# Distantly-Supervised Named Entity Recognition with Noise-Robust Learning and Language Model Augmented Self-Training

**Yu Meng, Yunyi Zhang, Jiaxin Huang, Xuan Wang,**
**Yu Zhang, Heng Ji, Jiawei Han**
University of Illinois Urbana-Champaign, IL, USA
{yumeng5, yzhan238, jiaxinh3, xwang174,
yuz9, hengji, hanj}@illinois.edu

## Abstract

We study the problem of training named entity recognition (NER) models using only distantly-labeled data, which can be automatically obtained by matching entity mentions in the raw text with entity types in a knowledge base. The biggest challenge of distantly-supervised NER is that the distant supervision may induce incomplete and noisy labels, rendering the straightforward application of supervised learning ineffective. In this paper, we propose (1) a noise-robust learning scheme comprised of a new loss function and a noisy label removal step, for training NER models on distantly-labeled data, and (2) a self-training method that uses contextualized augmentations created by pre-trained language models to improve the generalization ability of the NER model. On three benchmark datasets, our method achieves superior performance, outperforming existing distantly-supervised NER models by significant margins[1].

## 1 Introduction

Named entity recognition (NER), which aims at identifying real-world entity mentions (*e.g.*, person, location and organization names) from texts, is a fundamental task in natural language processing with a wide range of applications, including question answering (Khalid et al., 2008), knowledge base construction (Etzioni et al., 2005), text summarization (Aramaki et al., 2009) and dialog systems (Bowden et al., 2018). In recent years, deep neural models (Devlin et al., 2019; Huang et al., 2015; Lample et al., 2016; Ma and Hovy, 2016) have achieved enormous success for NER, thanks to their strong representation learning power that accurately captures the entity semantics in textual contexts. However, a common bottleneck of applying deep learning models is the acquisition of abundant high-quality human annotations, and this is especially the case for training NER models, which require every entity mention to be labeled in a sequence.

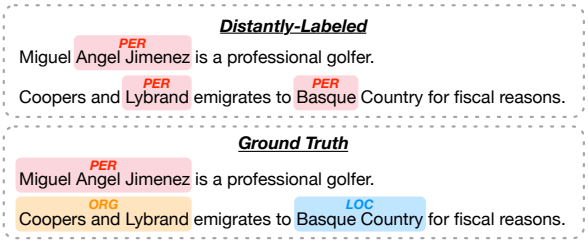


Figure 1: Distant labels obtained with knowledge bases may be incomplete and noisy, resulting in wrongly-labeled tokens.

To eliminate the need for human annotations, one direction is to use distant supervision for automatic generation of entity labels. The common practice is to match entity mentions in the target corpus with typed entities in external gazetteers or knowledge bases. Unfortunately, such a distant labeling process inevitably introduces incomplete and noisy entity labels, because (1) the distant supervision source has limited coverage of the entity mentions in the target corpus, and (2) some entities can be matched to multiple types in the knowledge bases—such ambiguity cannot be resolved by the context-free matching process. Figure 1 shows that some "*person*" mentions may be partially labeled (or not labeled at all in other cases), and some entities with multiple possible types may be mislabeled.

Due to the existence of such noise, straightforward application of supervised learning to distantly-labeled data will yield deteriorated performance, because neural models have the strong capacity to fit to the given (noisy) data. Some previous studies on distantly-supervised NER directly treat distant labels as if ground truth for model training and rely on simple tricks such as applying early stopping (Liang et al., 2020) and labeling entities with

[1]Code can be found at https://github.com/yumeng5/RoSTER.



*Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10367–10378
November 7–11, 2021. 

multiple types (Shang et al., 2018) to handle the noise. Others require an additional manually labeled training set for building a noise classification model (Onoe and Durrett, 2019).

In this paper, we study the distantly-supervised NER problem without requiring any human annotations. Our method consists of two steps: (1) noise-robust learning, and (2) language model augmented self-training. In the first step, we explicitly address the label noise by using a noise-robust loss function and removing noisy labels. In the second step, we use the model's high-confidence predictions for self-training to improve generalization, wherein a pre-trained language model is used to not only initialize the NER model, but also generate contextualized augmentations. Our method is named **RoSTER**, for **Ro**bust learning and **S**elf-**T**raining for distantly-supervised **E**ntity **R**ecognition.

The contributions of this paper are as follows:

- We propose a noise-robust learning scheme for distantly-supervised NER, comprised of a noise-robust loss function and a noisy label removal step.
- We propose a new unsupervised contextualized augmentation approach for NER using pre-trained language models. Combined with self-training, the created augmentations improve the model's generalization ability.
- On three benchmark datasets, **RoSTER** outperforms existing distantly-supervised NER approaches by significant margins.

## 2 Method

In this section, we (1) briefly describe how to obtain distantly-labeled data, (2) introduce our noise-robust learning scheme and (3) propose a self-training method with a new contextualized augmentation generation technique. We assume the pre-trained RoBERTa (Liu et al., 2019) model is used as our backbone model, but our proposed methods can be integrated with other architectures (*e.g.*, LSTM-based (Ma and Hovy, 2016)) as well.

### 2.1 Distant Label Generation

Given an unlabeled corpus, the distant labels are usually obtained by matching entities in the corpus with those in the external knowledge bases or gazetteers with typing information. In this work, instead of introducing new distant label generation methods, we follow the previous work (Liang et al., 2020) for this step: (1) potential entities are determined via POS tagging and hand-crafted rules, (2) their types are acquired by querying Wikidata using SPARQL (Vrandečić and Krötzsch, 2014), and (3) additional gazetteers from multiple online resources are used for matching more entities in the corpus.

### 2.2 Noise-Robust Learning

We first overview the common setup for training NER models, and then propose two designs that work jointly for distantly-supervised NER, motivated by the challenges of learning with noisy labels: (1) a new loss function, and (2) noisy label removal. Finally, ensembling multiple models is helpful for stabilizing the model performance.

NER systems are usually trained as a sequence labeling model that classifies every token in a sequence into a set of entity types or non-entity. The label space depends on the tagging scheme used (*e.g.*, the BIO format distinguishes begin/inside/outside of named entities). Specifically, given a sequence $\boldsymbol{x} = [x_1, \ldots, x_n]$ of $n$ tokens and their corresponding categorical labels $\boldsymbol{y} = [y_1, \ldots, y_n]$, an NER model parameterized by $\boldsymbol{\theta}$ is trained to minimize some classification loss that encourages the model to correctly predict the given labels. The cross entropy (CE) loss is most commonly used for such a purpose:

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{n} \log f_{i,y_i}(\boldsymbol{x}; \boldsymbol{\theta}),$$

where $f_{i,j}(\boldsymbol{x}; \boldsymbol{\theta})$ is the model's predicted probability of token $x_i$ belonging to class $j$ (*i.e.*, the softmax layer outputs).

The gradient of $\mathcal{L}_{\text{CE}}$ is (via the chain rule):

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{CE}} = -\sum_{i=1}^{n} \frac{\nabla_{\boldsymbol{\theta}} f_{i,y_i}(\boldsymbol{x}; \boldsymbol{\theta})}{f_{i,y_i}(\boldsymbol{x}; \boldsymbol{\theta})}. \quad (1)$$

Due to the $f_{i,y_i}(\boldsymbol{x}; \boldsymbol{\theta})$ term as the denominator, the tokens on which the model's prediction is less congruent with the provided labels (*i.e.*, $f_{i,y_i}(\boldsymbol{x}; \boldsymbol{\theta})$ is smaller) will be implicitly weighed more during the gradient update. Such a mechanism grants better model convergence when trained with clean data (*i.e.*, $\boldsymbol{y}$ are ground truth labels), because more emphasis is put on difficult tokens. However, when the labels are noisy, training with the cross entropy loss can cause overfitting to the wrongly-labeled tokens (*e.g.*, the two sentences in Figure 1).

Contrary to cross entropy loss which is sensitive to noise, the mean absolute error (MAE) loss, which is commonly used in regression tasks, has been shown inherently noise-tolerant when used for classification (Ghosh et al., 2017) and is defined as follows (omitting the constant scale factor 2):

$$\mathcal{L}_{\text{MAE}} = \sum_{i=1}^{n} \left(1 - f_{i,y_i}(\boldsymbol{x};\boldsymbol{\theta})\right),$$

and its gradient is given by

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{MAE}} = -\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} f_{i,y_i}(\boldsymbol{x};\boldsymbol{\theta}). \quad (2)$$

By comparing Eq. (2) with Eq. (1), we observe that $\mathcal{L}_{\text{MAE}}$ is more noise-robust than $\mathcal{L}_{\text{CE}}$ because Eq. (2) treats every token equally for gradient update, allowing the learning process to be dominated by the correct majority in distant labels. However, using $\mathcal{L}_{\text{MAE}}$ for training deep neural models generally worsens the convergence efficiency and effectiveness due to the inability of adjusting for challenging training samples, and leads to suboptimal model performance compared to using $\mathcal{L}_{\text{CE}}$ (Zhang and Sabuncu, 2018).

**Generalized Cross Entropy.** To balance between model convergence and noise-robustness, we propose to use the generalized cross entropy (GCE) loss (Zhang and Sabuncu, 2018) for training distantly-supervised NER models, inspired by the $q$-order entropy (Ferrari et al., 2010), defined as follows:

$$\mathcal{L}_{\text{GCE}} = \sum_{i=1}^{n} \frac{1 - f_{i,y_i}(\boldsymbol{x};\boldsymbol{\theta})^q}{q}, \quad (3)$$

where $0 < q < 1$ is a hyperparameter: When $q \to 1$, $\mathcal{L}_{\text{GCE}}$ approximates $\mathcal{L}_{\text{MAE}}$; when $q \to 0$, $\mathcal{L}_{\text{GCE}}$ approximates $\mathcal{L}_{\text{CE}}$ (using L'Hôpital's rule; see Appendix A for the derivation). The gradient is computed as:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{GCE}} = -\sum_{i=1}^{n} \frac{\nabla_{\boldsymbol{\theta}} f_{i,y_i}(\boldsymbol{x};\boldsymbol{\theta})}{f_{i,y_i}(\boldsymbol{x};\boldsymbol{\theta})^{1-q}}. \quad (4)$$

Comparing Eq. (4) to Eq. (1), it can be observed that $\mathcal{L}_{\text{GCE}}$ is more noise-robust than $\mathcal{L}_{\text{CE}}$ because less weights are given to tokens on which the model prediction is less consistent with the given labels (note $f_{i,y_i}(\boldsymbol{x};\boldsymbol{\theta})^{1-q} > f_{i,y_i}(\boldsymbol{x};\boldsymbol{\theta})$ for $q > 0$ and $f_{i,y_i}(\boldsymbol{x};\boldsymbol{\theta}) < 1$). Comparing Eq. (4) to Eq. (2), it can be seen that $\mathcal{L}_{\text{GCE}}$ facilitates better learning dynamics than $\mathcal{L}_{\text{MAE}}$ because difficult tokens are given more attention to.

**Noisy Label Removal.** Even when a noise-robust loss is used, mislabeled tokens still deteriorate the model performance as long as they are included in training. Unfortunately, without any prior knowledge about which tokens are mislabeled, it is challenging to automatically detect them. We propose a simple threshold-based strategy to remove noisy labels: At first, all tokens along with their distant labels will be used for model training; later, those tokens on which the model prediction does not strongly agree with its distant label (*i.e.*, $f_{i,y_i}(\boldsymbol{x};\boldsymbol{\theta}) \leq \tau$ where $\tau$ is a threshold value) will be excluded from the training set (*i.e.*, not calculated in the loss). The intuition is straightforward: Since our loss function is noise-robust, the learned model will be dominated by the correct majority in the distant labels instead of quickly overfitting to label noise; if the model prediction disagrees with some given labels, they are potentially wrong.

Specifically, we extend Eq. (3) to incorporate the aforementioned design, as follows:

$$\mathcal{L}_{\text{GCE}} = \sum_{i=1}^{n} w_i \frac{1 - f_{i,y_i}(\boldsymbol{x};\boldsymbol{\theta})^q}{q}, \quad (5)$$

where $w_i = 1$ at the start of training and is periodically updated once every several batches as $w_i = \mathbb{1}\left(f_{i,y_i}(\boldsymbol{x};\boldsymbol{\theta}) > \tau\right)$, where $\mathbb{1}(\cdot)$ is the indicator function.

**Model Ensemble for Better Stability.** Due to the stochasticity involved in training neural networks (*e.g.*, dataset shuffling, network random initialization and dropout), models trained with the same algorithm will have different predictions on the same dataset, and this is especially true when inconsistent noisy signals from the distant labels may further disturb model training. As such, model ensemble is commonly used to suppress the noise and provide better stability, by combining the predictions of multiple models (Laine and Aila, 2017; Nguyen et al., 2019). The rationale is that the model's predictions are likely to be consistent on clean data while inconsistent and oscillating on wrongly-labeled data, and ensembling multiple models enhances consistent predictions and neutralizes inconsistent ones.

We perform model ensemble by simply training $K$ models $\{\boldsymbol{\theta}_k\}_{k=1}^{K}$ via Eq. (5) on the same distantly-labeled corpus with different random seeds controlling the randomness of the training process. A new model $\boldsymbol{\theta}_{\text{ENS}}$ is finally trained to approximate the average prediction of the $K$ models

on all tokens by minimizing the Kullback–Leibler (KL) divergence loss:

$$\mathcal{L}_{\text{ENS}} = \sum_{i=1}^{n} \text{KL}\left(\bar{f}_i\left(\boldsymbol{x}; \{\boldsymbol{\theta}_k\}_{k=1}^{K}\right) \| f_i(\boldsymbol{x}; \boldsymbol{\theta}_{\text{ENS}})\right), \tag{6}$$

where $\bar{f}_i\left(\boldsymbol{x}; \{\boldsymbol{\theta}_k\}_{k=1}^{K}\right) = \frac{1}{K}\sum_{k=1}^{K} f_i(\boldsymbol{x}; \boldsymbol{\theta}_k)$ is the $K$ models' averaged prediction, and we find that $K = 5$ is sufficient to provide stable ensembled model performance.

**Remarks.** While our methods introduce two additional hyperparameters $q$ and $\tau$, their values can be kept same for different datasets to avoid parameter tuning. We will also show in Section 3 that the model performance is rather insensitive to these hyperparameter values within a reasonable range.

### 2.3 Language Model Augmented Self-Training

After the noise-robust learning step, we further fine-tune the resulting model (*i.e.*, trained with Eq. (6)) via a self-training step on the same corpus, but without the distant labels, for two reasons: (1) The clean signals in the distant supervision have been exploited via noise-robust learning, but some tokens may have not been fully leveraged by the model since they are excluded by the noisy label removal step. The self-training step aims to bootstrap on all tokens using the model's own predictions to improve its generalization ability. Similar self-training ideas have been explored in classification tasks (Meng et al., 2018, 2019, 2020). (2) The pre-trained language model (PLM) has only been used to initialize the NER model for fine-tuning, while PLMs (without fine-tuning) encode factual and relational knowledge through pre-training (Petroni et al., 2019) that may complement the NER model training. The self-training step thus also brings additional pre-trained knowledge for better model generalization by creating contextualized augmentations using a PLM. Figure 2 shows an overview.

**Contextualized Augmentations with PLMs.** Many PLMs (Devlin et al., 2019; Lan et al., 2020; Liu et al., 2019) are pre-trained with the masked language modeling (MLM) task on large-scale text corpora carrying general knowledge like the Wikipedia. Previous studies (Jiang et al., 2020; Petroni et al., 2019) have shown that entity-related knowledge can be extracted from a PLM (without any fine-tuning) by querying it via cloze templates and gathering the PLM's MLM outputs.

Given that the MLM task shares high similarity with the NER task (*i.e.*, both leverage the contextual information within the sequence for token-level classification) and that the MLM outputs contain general knowledge acquired during pre-training, we propose to use the pre-trained RoBERTa model (without fine-tuning) $\boldsymbol{\theta}_{\text{PRE}}$ for creating label-preserving augmentations (*i.e.*, not changing the entity type label or non-entity label) of the original sequences in the corpus in order to complement the NER learning with pre-trained knowledge. Specifically, for each sequence $\boldsymbol{x}$ in the corpus, we randomly mask out $15\%$ of its tokens (*i.e.*, replace them with the `[MASK]` token), and feed the partially masked sequence $\hat{\boldsymbol{x}}$ into the pre-trained RoBERTa model. Finally, the augmented sequence $\boldsymbol{x}' = [x'_1, \ldots, x'_n]$ is created by sampling from the MLM output probability for each token:

$$x'_i \sim p_i^{\text{MLM}}\left(\hat{\boldsymbol{x}}; \boldsymbol{\theta}_{\text{PRE}}\right), \tag{7}$$

where $p_i^{\text{MLM}}\left(\hat{\boldsymbol{x}}; \boldsymbol{\theta}_{\text{PRE}}\right)$ is the MLM probability of the pre-trained RoBERTa model on the $i$th token.

The augmented sequence $\boldsymbol{x}'$ will be semantically similar to the original sequence $\boldsymbol{x}$, and the replaced tokens in $\boldsymbol{x}'$ that are different from those in $\boldsymbol{x}$ are likely to be label-preserving because PLMs are good at predicting missing words in the given context, which are usually interchangeable to the original ones. To further enforce the label-preserving constraint of the augmented sequence, we (1) sample $x'_i$ only from the top-5 terms given by $p_i^{\text{MLM}}\left(\hat{\boldsymbol{x}}; \boldsymbol{\theta}_{\text{PRE}}\right)$ to avoid low-quality replacements, and (2) require $x'_i$ to have the same capitalization and tokenization with $x_i$ (*i.e.*, if $x_i$ is capitalized or is a subword, so should $x'_i$).

Using PLMs to perform augmentation for NER has the major benefit of being *unsupervised* and *contextualized*. Without PLMs, one may still perform augmentation by replacing an entity in the sequence with another of the same type in the distant supervision source (Dai and Adel, 2020). However, such an approach requires prior knowledge about the entity type in the sequence (*i.e.*, it does not work for non-entities or entities not matched with distant labels), and the augmentation is context-free, which may create low-quality and invalid sequences (*e.g.*, it does not fit the context to replace a technology company with a news agency although they both belong to the "*organization*" entity type).

**Self-Training.** The goals of self-training (ST) are two-fold: (1) use the model's high-confidence

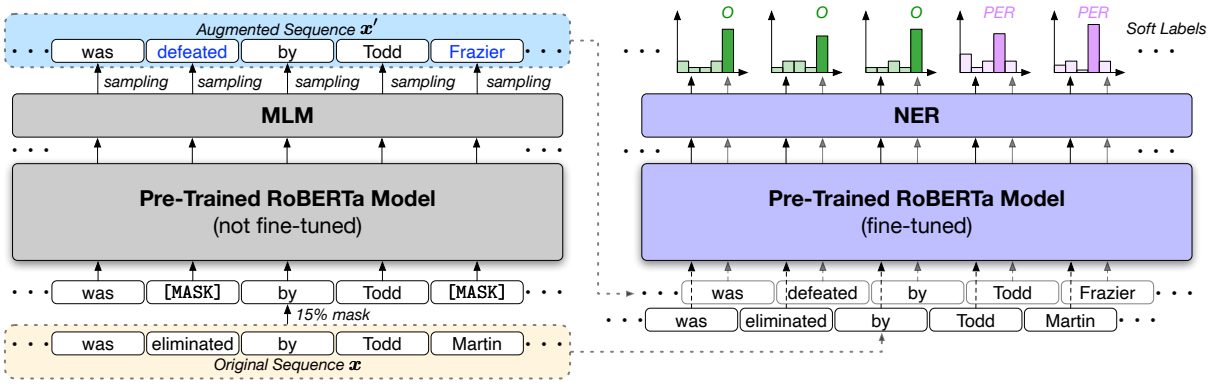


Figure 2: Overview of language model augmented self-training. Only a part of the sequence is shown; the original sequence is "Renzo Furlan was eliminated by Todd Martin in the tournament." We feed the partially masked original sequence into a pre-trained RoBERTa model and sample from its MLM output probability to obtain an augmented sequence (replaced tokens are marked in blue). Then the NER model is trained with both original and augmented sequences as inputs to approximate the soft labels.

predictions that are likely to be reliable for guiding the model refinement on all tokens, and (2) encourage the model to generate consistent predictions on original sequences and augmented ones, based on the principle that a generalizable model should produce similar predictions for similar inputs. To fulfill these goals, we iteratively use the model's current predictions to derive soft labels and gradually update the model so that its predictions on both the original and the augmented sequences approximate the soft labels.

Specifically, at the beginning of self-training, the model $\boldsymbol{\theta}^{(0)}$ is initialized to be the model trained with Eq. (6). Then at each iteration $t$, we derive new soft labels $\boldsymbol{y}^{(t+1)}$ that enhance high-confidence predictions while demote low-confidence ones via squaring and normalizing the current predictions on the original sequence $\boldsymbol{x}$, following the soft labeling formula by (Xie et al., 2016):

$$y_{i,j}^{(t+1)} = \frac{f_{i,j}\left(\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right)^2 / g_j}{\sum_{j'}\left(f_{i,j}\left(\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right)^2 / g_{j'}\right)}, \quad (8)$$
$$g_j = \sum_i f_{i,j}\left(\boldsymbol{x};\boldsymbol{\theta}^{(t)}\right).$$

Then the model $\boldsymbol{\theta}^{(t+1)}$ at the next iteration is updated by approximating the soft labels with both the original sequence and the augmented sequence as inputs, via the KL divergence loss:

$$\mathcal{L}_{\text{ST}} = \sum_{i=1}^{n} \text{KL}\left(y_i^{(t+1)} \middle\| f_i\left(\boldsymbol{x};\boldsymbol{\theta}^{(t+1)}\right)\right) + \sum_{i=1}^{n} \text{KL}\left(y_i^{(t+1)} \middle\| f_i\left(\boldsymbol{x}';\boldsymbol{\theta}^{(t+1)}\right)\right). \quad (9)$$

Using Eq. (9) to train the model not only guides the model learning with its high-confidence predictions, but also encourages consistent predictions between the original and augmented sequences. From another perspective, such a training process gradually propagates confident label information from original examples to augmented ones so that the model is trained with more data for better generalization.

We note that the soft labels in Eq. (8) are computed on all entity types, excluding the non-entity class (*i.e.*, the "O" class). This is because the "O" class usually has many more tokens than any entity type class, while Eq. (8) encourages balanced assignments of target soft labels.

### 2.4 Overall Algorithm

We summarize the entire training procedure in Algorithm 1. Lines 2-9: We train $K$ models with different seeds on the distantly-labeled data using the noise-robust loss. At the end of each iteration, noisy labels are removed based on the model's predictions (Line 9). Line 10: An ensembled model is trained. Line 12: Contextualized augmentations are created with the pre-trained RoBERTa model. Lines 16-18: The soft labels are iteratively computed, and the model is updated to approximate the soft labels on both the original and the augmented sequences.

## 3 Experiments

### 3.1 Datasets

We use three benchmark datasets for NER: CoNLL03 (Sang and De Meulder, 2003), OntoNotes5.0 (Weischedel et al., 2013) which we

**Algorithm 1: RoSTER** training.

**Input:** An unlabeled text corpus $\{\boldsymbol{x}\}$;
external knowledge bases $\Phi$;
pre-trained RoBERTa model $\boldsymbol{\theta}_{\text{PRE}}$.
**Output:** A trained NER model $\boldsymbol{\theta}$.

$\{\boldsymbol{y}\} \leftarrow$ Distant label generation with $\Phi$ ;
// Train $K$ models for ensemble;
**for** $k \leftarrow 0$ *to* $K-1$ **do**
  $\boldsymbol{\theta}_k \leftarrow \boldsymbol{\theta}_{\text{PRE}}$;
  $\{w_i\}_{i=1}^n \leftarrow 1$;
  // Train for $M$ iterations;
  **for** $m \leftarrow 1$ *to* $M$ **do**
    $\boldsymbol{\theta}_k \leftarrow$ Train with Eq. (5);
    $\{w_i \leftarrow \mathbb{1}\left(f_{i,y_i}(\boldsymbol{x}; \boldsymbol{\theta}_k) > \tau\right)\}_{i=1}^n$;
$\boldsymbol{\theta}_{\text{ENS}} \leftarrow$ Train with Eq. (6);
// Augmentation;
$\{\boldsymbol{x}'\} \leftarrow$ Eq. (7);
// Self-training;
$\boldsymbol{\theta}^{(0)} \leftarrow \boldsymbol{\theta}_{\text{ENS}}$;
// Train for $T$ iterations;
**for** $t \leftarrow 0$ *to* $T-1$ **do**
  $\boldsymbol{y}^{(t+1)} \leftarrow$ Eq. (8);
  $\boldsymbol{\theta}^{(t+1)} \leftarrow$ Train with Eq. (9);
Return $\boldsymbol{\theta} = \boldsymbol{\theta}^{(T)}$;

| Dataset | # Types | # Train | # Test |
|---|---|---|---|
| CoNLL03 | 4 | 14,041 | 3,453 |
| OntoNotes5.0 | 18 | 59,924 | 8,262 |
| Wikigold | 4 | 1,142 | 274 |

Table 1: Dataset statistics with the number of entity types and the number of training/test sequences.

follow the pre-processing of (Chiu and Nichols, 2016), and Wikigold (Balasuriya et al., 2009). The dataset statistics are shown in Table 1. All datasets are in English language.

### 3.2 Compared Methods

We compare with a wide range of state-of-the-art distantly-supervised methods and supervised methods. Fully supervised methods use the ground truth training set for model training. Distantly-supervised methods use the distantly-labeled training set obtained as in (Liang et al., 2020). All methods are evaluated on the test set.

**Distantly-supervised methods:**

- **Distant Match**: This is the baseline that reports the distant supervision quality (*i.e.*, compares distantly-labeled results with the ground truth).
- **Distant RoBERTa**: We fine-tune a pre-trained RoBERTa model on distantly-labeled data as if they are ground truth with the standard supervised learning.
- **AutoNER** (Shang et al., 2018): It trains the neural model with a "Tie or Break" tagging scheme. Ambiguous tokens are assigned with all possible labels.
- **BOND** (Liang et al., 2020): It first trains a RoBERTa model on distantly-labeled data with early stopping, and then uses a teacher-student framework to iteratively self-train the model.

**Supervised methods:**

- **BiLSTM-CNN-CRF** (Ma and Hovy, 2016): It was one of the state-of-the-art NER models before the appearance of pre-trained language models, using bidirectional LSTM, CNN and CRF. It is trained from scratch on the training data without any pre-trained knowledge.
- **RoBERTa**: We fine-tune a pre-trained RoBERTa model on the ground truth training data.

### 3.3 Experiment Settings

We use the pre-trained RoBERTa-base model as the backbone model (for our method and baselines). For the three datasets CoNLL03, OntoNotes5.0, and Wikigold, the maximum sequence lengths are set to be 150, 180, and 120 tokens. For all three datasets: The training batch size is 32; the hyper-parameters $\tau$ and $q$ used by Eq. (5) are both set as 0.7; the number of models for ensemble $K = 5$; we use Adam (Kingma and Ba, 2015) as the optimizer. The peak learning rate is $3e-5$, $1e-5$ and $5e-7$ for noise-robust training, ensemble model training and self-training, respectively, with linear decay. The model is run on 2 NVIDIA GeForce GTX 1080 Ti GPUs. More implementation details can be found at Appendix C.

### 3.4 Main Results

Table 2 presents the performance of all methods measured by precision, recall and F1 scores. On all three datasets, **RoSTER** achieves the best performance among distantly-supervised methods. Specifically, (1) the Distant RoBERTa baseline only slightly improves the distant labeling results,

| | Methods | CoNLL03 | | | OntoNotes5.0 | | | Wikigold | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | **Pre.** | **Rec.** | **F1** | **Pre.** | **Rec.** | **F1** | **Pre.** | **Rec.** | **F1** |
| Distant-Sup. | **Distant Match** | 0.811 | 0.638 | 0.714 | 0.745* | 0.693* | 0.718* | 0.479 | 0.476 | 0.478 |
| | **Distant RoBERTa** | 0.837* | 0.633* | 0.721* | 0.760* | 0.715* | 0.737* | 0.603* | 0.532* | 0.565* |
| | **AutoNER** | 0.752 | 0.604 | 0.670 | 0.731* | 0.712* | 0.721* | 0.435 | 0.524 | 0.475 |
| | **BOND** | 0.821 | 0.809 | 0.815 | **0.774*** | 0.701* | 0.736* | 0.534 | 0.686 | 0.600 |
| | **RoSTER (Ours)** | **0.859** | **0.849** | **0.854** | 0.753 | **0.789** | **0.771** | **0.649** | **0.710** | **0.678** |
| Sup. | **BiLSTM-CNN-CRF** | 0.914 | 0.911 | 0.912 | 0.888* | 0.887* | 0.887* | 0.554 | 0.543 | 0.549 |
| | **RoBERTa** | 0.906* | 0.917* | 0.912* | 0.886* | 0.890* | 0.888* | 0.853 | 0.876 | 0.864 |

Table 2: Performance of all methods on three datasets measured by precision (Pre.), recall (Rec.) and F1 scores. Baseline results marked with * are our own runs; others are reported by (Liang et al., 2020).

| **Ablations** | **Pre.** | **Rec.** | **F1** |
|---|---|---|---|
| **RoSTER** | 0.859 | 0.849 | 0.854 |
| **w/o GCE** | 0.817 | 0.843 | 0.830 |
| **w/o NR** | 0.830 | 0.836 | 0.833 |
| **w/o ST** | 0.844 | 0.812 | 0.828 |

Table 3: Ablation study on CoNLL03 dataset. We compare our full method with ablations (see texts for the abbreviation meanings).

| **Ablations** | **Mean (Std.) F1** |
|---|---|
| **w. ensemble** | 0.828 (0.009) |
| **w/o ensemble** | 0.817 (0.025) |

Table 4: Mean and standard deviation (std.) F1 scores of 5 runs (before self-training) with and without model ensemble on CoNLL03 dataset.

showing that directly applying supervised learning to distantly-labeled data will lead to overfitting to label noise and poor model generalization; (2) **RoSTER** consistently outperforms AutoNER and BOND, demonstrating the superiority of our proposed noise-robust learning and self-training approach when trained on distantly-labeled data.

For further comparison with supervised methods, we vary the number of ground truth training sequences used for supervised RoBERTa, and show its performance in Figure 3(a). The performance of **RoSTER** is equivalent to using $1,000$ cleanly annotated sequences for supervised RoBERTa.

### 3.5 Ablation Study

To further validate the effectiveness of each component, we compare **RoSTER** with the following ablations by removing one component at a time: (1) replace the GCE loss in Eq. (5) with cross entropy loss (**w/o GCE**); (2) do not perform noisy label removal (**w/o NR**); (3) do not perform self-training (**w/o ST**). The results are shown in Table 3. It can be seen that **w/o GCE** and **w/o NR** both lead to worse performance than the full model, confirming the necessity of jointly using both designs in noise-robust learning; **w/o ST** also reduces performance, showing that bootstrapping the model with its own high-confidence predictions benefits the model's generalization.

We also study the effect of model ensemble by running noise-robust training (without subsequent self-training) with 5 different seeds and reporting the mean and standard deviation F1 in Table 4. Ensembling multiple models slightly improves the mean result and greatly reduces the variance.

### 3.6 Parameter Study

We study the effect of two important hyperparameters $q$ and $\tau$ used in Eq. (5) on the model performance. We separately vary the value of $q$ or $\tau$ in range $[0.1, 0.3, 0.5, 0.7, 0.9]$ while keeping the other's value as default (both use $0.7$ as the default value). The change in model performance (measured by F1) is shown in Figure 3(b). Overall, the performance is rather insensitive to the two hyperparameters in the $0.5 - 0.9$ range. When $q \to 1$, $\mathcal{L}_{\text{GCE}}$ approximates $\mathcal{L}_{\text{MAE}}$, having good noise-robustness but poor convergence effectiveness; when $q \to 0$, $\mathcal{L}_{\text{GCE}}$ approximates $\mathcal{L}_{\text{CE}}$, having good convergence but weak noise-robustness. Setting $q = 0.7$ achieves a good balance between convergence and noise-robustness. When $\tau \to 0$, all distant labels will be used for model training, and the model performance will suffer from the noise in them; when $\tau \to 1$, many correct labels will be removed, and there will be insufficient training data. Setting $\tau = 0.7$ allows removing noisy

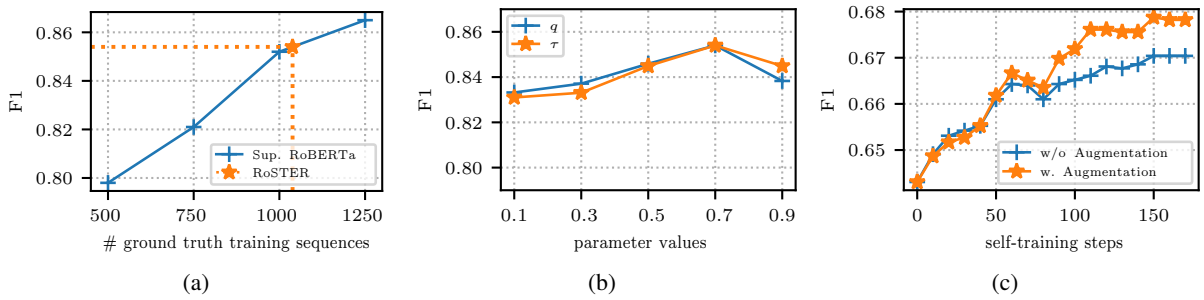


Figure 3: (a) (On CoNLL03) Supervised RoBERTa with different number of ground truth training sequences. **RoSTER** trained on distantly-labeled data is equivalent to supervised RoBERTa using around $1,000$ ground truth sequences. (b) (On CoNLL03) Parameter study. (c) (On Wikigold) Self-training with and without augmentation.

| |
|---|
| **Original**: Swiss Bank Corp sets warrants on DTB-Bund-Future.<br>**Augmentation**: Swiss Investment Corp sets warrants for HTB-Bund-Future. |
| **Original**: Chelsea Clinton was carefully shielded from the exposure of public life.<br>**Augmentation**: Hillary Clinton was largely shielded from the spotlight of public life. |

Table 5: Original sequences and generated augmentations. Replaced words are marked in blue.

labels while keeping enough reliable training data.

### 3.7 Study of Augmentation

We study the effectiveness of the generated contextualized augmentations for the self-training step. We run the self-training step with and without using the augmentations (*i.e.*, including or excluding the second term in Eq. (9)) on the Wikigold dataset and show the results in Figure 3(c). Even without augmentations, the self-training improves the model by using high-confidence predictions for self-refinement; with augmentations, the model is trained with more data and eventually generalizes better with higher test set performance. Two concrete augmentation examples are shown in Table 5.

### 3.8 Case Study

Finally, we perform case study to understand the advantage of **RoSTER** with a concrete example in Table 6. We show the prediction of **AutoNER**, **BOND** and **RoSTER** on a training sequence with label noise. **AutoNER** mainly learns from the given distant labels and slightly generalizes (labels "China" separately as a location entity); **BOND** is able to generalize better for more complete entity detection because it has a self-training step that bootstraps the model on the training set without completely overfitting to distant labels; however, it is still impacted by label noise. **RoSTER** is able to detect the noisy labels via the noise-robust learning step, and then it further re-estimates the true labels in the self-training step with the help of the reliable signals it learns from the clean data as well as the pre-trained knowledge from PLMs via augmentation, instead of relying purely on distant labels.

## 4 Related Work

The effectiveness of deep neural models for NER usually comes with the cost of annotating large amounts of training data. To alleviate the human annotation burden when applying deep models, several studies propose to train NER models with weakly/distantly-labeled data. For weakly-supervised NER, previous studies have explored cross lingual knowledge transfer from high resource languages to low resource languages (Feng et al., 2018; Ni et al., 2017; Xie et al., 2018), aggregating multiple weak labeling functions (Lison et al., 2020; Safranchik et al., 2020) or leveraging sentence-level labels (Kruengkrai et al., 2020). Few-shot approaches (Huang et al., 2021) have also been explored to leverage very few labeled data for NER model training.

Our work is more closely related to distantly-supervised NER which uses external gazetteers or knowledge bases to automatically derive entity labels. Along this line, different methods have been proposed to leverage the distant supervision, such as propagating reliable type information on graphs (Ren et al., 2015), designing new model components to handle multiple possible labels (Shang et al., 2018), employing additional models to classify noisy data (Onoe and Durrett,

| |
|---|
| **Distant Match**: Shanghai-Ek [Chor]$_{\text{PER}}$ is jointly owned by the Shanghai Automobile Corporation and [Ek Chor]$_{\text{PER}}$ China Motorcycle.<br>**Ground Truth**: [Shanghai-Ek Chor]$_{\text{ORG}}$ is jointly owned by the [Shanghai Automobile Corporation]$_{\text{ORG}}$ and [Ek Chor China Motorcycle]$_{\text{ORG}}$. |
| **AutoNER**: Shanghai-Ek [Chor]$_{\text{PER}}$ is jointly owned by the Shanghai Automobile Corporation and [Ek Chor]$_{\text{PER}}$ [China]$_{\text{LOC}}$ Motorcycle.<br>**BOND**: [Shanghai-Ek Chor]$_{\text{PER}}$ is jointly owned by the [Shanghai]$_{\text{LOC}}$ [Automobile Corporation]$_{\text{ORG}}$ and [Ek Chor]$_{\text{PER}}$ [China Motorcycle]$_{\text{ORG}}$.<br>**RoSTER**: [Shanghai-Ek Chor]$_{\text{ORG}}$ is jointly owned by the [Shanghai Automobile Corporation]$_{\text{ORG}}$ and [Ek Chor China Motorcycle]$_{\text{ORG}}$. |

Table 6: Case study with **RoSTER** and baselines. The sentence is from CoNLL03.

2019), formulating the task as a positive-unlabeled learning problem (Peng et al., 2019), and adopting early stopping to prevent the model from overfitting to distant labels (Liang et al., 2020). However, previous methods either do not explicitly address the noise in the distantly-labeled data (*i.e.*, treating them as if they are ground truth), or require an additional set of manually-labeled data to train a denoising model. Our method addresses the label noise with a noise-robust learning scheme and a self-training step for better generalization, without using any ground truth data.

Our study is also related to data augmentation techniques. In NLP, data augmentation is well developed for text classification, by either creating real text sequences (Xie et al., 2020) via back translation (Sennrich et al., 2016) or in the hidden states of the model via perturbations (Miyato et al., 2017) or interpolations by mixing up labels (Chen et al., 2020). However, these techniques cannot be readily used for the NER task. (Dai and Adel, 2020) study a set of simple augmentation methods for the NER task, like synonym replacement, mention replacement or segment shuffling. Nevertheless, these augmentations are context-free which may generate unreasonable sequences or require additional sources like the WordNet. Our proposed augmentation method is unsupervised and contextualized, generating high-quality sequences thanks to the pre-trained knowledge of PLMs and reliably improving model generalization.

## 5 Conclusion and Future Work

In this paper, we study the distantly-supervised NER problem without using any human annotations but only distantly-labeled data. For better model training with noisy data, we propose a noise-robust learning scheme, consisting of a new loss function and a noisy label removal step. To further improve the model generalization, we propose a self-training method that guides model refinement with its own high-confidence predictions and enforces the model to make consistent predictions on original and augmented sequences generated by PLMs. Our method achieves strong performance on three benchmark datasets, outperforming previous distantly-supervised NER methods.

The techniques proposed in this paper are generalizable for future studies: The noise-robust learning scheme may also be applied to other NLP problems where labels may contain noise (*e.g.*, obtained via crowdsourcing from non-experts); the augmentation and self-training method may be helpful for other settings like semi-supervised or few-shot learning. One may also consider exploring larger pre-trained language models (*e.g.*, RoBERTa-large) or more recent pre-trained language models (*e.g.*, COCO-LM (Meng et al., 2021)) for the distantly-supervised NER task.

## Acknowledgments

Research was supported in part by US DARPA KAIROS Program No. FA8750-19-2-1004, SocialSim Program No. W911NF-17-C-0099, and INCAS Program No. HR001121C0165, National Science Foundation IIS-19-56151, IIS-17-41317, and IIS 17-04532, and the Molecule Maker Lab Institute: An AI Research Institutes program supported by NSF under Award No. 2019897. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily represent the views, either expressed or implied, of DARPA or the U.S. Government. We thank anonymous reviewers for valuable and insightful feedback.

## References


Eiji Aramaki, Yasuhide Miura, Masatsugu Tonoike, Tomoko Ohkuma, Hiroshi Masuichi, and Kazuhiko Ohe. 2009. Text2table: Medical text summarization system based on named entity recognition and modality identification. In *BioNLP*.

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

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 2018*.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. MixText: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *ACL*.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *TACL*.

Xiang Dai and Heike Adel. 2020. An analysis of simple data augmentation for named entity recognition. In *COLING*.

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

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*.

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*.

Davide Ferrari, Yuhong Yang, et al. 2010. Maximum lq-likelihood estimation. *The Annals of Statistics*.

Aritra Ghosh, Himanshu Kumar, and PS Sastry. 2017. Robust loss functions under label noise for deep neural networks. In *AAAI*.

Jiaxin Huang, Chunyuan Li, Krishan Subudhi, Damien Jose, Shobana Balakrishnan, Weizhu Chen, Baolin Peng, Jianfeng Gao, and Jiawei Han. 2021. Few-shot named entity recognition: An empirical baseline study. In *EMNLP*.

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

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *TACL*.

Mahboob Alam Khalid, Valentin Jijkoun, and Maarten De Rijke. 2008. The impact of named entity normalization on information retrieval for question answering. In *ECIR*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Canasai Kruengkrai, Thien Hai Nguyen, Sharifah Mahani Aljunied, and Lidong Bing. 2020. Improving low-resource named entity recognition using joint sentence and token labeling. In *ACL*.

Samuli Laine and Timo Aila. 2017. Temporal ensembling for semi-supervised learning. In *ICLR*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL*.

Zhen-Zhong 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*.

Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. 2020. BOND: BERT-assisted open-domain named entity recognition with distant supervision. In *KDD*.

Pierre Lison, Jeremy Barnes, Aliaksandr Hubin, and Samia Touileb. 2020. Named entity recognition without labelled data: A weak supervision approach. In *ACL*.

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*, abs/1907.11692.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *ACL*.

Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2018. Weakly-supervised neural text classification. In *CIKM*.

Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2019. Weakly-supervised hierarchical text classification. In *AAAI*.

Yu Meng, Chenyan Xiong, Payal Bajaj, Saurabh Tiwary, Paul Bennett, Jiawei Han, and Xia Song. 2021. COCO-LM: Correcting and contrasting text sequences for language model pretraining. *arXiv preprint arXiv:2102.08473*.

Yu Meng, Yunyi Zhang, Jiaxin Huang, Chenyan Xiong, Heng Ji, Chao Zhang, and Jiawei Han. 2020. Text classification using label names only: A language model self-training approach. In *EMNLP*.

Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *ICLR*.

Duc Tam Nguyen, Chaithanya Kumar Mummadi, Thi Phuong Nhung Ngo, Thi Hoai Phuong Nguyen, Laura Beggel, and Thomas Brox. 2019. SELF: Learning to filter noisy labels with self-ensembling. In *ICLR*.

Jian Ni, Georgiana Dinu, and Radu Florian. 2017. Weakly supervised cross-lingual named entity recognition via effective annotation and representation projection. In *ACL*.

Yasumasa Onoe and Greg Durrett. 2019. Learning to denoise distantly-labeled data for entity typing. In *NAACL*.

Minlong Peng, Xiaoyu Xing, Qi Zhang, Jinlan Fu, and Xuan-Jing Huang. 2019. Distantly supervised named entity recognition using positive-unlabeled learning. In *ACL*.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? In *EMNLP*.

Xiang Ren, Ahmed El-Kishky, Chi Wang, Fangbo Tao, Clare R Voss, and Jiawei Han. 2015. ClusType: Effective entity recognition and typing by relation phrase-based clustering. In *KDD*.

Esteban Safranchik, Shiying Luo, and Stephen Bach. 2020. Weakly supervised sequence tagging from noisy rules. In *AAAI*.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *ACL*.

Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary. In *EMNLP*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*.

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*.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A Smith, and Jaime G Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *EMNLP*.

Junyuan Xie, Ross B. Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *ICML*.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. In *NeurIPS*.

Zhilu Zhang and Mert R Sabuncu. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. In *NeurIPS*.

## A Generalized Cross Entropy

The Generalized Cross Entropy (GCE) loss (Eq. (3), also shown below),

$$\mathcal{L}_{\text{GCE}} = \sum_{i=1}^{n} \frac{1 - f_{i,y_i}(\boldsymbol{x};\boldsymbol{\theta})^q}{q},$$

is a generalized version of the cross entropy (CE) loss, as $q \to 0$, $\mathcal{L}_{\text{GCE}} \to \mathcal{L}_{\text{CE}}$, shown as follows:

$$\begin{aligned}
\lim_{q\to 0} \mathcal{L}_{\text{GCE}} &= \lim_{q\to 0} \sum_{i=1}^{n} \frac{1 - f_{i,y_i}(\boldsymbol{x};\boldsymbol{\theta})^q}{q} \\
&= \lim_{q\to 0} \sum_{i=1}^{n} \frac{\frac{d}{dq}\left(1 - f_{i,y_i}(\boldsymbol{x};\boldsymbol{\theta})^q\right)}{\frac{d}{dq}(q)} \\
&= \lim_{q\to 0} \sum_{i=1}^{n} \frac{-f_{i,y_i}(\boldsymbol{x};\boldsymbol{\theta})^q \log f_{i,y_i}(\boldsymbol{x};\boldsymbol{\theta})}{1} \\
&= -\sum_{i=1}^{n} \log f_{i,y_i}(\boldsymbol{x};\boldsymbol{\theta}) \\
&= \mathcal{L}_{\text{CE}}.
\end{aligned}$$

The second line is obtained by applying L'Hôpital's rule.

## B Baseline Sources

We use the following sources for baseline implementation:

- **Distant RoBERTa**: We use the Huggingface Transformer library for the RoBERTa model: https://huggingface.co/transformers/.
- **AutoNER**: We use the authors' released code: https://github.com/shangjingbo1226/AutoNER.
- **BOND**: We use the authors' released code: https://github.com/cliang1453/BOND/.

The results reported in Table 2 are obtained by taking the higher value of (1) our own run and (2) the corresponding performance reported in (Liang et al., 2020).

## C Implementation Details

**Tagging Scheme for Distantly-Supervised NER.** Instead of using BIO/BIOES tagging scheme, we use the binary IO format (*i.e.*, only distinguish whether a token is a part of an entity or not) following previous work (Peng et al., 2019), mainly because the distant labeling process may induce partially matched entities (*e.g.*, the first sentence in Figure 1), and the beginning/ending token of the entity can be inaccurate.

**Dropping Non-Entity Tokens From Distant Labels.** We find it beneficial to randomly exclude a portion of distantly-labeled non-entity tokens (we dropped $50\%$ non-entity tokens for all three datasets in the experiments) from training. This is probably because the distant labeling process fails to detect some entities which will be mislabeled as the "O" class, and randomly dropping non-entity tokens reduces the number of such false negative labels.

**Implementation of the NER Head.** Different from the common setup of fine-tuning PLMs for NER, we implement the NER head in Figure 2 as two linear layers instead of one: One linear layer classifies entity tokens against non-entity ones (*i.e.*, binary classification), and the other linear layer classifies all entity type classes. During the self-training step, the former is trained to maintain its predictions, while the latter is trained to approximate the soft labels.

**Noisy Label Removal for Minority Types.** Some minority types may have very few labeled tokens and the model will output low-confidence predictions on them due to insufficient training. To make sure those tokens are not treated as noisy ones and removed from training, we do not perform noisy label removal on entity type classes (*i.e.*, keep all tokens labeled as those classes) of which $> 90\%$ tokens are predicted with confidence lower than the threshold.