 2018. SlugNERDS: A Named Entity Recognition Tool for Open Domain Dialogue Systems. In *LREC*.

[3] Yixin Cao, Zikun Hu, Tat-seng Chua, Zhiyuan Liu, and Heng Ji. 2019. Low-Resource Name Tagging Learned with Weakly Labeled Data. In *EMNLP-IJCNLP*. 261–270.

[4] Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc V. Le. 2018. Semi-Supervised Sequence Modeling with Cross-View Training. In *EMNLP*.

[5] Ryan Cotterell and Kevin Duh. 2017. Low-Resource Named Entity Recognition with Cross-lingual, Character-Level Neural Conditional Random Fields. In *IJCNLP*. Asian Federation of Natural Language Processing, 91–96.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.

[7] Xiaocheng Feng, Xiachong Feng, Bing Qin, Zhangyin Feng, and Ting Liu. 2018. Improving Low Resource Named Entity Recognition using Cross-lingual Knowledge Transfer. In *IJCAI*. 4071–4077.

[8] Jason Fries, Sen Wu, Alex Ratner, and Christopher Ré. 2017. Swellshark: A generative model for biomedical named entity recognition without labeled data. *arXiv preprint arXiv:1704.06360* (2017).

[9] Athanasios Giannakopoulos, Claudiu Musat, Andreea Hossmann, and Michael Baeriswyl. 2017. Unsupervised Aspect Term Extraction with B-LSTM & CRF using Automatically Labelled Datasets. In *the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. 180–188.

[10] Fréderic Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van de Walle. 2015. Multimedia lab@ acl wnut ner shared task: Named entity recognition for twitter microposts using distributed word representations. In *WNUT*. 146–153.

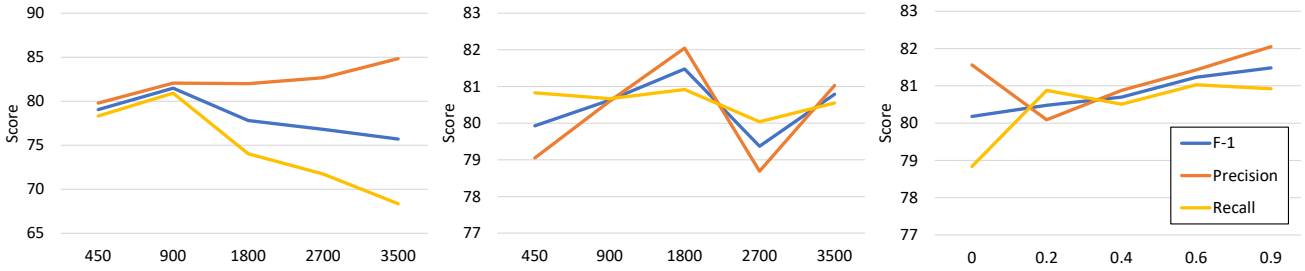
[11] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).

[12] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2019. SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization. *arXiv preprint arXiv:1911.03437* (2019).

[13] Deniz Karatay and Pinar Karagoz. 2015. User Interest Modeling in Twitter with Named Entity Recognition. *Making Sense of Microposts (# Microposts2015)* (2015).

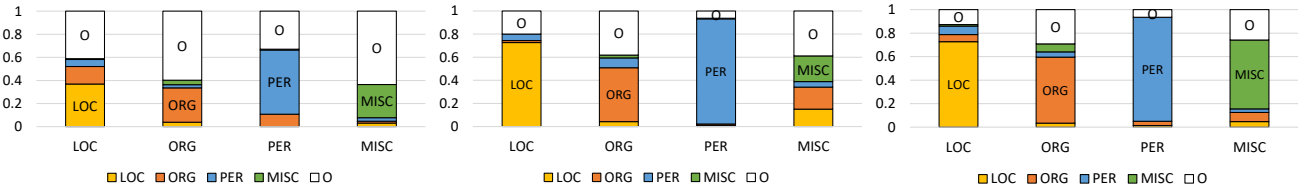
[14] Mahboob Alam Khalid, Valentin Jijkoun, and Maarten De Rijke. 2008. The impact of named entity normalization on information retrieval for question answering. In *ECIR*. Springer, 705–710.

[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).



(a) The Early Stopping Time of Stage I – T_1 (b) The Early Stopping Time in Stage II – T_3 (c) The Confidence Threshold of Stage II – ϵ

Figure 7: Parameter Study using CoNLL03: F_1, Precision, Recall on Testing Set (in %)



(a) Knowledge Base Matching (b) Stage I (c) Stage II

Figure 8: Recall of Knowledge Base Matching and different stages of BOND. The horizontal axis denotes the true entity type. The segments in a bar denote the portions of the entities being classified into different entity types.

[16] John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

[17] Ouyu Lan, Xiao Huang, Bill Yuchen Lin, He Jiang, Liyuan Liu, and Xiang Ren. 2020. Learning to Contextually Aggregate Multi-Source Supervision for Sequence Labeling. In *ACL*.

[18] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR*.

[19] Qi Li, Haibo Li, Heng Ji, Wen Wang, Jing Zheng, and Fei Huang. 2012. Joint bilingual name tagging for parallel corpora. In *CIKM*. 1727–1731.

[20] Nut Limsopatham and Nigel Collier. 2016. Bidirectional LSTM for named entity recognition in Twitter messages. (2016).

[21] Angli Liu, Jingfei Du, and Veselin Stoyanov. 2019. Knowledge-augmented language model and its application to unsupervised named-entity recognition. In *NAACL*.

[22] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. On the Variance of the Adaptive Learning Rate and Beyond. In *ICLR*.

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

[24] Edward Loper and Steven Bird. 2002. NLTK: the natural language toolkit. *arXiv preprint cs/0205028* (2002).

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

[26] Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2018. Weakly-supervised neural text classification. In *CIKM*. 983–992.

[27] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE T-PAMI* 41, 8 (2018), 1979–1993.

[28] Jian Ni, Georgiana Dinu, and Radu Florian. 2017. Weakly Supervised Cross-Lingual Named Entity Recognition via Effective Annotation and Representation Projection. In *ACL*. 1470–1480.

[29] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. 1532–1543.

[30] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *NAACL-HLT*. 2227–2237.

[31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683* (2019).

[32] Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Multilingual NER transfer for low-resource languages. *arXiv preprint arXiv:1902.00193* (2019).

[33] Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*. 147–155.

[34] Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. 2005. Semi-Supervised Self-Training of Object Detection Models. In *WACV*. 29–36.

[35] Erik F Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050* (2003).

[36] Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. 2018. Learning Named Entity Tagger using Domain-Specific Dictionary. In *EMNLP*. 2054–2064.

[37] Benjamin Strauss, Bethany Toma, Alan Ritter, Marie-Catherine De Marneffe, and Wei Xu. 2016. Results of the wnut16 named entity recognition shared task. In *WNUT*. 138–144.

[38] Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NIPS*. 1195–1204.

[39] Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *CoNLL-2003*. 142–147.

[40] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.

[41] Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA* 23 (2013).

[42] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* (2016).

[43] Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *ICML*. 478–487.

[44] Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A. Smith, and Jaime Carbonell. 2018. Neural Cross-Lingual Named Entity Recognition with Minimal Resources. In *EMNLP*. 369–379. https://doi.org/10.18653/v1/D18-1034

[45] Yaosheng Yang, Wenliang Chen, Zhenghua Li, Zhengqiu He, and Min Zhang. 2018. Distantly supervised ner with partial annotation learning and reinforcement learning. In *COLING*. 2159–2169.

[46] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*. 5754–5764.

[47] GuoDong Zhou and Jian Su. 2002. Named entity recognition using an HMM-based chunk tagger. In *ACL*. 473–480.

[48] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*. 19–27.

A DETAILED DESCRIPTION OF DISTANT LABEL GENERATION

A.1 External Knowledge Bases

Wikidata is a collaborative and free knowledge base for the acquisition and maintenance of structured data. It contains over 100 million tokens extracted from the set of verified articles on Wikipedia. Wikidata knowledge imposes a high degree of structured organization. It provides a SPARQL query service for users to obtain entity relationships.

Multi-sources Gazetteers. For each dataset, we build a gazetteer for each entity type. Take CoNLL03 as an example, we build a gazetteer for the type PER by collecting data from multiple online sources including Random Name[5], US First Names Database[6], Word Lists[7], US Census Bureau[8], German Surnames[9], Surnames Database[10] and Surname List[11]. We build a gazetteer for the type ORG by collecting data from Soccer Team[12], Baseball Team[13] and Intergovernmental Organization[14]. We will release all gazetteers and codes for matching distant labels after the paper is accepted for publication.

A.2 Distant Labels Generation Details

We first find potential entities by POS tagging obtained from POS tagger, e.g., NLTK [24]. We then match these potential entities by using Wikidata query service. Specifically, we use SPARQL to query the parent categories of an entity in the knowledge tree. We continue querying to the upper levels until a category corresponding to a type is found. For entities with ambiguity (e.g., those linked with multiple parent categories), we discard them during the matching process (i.e., we assign them with type O). The above procedure is summarized in Figure 9.

We then build, for each entity type in each dataset, a multi-sources gazetteer by crawling online data sources. Following the previous exact string matching methods[9, 35], we match an entity with a type if the entity appears in the gazetteer for that type.

For the unmatched tokens, we further use a set of hand-crafted rules to match entities. We notice that among the true entities, there is usually a stamp word. We match a potential entity with a type if there exists a stamp word in this entity that has frequent occurrence in that type. For example, "Inc." frequently occurs in organization names, thus the appearance of "Inc." indicates that the entity labels of words in the "XXX Inc." should be B-ORG or I-ORG).

Note that for Twitter, we do not build our own multi-sources gazetteer. We directly use the baseline system proposed in [10] to generate the distant labels.

[5]https://github.com/dominictarr/random-name
[6]https://data.world/len/us-first-names-database
[7]https://github.com/imsky/wordlists
[8]https://www2.census.gov/topics/genealogy/2010surnames/
[9]https://ziegenfuss.bplaced.net/zfuss/surnames-all.php?tree=1
[10]https://www.surnamedb.com/Surname
[11]https://surnameslist.org/
[12]https://footballdatabase.com/ranking/world
[13]https://www.ducksters.com/sports/list_of_mlb_teams.php
[14]https://en.wikipedia.org/wiki/List_of_intergovernmental_organizations

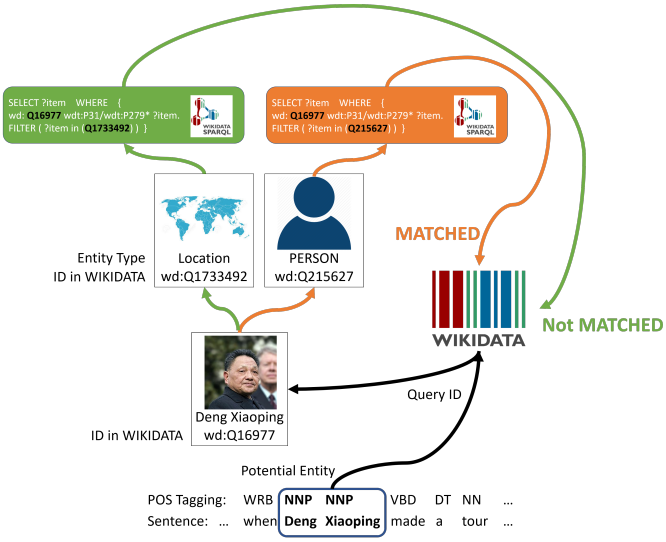


Figure 9: Illustration of matching entities from Wikidata

B BASELINE SETTINGS

For the baselines, we implement LSTM-CNN-CRF with Pytorch[15] and use the pre-trained 100 dimension GloVe Embeddings [29] as the input vector. Then, we set the dimension of character-level embeddings to 30 and feed them into a 2D convolutional neural network (CNN) with kernel width as 3. Then, we tune the output dimension in range of [25, 50, 75, 100, 150] and report the best performance. We train the model for 50 epochs with early stopping. We use SGD with momentum with $m = 0.9$ and set the learning rate as 2×10^{-3}. We set the dropout rate to 0.5 for linear layers after LSTM. We tune weight decay in range of $[10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}]$ and report the best performance.

For other baselines, we follow the officially released implementation from the authors: (1) AutoNER: https://github.com/shangjingbo1226/AutoNER; (2) LRNT: https://github.com/zig-kwin-hu/Low-Resource-Name-Tagging.

C IMPLEMENTATION DETAILS OF BOND

All implementation are based on the Huggingface Transformer codebase [16].

C.1 Adapting RoBERTa to the NER task

We choose RoBERTa-base as the backbone model of our NER model. A linear classification layer is built upon the pre-trained RoBERTa-base as illustrated in Figure 10.

C.2 Pseudo-labels Generation Details

BERT uses WordPiece [42] for tokenization of the input text. When the teacher model predicts a set of pseudo-labels for all training data in Stage II, it assign labels for padded tokens as well. We ignore those labels in training and loss computation step by label masking.

[15]https://pytorch.org/
[16]https://github.com/huggingface/transformers

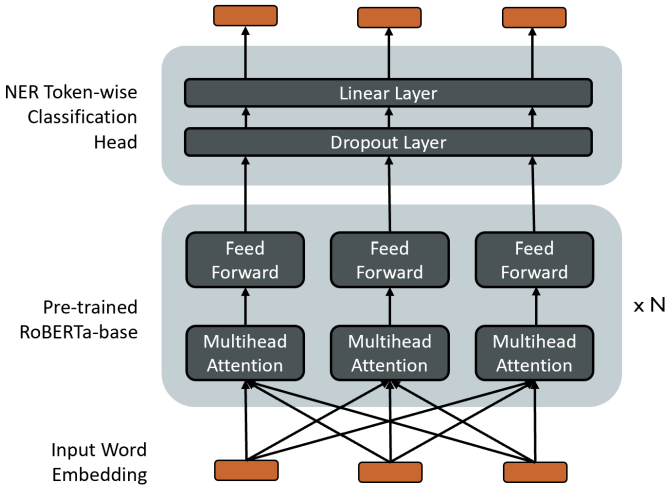


Figure 10: The NER Model with Pre-trained RoBERTa

C.3 Parameter Settings

There are several key parameters in our model: 1) For CoNLL03, we choose $T_1 = 900$ (about 1 epoch) and $T_3 = 1756$ (about 2 epochs). For Tweet, we choose $T_1 = 900$ and $T_3 = 900$. For OntoNotes5, we choose $T_1 = 16500$ and $T_3 = 1000$. For Webpage, we choose $T_1 = 300$ and $T_3 = 200$. For Wikigold, we choose $T_1 = 350$ and $T_3 = 700$. As for T_2, we stop training when the number of total training epochs reaches 50 for all datasets. 2) We choose 10^{-5} as the learning rate for CoNLL03, Webpage and Wikigold and 2×10^{-5} for OntoNotes5, Twitter, all with learning rate linear decay of 10^{-4}. 3) We use AdamW with β_1=0.9 and β_2=0.98 as optimizer for all datasets. 4) We set ϵ=0.9 for all datasets. 5) The training batch size is 16 for all datasets except OntoNotes5.0, which uses 32 as the training batch size. 6) For the NER token-wise classification head, we set dropout rate as 0.1 and use a linear classification layer with hidden size 768. For MT, we set ramp-up step as 300 for CoNLL03, 200 for Tweet, 200 for OntoNotes5.0, 300 for Webpage and 40 for Wikigold. We choose the moving average parameters as $\alpha_1 = 0.99$ and $\alpha_2 = 0.995$ for all datasets. For VAT, we set the perturbation size $\epsilon_{vat} = 10^{-4}$.

C.4 Multiple Re-initialization

Multiple Re-initialization is implemented as follows: In Stage II, as the performance of the student model no longer improves, we re-initialize it from the pre-trained RoBERTa-base and start a new self-training iteration.

C.5 Combine BOND w/ MT&VAT

MT&VAT can easily combined with BOND as follows: During training, we update the student model by minimize the sum of weighted MT (or VAT) loss and Eq. (7). The weight of MT (or VAT) loss is selected in $[10, 1, 10^{-1}, 10^{-2}, 10^{-3}]$ using development set.