Robust Self-Augmentation for Named Entity Recognition with Meta Reweighting

Linzhi Wu[1*], Pengjun Xie[*], Jie Zhou[2*], Meishan Zhang[3†], Chunping Ma, Guangwei Xu, Min Zhang[3]

[1]School of New Media and Communication, Tianjin University
[2]School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University
[3]Institute of Computing and Intelligence, Harbin Institute of Technology (Shenzhen)

{tjuwlz2020, machunpingjj}@163.com, sanny02@sjtu.edu.cn
{xpjandy, mason.zms, ahxgwOnePiece}@gmail.com, zhangmin2021@hit.edu.cn

Abstract

Self-augmentation has received increasing research interest recently to improve named entity recognition (NER) performance in low-resource scenarios. Token substitution and mixup are two feasible heterogeneous self-augmentation techniques for NER that can achieve effective performance with certain specialized efforts. Noticeably, self-augmentation may introduce potentially noisy augmented data. Prior research has mainly resorted to heuristic rule-based constraints to reduce the noise for specific self-augmentation methods individually. In this paper, we revisit these two typical self-augmentation methods for NER, and propose a unified meta-reweighting strategy for them to achieve a natural integration. Our method is easily extensible, imposing little effort on a specific self-augmentation method. Experiments on different Chinese and English NER benchmarks show that our token substitution and mixup method, as well as their integration, can achieve effective performance improvement. Based on the meta-reweighting mechanism, we can enhance the advantages of the self-augmentation techniques without much extra effort.

1 Introduction

Named entity recognition (NER), which aims to extract predefined named entities from a piece of unstructured text, is a fundamental task in the natural language processing (NLP) community, and has been studied extensively for several decades (Hammerton, 2003; Huang et al., 2015; Chiu and Nichols, 2016; Ma and Hovy, 2016). Recently, supervised sequence labeling neural models have been exploited most popularly for NER, leading to state-of-the-art (SOTA) performance (Zhang and Yang, 2018; Li et al., 2020a; Ma et al., 2020).

Although great progress has been made, developing an effective NER model usually requires a

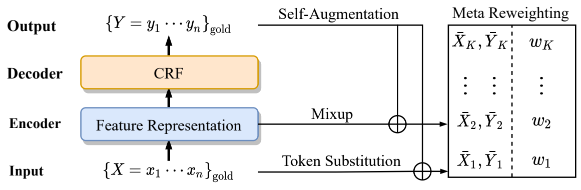
Output
$\{Y = y_1 \cdots y_n\}_{\text{gold}}$
Self-Augmentation
Meta Reweighting
Decoder
CRF
Encoder
Feature Representation
Mixup
Input
$\{X = x_1 \cdots x_n\}_{\text{gold}}$
Token Substitution
$\tilde{X}_K, \tilde{Y}_K$
w_K
$\tilde{X}_2, \tilde{Y}_2$
w_2
$\tilde{X}_1, \tilde{Y}_1$
w_1

Figure 1: The main idea of our work, where the two heterogeneous self-augmentation methods (i.e., token substitution and mixup) are integrated by a unified meta reweighting framework.

large-scale and high-quality labeled training corpus, which is often difficult to be obtained in real-world scenarios due to the expensive and time-consuming annotations by human experts. Moreover, it would be extremely serious because the target language, target domain, and the desired entity type could all be infinitely varied. As a result, the low-resource setting with only a small amount of annotated corpus available is far more common in practice, even though it may result in significant performance degradation due to the overfitting problem.

Self-augmentation is a prospective solution to this problem, which has received widespread attention (Zhang et al., 2018; Wei and Zou, 2019; Dai and Adel, 2020; Chen et al., 2020; Karimi et al., 2021). The major motivation is to generate a pseudo training example set deduced from the original gold-labeled training data automatically. For NER, a token-level task, the feasible self-augmentation techniques include token substitution (Dai and Adel, 2020; Zeng et al., 2020) and mixup (Zhang et al., 2020a; Chen et al., 2020), which are deformed at the ground-level inputs and the high-level hidden representations, respectively.

Nonetheless, there are still some limitations currently for the above token substitution and mixup methods. For one thing, both of them require some specialized efforts to improve their effectiveness due to the potential noise introduced by the self-augmentation, which may restrict the valid seman-

[*]Equal contributions.
[†]Corresponding author.

tic representation of the augmented data. For instance, token substitution is typically limited to the named entities in the training corpus (Wu et al., 2019), and the mixup tends to be imposed on the example pairs with small semantic distance gaps (Chen et al., 2020). For another thing, though the two techniques seem to be orthogonal and probably complementary to each other, it remains a potential challenge to effectively and naturally integrate them.

In this work, we revisit the token substitution and mixup methods for NER, and investigate the two heterogeneous techniques under a unified meta reweighting framework (as illustrated in Figure 1). First, we try to relax the previous constraints to a broader scope for these methods, allowing for more diverse and larger-scale pseudo training examples. However, this would inevitably produce some low-quality augmented examples (i.e., noisy pseudo data) in terms of linguistic correctness, which may negatively affect the model performance. To this end, we present a meta reweighting strategy for controlling the quality of the augmented examples and leading to noise-robust training. Also, we can naturally integrate the two methods by using the example reweighting mechanism, without any specialization in a specific self-augmentation method.

Finally, we carry out experiments on several Chinese and English NER benchmark datasets to evaluate our proposed methods. We mainly focus on the low-resource settings, which can be simulated by using only part of the standard training set when the scale is large. Experimental results show that both our token substitution and mixup method coupled with the meta-reweighting can effectively improve the performance of our baseline model, and the combination can bring consistent improvement. Positive gains become more significant as the scale of the training data decreases, indicating that our self-augmentation methods can handle the low-resource NER well. In addition, our methods can still work even with a large amount of training data. The code is available at `https://github.com/LindgeW/MetaAug4NER`.

2 Our Approach

In this section, we firstly describe our baseline model. Then, we present our self-augmentation methods to enhance the baseline model in the low-resource settings. Finally, we elaborate on our meta reweighting strategy, which aims to alleviate the negative impact of the noisy augmented examples caused by the self-augmentation while also elegantly combining these augmentation methods.

2.1 Baseline Model

NER task is typically formulated as a sequence labeling problem, which transforms entities/non-entities into token-level boundary label sequence by using the BIO or BIOES schema (Huang et al., 2015; Lample et al., 2016). In this work, we adopt BERT-BiLSTM-CRF as our basic model architecture which consists of four components: (1) input representation, (2) BiLSTM encoding, (3) CRF decoding, and (4) training objective.

Input Representation Given an input sequence $X = (x_1, \cdots, x_n)$ of length n, we first convert it into sequential hidden vectors using the pre-trained BERT (Devlin et al., 2019):

$$\boldsymbol{e}_1, \cdots, \boldsymbol{e}_n = \text{BERT}(X), \tag{1}$$

where each token is mapped to a contextualized representation correspondingly.

Encoding We use a bidirectional LSTM layer to further extract the contextual representations, where the process can be formalized as:

$$\boldsymbol{h}_1, \cdots, \boldsymbol{h}_n = \text{BiLSTM}(\boldsymbol{e}_1, \cdots, \boldsymbol{e}_n), \tag{2}$$

where $\boldsymbol{h}_i$ is the hidden state output of the i-th token in the sequence ($i \in [1, n]$).

Decoding First, a linear transformation layer is used to calculate the initial label scores. Then, a label transition matrix $\boldsymbol{T}$ is used to model the label dependency. Let $Y = (y_1, \cdots, y_n)$ be a label sequence, the score $s(Y|X)$ can be computed by:

$$\begin{aligned} \boldsymbol{o}_i &= \boldsymbol{W}\boldsymbol{h}_i + \boldsymbol{b}, \\ s(Y|X) &= \sum_{i=1}^{n}(\boldsymbol{T}_{y_{i-1},y_i} + \boldsymbol{o}_i[y_i]), \end{aligned} \tag{3}$$

where $\boldsymbol{W}$, $\boldsymbol{b}$ and $\boldsymbol{T}$ are the model parameters. Finally, we employ the Viterbi algorithm (Viterbi, 1967) to find the best label sequence Y.

Training Objective We exploit the sentence-level cross-entropy objective for training. Given a gold-labeled training example (X, Y), we have the conditional probability $p(Y|X)$ based on the

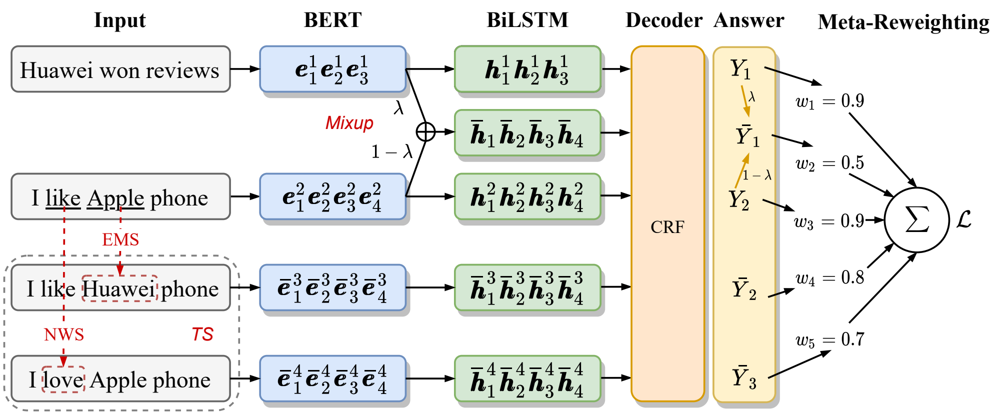

Figure 2: An overview of the self-augmentation framework for NER.

scoring function defined in Equation 3, and then apply a cross-entropy function to calculate the single example loss:

$$p(Y|X) = \frac{\exp\big(s(Y|X)\big)}{\sum_{\widetilde{Y}} \exp\big(s(\widetilde{Y}|X)\big)}, \qquad (4)$$
$$\mathcal{L}(X,Y) = -\log p(Y|X).$$

where $\widetilde{Y}$ denotes the candidate label sequences.

2.2 Self-Augmentation

Self-augmentation methods can reduce the demand for abundant manually-annotated examples, which can be implemented at the input level and representation level. Token substitution and mixup are two popular methods for NER that correspond to these two distinct levels. Here, we try to extend these two self-augmentation methods.

Token Substitution Token substitution aims to generate pseudo examples based on the original gold-labeled training data by replacing the tokens of input sentence with their synonym alternatives (Wu et al., 2019; Dai and Adel, 2020). For NER, Wu et al. (2019) adopted this method to obtain performance gains on Chinese datasets where the substituted objects are limited to named entities. Dai and Adel (2020) empirically demonstrated the superiority of synonym replacement among various augmentation schemes where the synonyms are retrieved from the off-the-shelf WordNet thesaurus.

Our token substitution is performed by building a synonym dictionary, which covers the named entity synonyms as well as numerous normal word synonyms. Following Wu et al. (2019), we treat all entities of the same type from the training set as synonyms, which are added to the entity dictionary. We name it as entity mention substitution (EMS). Meanwhile, we extend the substitution to non-entity tokens (i.e., the corresponding label is 'O'), which is named as normal word substitution (NWS). Since unlabeled data in a specific domain is easily accessible, we adopt the word2vec-based algorithm (Mikolov et al., 2013; Pennington et al., 2014) to mine tokens with similar semantics on Wikidata via distributed word representation (Yamada et al., 2020), and build a normal word synonym dictionary from the k-nearest token set based on cosine similarity distance. Note that this scheme does not require access to thesaurus for a specific domain in order to obtain synonyms.

Figure 2 presents an example of token substitution, where EMS and NWS are both involved. Specifically, for a given gold-labeled training example (X, Y), we replace the entity token of X with a sampled entity from the entity dictionary which has the same entity type, and meanwhile replace the non-entity token of X with a sampled synonym. Then, we can obtain a pseudo example $(\bar{X}, \bar{Y})$. Especially, we balance the EMS and NWS strategies based on a ratio γ by adjusting the percentage of EMS operations, aiming for a good trade-off between entity diversity and context diversity. And, we refer to this method as TS in the rest of this paper for short.

Mixup for CRF Unlike token substitution performed at the ground input, the mixup technique (Zhang et al., 2018) generates virtual examples at the feature representation level in the NLP field (Guo et al., 2019). The main idea is to perform linear interpolations on both the input and ground-truth output between randomly sampled example pairs from the given training set. Chen et al. (2020) presented the first work based on the token classification framework for the NER task, and their mixup strategy is constrained to the examples pairs where the input sentences are semantically similar by using specific heuristic rules. Different from their method, we extend the mixup technique to the CRF decoding.

Formally, give an example pair (X_1, Y_1) and (X_2, Y_2) randomly sampled from the gold-labeled training set, we firstly obtain their vector representations through Equation 1, resulting in $\boldsymbol{e}_{1,1} \cdots \boldsymbol{e}_{1,n_1}$, and $\boldsymbol{e}_{2,1} \cdots \boldsymbol{e}_{2,n_2}$, respectively. Then we apply the linear interpolation to obtain a new virtual training example $(\bar{X}, \bar{Y})$. Here we assume a regularization of pair-wise linear interpolation over the input representations and the output scores, where the following attributes should be satisfied:

$$\begin{aligned} \bar{X}&: \begin{cases} \text{BERT}(\bar{X}) = \bar{\boldsymbol{e}}_1 \cdots \bar{\boldsymbol{e}}_n \\ \bar{\boldsymbol{e}}_i = \lambda \boldsymbol{e}_{1,i} + (1-\lambda)\boldsymbol{e}_{2,i}, i \in [1, n] \end{cases} \\ \bar{Y}&: s(\bar{Y}|\bar{X}) = \lambda s(Y_1|\bar{X}) + (1-\lambda) s(Y_2|\bar{X}), \end{aligned} \tag{5}$$

where $n = \max(n_1, n_2)$ [1] and λ is sampled from a Beta(α, α) distribution ($\lambda \in [0, 1]$ and $\alpha > 0$). According to this formulation, the loss function can be reformulated as:

$$\begin{aligned} \mathcal{L}(\bar{X}, \bar{Y}) &= -\log \frac{\exp\left(s(\bar{Y}|\bar{X})\right)}{\sum_{\widetilde{Y}} \exp\left(s(\widetilde{Y}|\bar{X})\right)} \\ &= \lambda \mathcal{L}(\bar{X}, Y_1) + (1-\lambda)\mathcal{L}(\bar{X}, Y_2). \end{aligned} \tag{6}$$

which aligns with the training objective of Equation 4. In this way, our mixup method can fit well with the structural decoding.

2.3 Meta Reweighting

Although the self-augmentation techniques can efficiently generate numerous pseudo training examples, how to control the quality of augmented examples is a potential challenge that cannot be overlooked. In particular, unlike sentence-level classification tasks, entity recognition is highly sensitive to the semantics of the context. While positive augmented examples can help our model advance, some low-quality augmented examples that are inevitably introduced during self-augmentation may hurt the final model performance.

In this paper, we leverage a meta reweighting mechanism to dynamically and adaptively assign the example-wise weights to each mini-batch of training data, motivated by Ren et al. (2018). The key idea is that a small and clean meta-data set is applied to guide the training of model parameters, and the loss produced by the mini-batch of meta-data is exploited to reweight the augmented examples in each batch online. Intuitively, if the data distribution and gradient-descent direction of the augmented example are similar to those of the sample in the meta-data set, our model could better fit this *positive* augmented sample and increase its weight, and vice versa. In other words, the clean and valid augmented examples are more likely to be fully trained.

Algorithm 1 The training procedure of the meta reweighting strategy

```
Input: Initial model parameters $\Theta^{(0)}$, clean training dataset $\mathcal{D}$, augmented training dataset $\hat{\mathcal{D}}$, batch size $m, n$, training steps $T$
Output: Updated model parameters $\Theta^{(T)}$
 1: for $t = 1$ to $T$ do
 2:     Initialize the trainable parameter $\epsilon$.
 3:     $\{x_c, y_c\} \leftarrow$ SampleMiniBatch($\mathcal{D}$, $m$).
 4:     $\{x_a, y_a\} \leftarrow$ SampleMiniBatch($\hat{\mathcal{D}}$, $n$).
 5:     $\mathcal{L}_a \leftarrow \sum_{i=1}^{n} \epsilon_i \mathcal{L}(f(x_{a,i}; \Theta^{(t)}), y_{a,i})$.
 6:     $\nabla\Theta^{(t)} \leftarrow$ Grad($\mathcal{L}_a, \Theta^{(t)}$).
 7:     $\hat{\Theta}^{(t)} \leftarrow \Theta^{(t)} - \beta\nabla\Theta^{(t)}$.
 8:     $\mathcal{L}_c \leftarrow \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(f(x_{c,i}; \hat{\Theta}^{(t)}), y_{c,i})$.
 9:     $\nabla\epsilon \leftarrow$ Grad($\mathcal{L}_c, \epsilon$).
10:     $\hat{w} \leftarrow$ Sigmoid($-\nabla\epsilon$).
11:     $w \leftarrow \frac{\hat{w}}{\sum_j \hat{w}_j + \delta}$.
12:     $\hat{\mathcal{L}}_a \leftarrow \sum_{i=1}^{n} w_i \mathcal{L}(f(x_{a,i}; \Theta^{(t)}), y_{a,i})$.
13:     $\nabla\Theta^{(t)} \leftarrow$ Grad($\hat{\mathcal{L}}_a, \Theta^{(t)}$).
14:     $\Theta^{(t+1)} \leftarrow$ OptimizerStep($\Theta^{(t)}, \nabla\Theta^{(t)}$).
15: end for
```

More specifically, suppose that we have a set of N augmented training examples $\hat{\mathcal{D}} = \{(X_i, Y_i)\}_{i=1}^{N}$, our final optimizing objective can be formulated as a weighted loss as follows:

$$\Theta^*(w) = \arg\min_{\Theta} \sum_{i=1}^{N} w_i \mathcal{L}(f(X_i; \Theta), Y_i), \tag{7}$$

where $w_i \geq 0$ is the learnable weight for the loss of i-th training example. $f(\cdot; \Theta)$ represents the forward process of our model (with parameter Θ). The optimal parameter w is further determined by minimizing the following loss computed on the meta example set $\mathcal{D} = \{(X_i^m, Y_i^m)\}_{i=1}^{M}$ ($M \ll$

[1] Special zero-vector pads are used to align two sequences with different lengths.

N):

$$w^* = \arg\min_{w} \frac{1}{M} \sum_{i=1}^{M} \mathcal{L}(f(X_i^m; \Theta^*(w)), Y_i^m), \tag{8}$$

Accordingly, we need to calculate the optimal Θ^* and w^* in Equation 7 and 8 based on two nested loops of optimization iteratively. For simplicity and efficiency, we take a single gradient-descent step for each training iteration to update them via an online-approximation manner. At every training step t, we sample a mini-batch augmented examples $\{(X_i, Y_i)\}_{i=1}^{n}$ initialized with the learnable weights ϵ. After a single optimization step, we have:

$$\hat{\Theta}^{(t+1)}(\epsilon) = \Theta^{(t)} - \beta \nabla_{\Theta} \sum_{i=1}^{n} \epsilon_i \mathcal{L}(f(X_i; \Theta), Y_i), \tag{9}$$

where β is the inner-loop step size. Based on the updated parameters, we then calculate the loss of the sampled mini-batch meta examples $\{(X_j^{meta}, Y_j^{meta})\}_{j=1}^{m}$:

$$\mathcal{L}^{meta}(\hat{\Theta}) = \frac{1}{m} \sum_{j=1}^{m} \mathcal{L}(f(X_j^{meta}; \hat{\Theta}^{(t+1)}), Y_j^{meta}), \tag{10}$$

To generalize the parameters $\hat{\Theta}$ well to the meta-data set, we take the gradients of ϵ w.r.t the meta loss to produce example weights and normalize it along mini-batch:

$$\begin{aligned} \hat{w}_i &= \sigma(-\nabla_{\epsilon_i} \mathcal{L}^{meta}(\hat{\Theta})\Big|_{\epsilon_i=0}), \\ w_i &= \frac{\hat{w}_i}{\sum_j \hat{w}_j + \delta}. \end{aligned} \tag{11}$$

where $\sigma(\cdot)$ is the sigmoid function and δ is a small value to avoid division by zero. Finally, we optimize the model parameters over augmented examples with the calculated weights.

Algorithm 1 illustrates the detailed training procedure of the meta reweighting strategy. It is noteworthy that the augmented training examples contain the original clean training examples, which serve as the unbiased meta-data. Since the algorithm execution just requires a clear definition of the training objective for the input examples, it is also well adaptable for the virtual augmented examples generated by our mixup method.

3 Experiments

3.1 Settings

Datasets To validate our methods, we conduct experiments on Chinese benchmarks: OntoNotes 4.0 (Weischedel et al., 2011) and Weibo NER (Peng and Dredze, 2015), as well as English benchmarks: CoNLL 2003 (Sang and Meulder, 2003) and OntoNotes 5.0[2] (Pradhan et al., 2013). The Chinese datasets are split into training, development and test sections following Zhang and Yang (2018) while we take the same data split as Benikova et al. (2014) and Pradhan et al. (2012) on the English datasets. We follow Lample et al. (2016) to use the BIOES tagging scheme for all datasets. The detailed statistics can be found in Table 4.

Dataset	Type	Train	Dev	Test
	#sent	15.7k	4.3k	4.3k
OntoNotes 4	#char	491.9k	200.5k	208.1k
	#entity	12.6k	6.6k	7.3k
	#sent	1.4k	0.27k	0.27k
Weibo	#char	73.8k	14.5k	14.8k
	#entity	1.9k	0.4k	0.4k
	#sent	15.0k	3.5k	3.7k
CoNLL03	#token	203.6k	51.4k	46.4k
	#entity	23.5k	5.9k	5.6k
	#sent	59.9k	8.5k	8.3k
OntoNotes 5	#token	1088.5k	147.7k	152.7k
	#entity	81.8k	11.1k	11.3k

Table 4: Statistics of datasets. #sent and #entity stand for the number of sentences and entity words, respectively.

Implementation Details We use one-layer BiLSTM and the hidden size is set to 768. The dropout ratio is set to 0.5 for the input and output of BiLSTM. Regarding BERT, we adopt BERT-base model[3] (BERT-base-cased for the English NER) and fine-tune the inside parameters together with all other module parameters. We use the AdamW(Loshchilov and Hutter, 2019) optimizer to update the trainable parameters with β_1=0.9 and β_2=0.99. For the BERT parameters, the learning rate is set to $2e{-}5$. For other module parameters excluding BERT, a learning rate of $1e{-}3$ and weight decay of $1e{-}4$ are used. Gradient clipping is used to avoid gradient explosion by a maximum value of 5.0. All the models are trained on NIVIDIA

[2]https://catalog.ldc.upenn.edu/LDC2013T19

[3]https://github.com/huggingface/transformers

Models	**ON 4**			**ON 5**			**CoNLL03**		
	5%	**10%**	**30%**	**5%**	**10%**	**30%**	**5%**	**10%**	**30%**
Baseline	75.07	76.14	80.88	81.22	83.51	86.27	85.12	87.11	89.24
+ TS *w/o* MR	74.58	75.94	79.83	82.12	83.82	86.23	85.93	87.66	89.14
+ TS *w/* MR	76.08	76.85	81.23	82.58	83.92	86.50	86.25	88.00	89.55
+ Mixup *w/o* MR	75.21	76.03	80.00	82.63	83.77	86.04	86.18	87.75	89.48
+ Mixup *w/* MR	76.15	76.75	80.97	82.83	84.12	86.60	86.33	88.03	89.75
+ Both *w/o* MR	76.33	76.91	81.40	82.85	84.33	86.88	86.51	88.10	89.96
+ Both (Final)	**76.82**	**77.13**	**81.66**	**82.98**	**84.52**	**87.09**	**86.76**	**88.25**	**90.12**
Dai and Adel (2020)	75.05	76.75	81.24	82.47	83.90	86.55	86.22	87.86	89.91
Chen et al. (2020)	–	–	–	–	–	–	84.85	87.85	89.87
Chen et al. (2020) (*Semi*)	–	–	–	–	–	–	86.33	88.78	90.25

Table 1: Results on OntoNotes and CoNLL03 using 5%, 10%, and 30% of the training data. *Semi*: additional 10,000 unlabeled training examples are used.

Models	**ON 4**	**Weibo**
Baseline	81.73	69.10
+ TS *w/o* MR	81.38	68.69
+ TS *w/* MR	81.85	69.61
+ Mixup *w/o* MR	81.68	69.96
+ Mixup *w/* MR	82.15	70.53
+ Both *w/* MR	82.33	71.15
+ Both (Final)	**82.48**	**71.42**
Meng et al. (2019)†	81.63	67.60
Hu and Wei (2020)	80.20	64.00
Mengge et al. (2020)	80.60	69.23
Li et al. (2020a)	81.82	68.55
Nie et al. (2020a)†	81.18	69.78
Nie et al. (2020b)	–	69.80
Li et al. (2020b)†	82.11	–
Ma et al. (2020)	82.81	70.50
Xuan et al. (2020)†	82.04	71.25
Liu et al. (2021)	82.08	70.75

Table 2: Performance comparisons using the full training data on OntoNotes 4 (ON 4) and Weibo. Previous SOTA results are also offered for comparisons. † denotes external knowledge is used.

Models	**CoNLL03**	**ON 5**
Baseline	91.23	88.22
+ TS *w/o* MR	90.98	87.55
+ TS *w/* MR	91.64	88.84
+ Mixup *w/o* MR	91.04	87.46
+ Mixup *w/* MR	91.42	88.98
+ Both *w/* MR	91.88	89.24
+ Both (Final)	**92.15**	**89.43**
Chen et al. (2020)	91.83	–
Clark et al. (2018)‡	92.60	88.80
Fisher and Vlachos (2019)	–	89.20
Li et al. (2020b)†	93.04	91.11
Yu et al. (2020)†	93.50	91.30
Xu et al. (2021)†	–	90.85

Table 3: Performance comparisons using the full training data on CoNLL03 and OntoNotes 5 (ON 5). Previous SOTA results are also offered for comparisons. ‡ means the multi-task learning with more unlabeled data. † denotes external knowledge is used.

Tesla V100 (32G) GPUs. The `higher` library[4] is utilized for the implementation of second-order optimization involved in Algorithm 1.

For the NWS, we use the word vectors trained on Wikipedia data[5] based on the GloVe model (Pennington et al., 2014) and build the synonym set for any given non-entity word based on the top-5 cosine similarity, where stop-words are excluded.

As mentioned in Section 2.2, we defined two core hyper-parameters for our self-augmentation methods, one for TS (i.e., γ) and the other for mixup (i.e., λ). Specifically, we set $\gamma = 20\%$ and λ by sampled from the $\mathrm{Beta}(\alpha, \alpha)$ distribution with $\alpha = 7$, where the details will be shown in the analysis section. Meanwhile, we conduct the augmentation up to 5 times corresponding to the original training data.

Evaluation We conduct each experiment by 5 times and report the average F1 score. The best-performing model on the development set is then used to evaluate on the test set.

[4]https://github.com/facebookresearch/higher
[5]https://dumps.wikimedia.org/

3.2 Main Results

The main results are presented in Table 1, 2 and 3, verifying the effectiveness of our method under the low-resource setting and the standard full-scale setting, respectively. Since Weibo is a small-scale dataset, we do not consider its partial training set for the low-resource setting.

Low-Resource Setting We randomly sample 5%, 10%, and 30% of the original training data from OntoNotes and CoNLL03 for simulation studies. Table 1 shows the results where F1 scores of the baseline, +TS, +Mixup, and +Both are reported. We can observe that: (1) the baseline performance will drop significantly as the size of training data get reduced gradually, which demonstrates the performance of the supervised NER model relies heavily on the scale of the labeled training data. (2) although the number of training examples has increased, vanilla self-augmentation (without meta reweighting) might degrade the model performance due to potentially unreliable pseudo-labeled examples. The meta reweighting strategy helps to adaptively weight the augmented examples during training, which combats the negative impact and leads to a stable and positive performance boost.

In addition, as the scale of the training data decreases, the effectiveness of the augmentation methods can be more significant, indicating that our self-augmentation methods are highly beneficial for the low-resource settings, and the two-stage combination of the two heterogeneous methods can yield better performance consistently.

Full-Scale Setting Table 2 and 3 show the results using full-scale training data. The results demonstrate that our baseline model is already strong. The model after vanilla augmentation could perform slightly worse since each training example is treated equally even if it is noisy. This also implies our meta reweighting makes great sense. Furthermore, our final model (+Both) can further achieve performance gains by integrating these self-augmentation methods with the meta reweighting mechanism. The overall trend is similar to the low-resource setting, but the gains are relatively smaller when the training data is sufficient. That may be attributed that the size of training data is large enough to narrow the performance gap between the baseline and augmented models. It also suggests that our method does not hurt the model performance even when using enough training data.

Comparison with Previous Work We also compare our method with previous representative SOTA work, where all referred systems exploit the pre-trained BERT model. As shown, compared to Dai and Adel (2020) and Chen et al. (2020), our method either outperforms or performs on par with theirs when using limited training data. For Chen et al. (2020), the pure mixup performs slightly better due to the well-designed example sampling strategy, but our overall framework outperforms theirs. Moreover, our method can match the performance of the semi-supervised setting that uses additional 10K unlabeled training data. Besides, our final model, without utilizing much external knowledge, can achieve very competitive results on the full training set in comparison to most previous systems.

3.3 Analysis

In this subsection, we further conduct detailed experimental analyses on the CoNLL03 dataset for a better understanding of our method. Our main concern is on the low-resource setting, therefore the models based on 5%, 10% and 30% of the original training data are our main focus.

Augmentation Times The size of augmented examples is an essential factor in final model performance. Typically, we examine the 5% CoNLL03 training data. As illustrated in Figure 3, the larger pseudo examples can obtain better performance in a certain range. However, as the times of augmentation increases, the uptrend of performance slows down. The improvement tends to be stable when the pseudo samples are increased to about 5 times the original training data. Excessively increasing the augmentation times does not necessarily bring consistent performance improvement. And we select an appropriate value for training data of different sizes from a range [1, 8].

Influence of γ for Token Substitution Regarding our TS strategy, we take both NWS and EMS into account simultaneously. The two parts are blended by a percentage parameter γ, namely γ for EMS and $1-\gamma$ for NWS. Here we examine the influence of γ in the sole self-augmentation model by TS. Figure 4 shows the results, where $\gamma = 0$ and $\gamma = 100\%$ denote the model with only NWS and EMS, respectively. As shown, our model can achieve the overall better performance when $\gamma = 20\%$, indicating that both of them are helpful

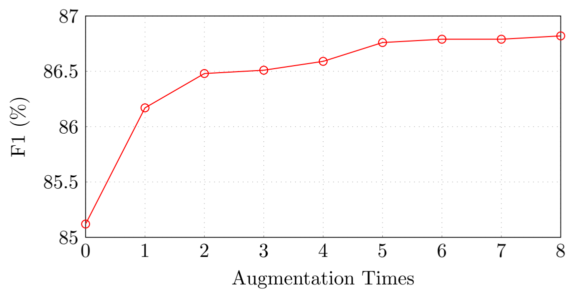

Figure 3: Influence of augmentation times for 5% training data of CoNLL03. Times=0 means original training data without any augmentation.

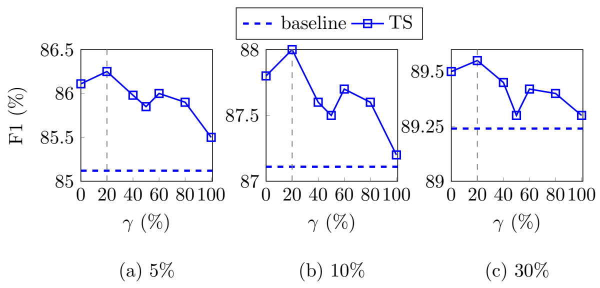

Figure 4: Performance against different EMS rate γ.

for the TS strategy, and NWS can be slightly better. One possible reason is that the entity words in original training examples are relatively sparse (i.e., the 'O' label is dominant), allowing the NWS to produce more diverse pseudo examples.

Mixup Parameter We further inspect the model with the mixup strategy alone so as to understand the important factors of the mixup model. First, we analyze the influence of the mixing parameter α. As depicted in Figure 5, we can see that α indeed affects the effectiveness of the mixup method greatly. Considering the feature of Beta distribution, the sampled λ will be more concentrated around 0.5 as the α value becomes large, resulting in a relatively balanced weight between the mixed example pairs. The model performance remains stable when α is around 7. Second, we study where to conduct the mixup operation since there are two main options in our framework, i.e., the hidden representations of either the BERT or BiLSTM for linear interpolation. Table 5 reports the comparison results, demonstrating the former is a better choice.

Case Study To further understand the effectiveness of the meta-reweighting mechanism, we present several high-quality and low-quality examples in Table 6. As shown, the difference between the positive and negative examples for TS could be

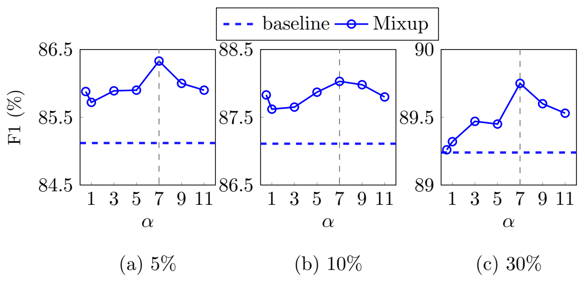

Figure 5: Performance against different mixing parameter by $\mathrm{Beta}(\alpha, \alpha)$ distribution.

Method	**5%**	**10%**	**30%**
Baseline	85.12	87.11	89.24
Mixing on BiLSTM	86.15	87.66	89.41
Mixing on BERT	**86.33**	**88.03**	**89.75**

Table 5: Performance comparison of the mixup strategy on BERT or BiLSTM layer.

reflected in the syntactic and semantic validity of the augmented examples. Similarly, for the mixup, it seems that the valid example pairs are more likely to generate positive augmented examples.

4 Related Work

In recent years, research on NER has concentrated on either enriching input text representations (Zhang and Yang, 2018; Nie et al., 2020b; Ma et al., 2020) or refining model architectures with various external knowledge (Zhang and Yang, 2018; Ye and Ling, 2018; Li et al., 2020a; Xuan et al., 2020; Li et al., 2020b; Yu et al., 2020; Shen et al., 2021). Particularly, NER model, with the aid of large pre-trained language models (Peters et al., 2018; Devlin et al., 2019; Liu et al., 2019), has achieved impressive performance gains. However, these models mostly depend on rich manual annotations, making it hard to cope with the low-resource challenges in real-world applications. Instead of pursuing a sophisticated model architecture, in this work, we exploit the BiLSTM-CRF model coupled with the pre-trained BERT as our basic model structure.

Self-augmentation methods have been widely investigated in various NLP tasks (Zhang et al., 2018; Wei and Zou, 2019; Dai and Adel, 2020; Zeng et al., 2020; Ding et al., 2020). The mainstream methods can be broadly categorized into three types: (1) token substitution (Kobayashi, 2018; Wei and Zou, 2019; Dai and Adel, 2020; Zeng et al., 2020), which performs local substitution for a given sentence, (2) paraphrasing (Kumar

Augmentation	Examples
Original	[Diana]$_{\text{PER}}$ met [Will Carling]$_{\text{PER}}$ at an exclusive gymnasium in [London]$_{\text{LOC}}$.
Positive TS	[Freddy Pinas]$_{\text{PER}}$ invited [John Marzano]$_{\text{PER}}$ at an available gym in [UK]$_{\text{LOC}}$.
Negative TS	[Tim Henman]$_{\text{PER}}$ visit [Simpson]$_{\text{PER}}$ at an available room in [NICE]$_{\text{LOC}}$.
Positive Mixup	French 1997 budget due around September 10 - Juppe.
	Jewish 1999 deficit due about October 20 M.Atherton.
Negative Mixup	Olympic champion Agassi meets Karim Alami of Morocco in the first round.
	Olympic champion Nathalie Lancien of France also missed the winning attack.

Table 6: Case study on positive and negative augmentation with respect to the TS and mixup.

et al., 2019; Xie et al., 2020; Zhang et al., 2020b), which involves sentence-level rewriting without significantly changing the semantics, and (3) mixup (Zhang et al., 2018; Chen et al., 2020; Sun et al., 2020), which carries out the feature-level augmentation. As a data-agnostic augmentation technique, mixup can help improve the generalization and robustness of our neural model acting as an useful regularizer (Verma et al., 2019). For NER, token substitution and mixup are very suitable and have been exploited successfully with specialized efforts (Dai and Adel, 2020; Chen et al., 2020; Zeng et al., 2020), while the paraphrasing strategy may result in structure incompleteness and token-label inconsistency, thus there has not been widely concerned yet. In this work, we mainly investigate the token substitution and mixup techniques for NER, as well as their integration. Despite the success of various self-augmentation methods, quality control may be an issue easily overlooked by most methods.

Many previous studies have explored the example weighting mechanism in domain adaption (Jiang and Zhai, 2007; Wang et al., 2017; Osumi et al., 2019). Xia et al. (2018) and Wang et al. (2019) looked into the example weighting methods for cross-domain tasks. Ren et al. (2018) adapted the MAML algorithm (Finn et al., 2017) and proposed a meta-learning algorithm to automatically weight training examples of the noisy label using a small unbiased validation set. Inspired by their work, we extend the meta example reweighting mechanism to the NER task, which is exploited to adaptively reweight mini-batch augmented examples during training. The main purpose is to mitigate the potential noise effects brought by the self-augmentation techniques, advancing a noise-robust model, especially in low-resource scenarios.

5 Conclusion

In this paper, we re-examine two heterogeneous self-augmentation methods (i.e., TS and mixup) for NER, extending them into more unrestricted augmentations without heuristic constraints. We further exploit a meta reweighting strategy to alleviate the potential negative impact of noisy augmented examples introduced by the aforementioned relaxation. Experiments conducted on several benchmarks show that our self-augmentation methods along with the meta reweighting mechanism are very effective in low-resource settings, and still work when enough training data is used. The combination of the two methods can lead to consistent performance improvement across all datasets. Since our framework is general and does not rely on a specific model backbone, we will further investigate other feasible model structures.

Acknowledgements

We thank the valuable comments of all anonymous reviewers. This work is supported by grants from the National Natural Science Foundation of China (No. 62176180).

References

Darina Benikova, Chris Biemann, Max Kisselew, and Sebastian Padó. 2014. Germeval 2014 named entity recognition shared task: Companion paper. *the KONVENS GermEval Shared Task on Named Entity Recognition.*

Jiaao Chen, Zhenghui Wang, Ran Tian, Zichao Yang, and Diyi Yang. 2020. Local additivity based data augmentation for semi-supervised NER. In *Proceedings of EMNLP, November 16-20, 2020*, pages 1241–1251.

Jason P. C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *TACL*, pages 357–370.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc V. Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of EMNLP, October 31 - November 4, 2018*, pages 1914–1925.

Xiang Dai and Heike Adel. 2020. An analysis of simple data augmentation for named entity recognition. In *Proceedings of COLING, December 8-13, 2020*, pages 3861–3867.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT, June 2-7, 2019*, pages 4171–4186.

Bosheng Ding, Linlin Liu, Lidong Bing, Canasai Kruengkrai, Thien Hai Nguyen, Shafiq Joty, Luo Si, and Chunyan Miao. 2020. DAGA: Data augmentation with a generation approach for low-resource tagging tasks. In *Proceedings of EMNLP*, pages 6045–6057.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of ICML, 6-11 August 2017*, volume 70, pages 1126–1135.

Joseph Fisher and Andreas Vlachos. 2019. Merge and label: A novel neural network architecture for nested NER. In *Proceedings of ACL, July 28- August 2, 2019*, pages 5840–5850.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019. Augmenting data with mixup for sentence classification: An empirical study. *CoRR*, abs/1905.08941.

James Hammerton. 2003. Named entity recognition with long short-term memory. In *Proceedings of HLT-NAACL, May 31 - June 1, 2003*, pages 172–175.

Dou Hu and Lingwei Wei. 2020. SLK-NER: exploiting second-order lexicon knowledge for chinese NER. In *SEKE, July 9-19, 2020*, pages 413–417.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in NLP. In *Proceedings of ACL*, pages 264–271.

Akbar Karimi, Leonardo Rossi, and Andrea Prati. 2021. AEDA: an easier data augmentation technique for text classification. In *Findings of EMNLP, 16-20 November, 2021*, pages 2748–2754.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of NAACL-HLT, June 1-6, 2018*, pages 452–457.

Ashutosh Kumar, Satwik Bhattamishra, Manik Bhandari, and Partha P. Talukdar. 2019. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. In *Proceedings of NAACL-HLT, June 2-7, 2019*, pages 3609–3619.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT, June 12-17, 2016*, pages 260–270.

Xiaonan Li, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2020a. FLAT: chinese NER using flat-lattice transformer. In *Proceedings of ACL, July 5-10, 2020*, pages 6836–6842.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020b. A unified MRC framework for named entity recognition. In *Proceedings of ACL, July 5-10, 2020*, pages 5849–5859.

Wei Liu, Xiyan Fu, Yue Zhang, and Wenming Xiao. 2021. Lexicon enhanced chinese sequence labeling using BERT adapter. In *Proceedings of ACL/IJCNLP, August 1-6, 2021*, pages 5847–5858.

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. *CoRR*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proceedings of ICLR 2019, May 6-9, 2019*.

Ruotian Ma, Minlong Peng, Qi Zhang, Zhongyu Wei, and Xuanjing Huang. 2020. Simplify the usage of lexicon in chinese NER. In *Proceedings of ACL, July 5-10, 2020*, pages 5951–5960.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of ACL, August 7-12, 2016*, pages 1064–1074.

Yuxian Meng, Wei Wu, Fei Wang, Xiaoya Li, Ping Nie, Fan Yin, Muyu Li, Qinghong Han, Xiaofei Sun, and Jiwei Li. 2019. Glyce: Glyph-vectors for chinese character representations. In *Proceedings of NeurIPS, December 8-14, 2019*, pages 2742–2753.

Xue Mengge, Bowen Yu, Tingwen Liu, Yue Zhang, Erli Meng, and Bin Wang. 2020. Porous lattice transformer encoder for Chinese NER. In *Proceedings of COLING*, pages 3831–3841.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NeurIPS*, pages 3111–3119.

Yuyang Nie, Yuanhe Tian, Yan Song, Xiang Ao, and Xiang Wan. 2020a. Improving named entity recognition with attentive ensemble of syntactic information. In *Findings of EMNLP 2020, 16-20 November 2020*, pages 4231–4245.

Yuyang Nie, Yuanhe Tian, Xiang Wan, Yan Song, and Bo Dai. 2020b. Named entity recognition for social media texts with semantic augmentation. In *Proceedings of EMNLP, November 16-20, 2020*, pages 1383–1391.

Kosuke Osumi, Takayoshi Yamashita, and Hironobu Fujiyoshi. 2019. Domain adaptation using a gradient reversal layer with instance weighting. In *16th International Conference on Machine Vision Applications, May 27-31, 2019*, pages 1–5.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *Proceedings of EMNLP, September 17-21, 2015*, pages 548–554.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP, October 25-29, 2014*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT, June 1-6, 2018*, pages 2227–2237.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *Proceedings of CoNLL, August 8-9, 2013*, pages 143–152.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Proceedings of EMNLP-CoNLL, July 13, 2012*, pages 1–40.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to reweight examples for robust deep learning. In *Proceedings of ICML, July 10-15, 2018*, volume 80, pages 4331–4340.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL, May 31 - June 1, 2003*, pages 142–147.

Yongliang Shen, Xinyin Ma, Zeqi Tan, Shuai Zhang, Wen Wang, and Weiming Lu. 2021. Locate and label: A two-stage identifier for nested named entity recognition. In *Proceedings of ACL/IJCNLP, August 1-6, 2021*, pages 2782–2794.

Lichao Sun, Congying Xia, Wenpeng Yin, Tingting Liang, Philip S. Yu, and Lifang He. 2020. Mixup-transformer: Dynamic data augmentation for NLP tasks. In *Proceedings of COLING, December 8-13, 2020*, pages 3436–3440.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states. In *Proceedings of ICML, 9-15 June 2019*, pages 6438–6447.

Andrew J. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theory*, 13(2):260–269.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. 2017. Instance weighting for neural machine translation domain adaptation. In *Proceedings of EMNLP, September 9-11, 2017*, pages 1482–1488.

Zhi Wang, Wei Bi, Yan Wang, and Xiaojiang Liu. 2019. Better fine-tuning via instance weighting for text classification. In *Proceedings of AAAI, January 27 - February 1, 2019*, pages 7241–7248.

Jason W. Wei and Kai Zou. 2019. EDA: easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of EMNLP-IJCNLP, November 3-7, 2019*, pages 6381–6387.

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

Fangzhao Wu, Junxin Liu, Chuhan Wu, Yongfeng Huang, and Xing Xie. 2019. Neural chinese named entity recognition via CNN-LSTM-CRF and joint training with word segmentation. In *Proceedings of WWW, May 13-17, 2019*, pages 3342–3348.

Rui Xia, Zhenchun Pan, and Feng Xu. 2018. Instance weighting for domain adaptation via trading off sample selection bias and variance. In *Proceedings of IJCAI*, page 4489–4495.

Qizhe Xie, Zihang Dai, Eduard H. Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. In *Proceedings of NeurIPS, December 6-12, 2020*.

Lu Xu, Zhanming Jie, Wei Lu, and Lidong Bing. 2021. Better feature integration for named entity recognition. In *Proceedings of NAACL-HLT, June 6-11, 2021*, pages 3457–3469.

Zhenyu Xuan, Rui Bao, and Shengyi Jiang. 2020. FGN: fusion glyph network for chinese named entity recognition. In *CCKS, November 12-15, 2020*, volume 1356, pages 28–40.

Ikuya Yamada, Akari Asai, Jin Sakuma, Hiroyuki Shindo, Hideaki Takeda, Yoshiyasu Takefuji, and Yuji Matsumoto. 2020. Wikipedia2vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from wikipedia. In *Proceedings of*

EMNLP: System Demonstrations, November 16-20, 2020, pages 23–30.

Zhi-Xiu Ye and Zhen-Hua Ling. 2018. Hybrid semi-markov CRF for neural sequence labeling. In *Proceedings of ACL, July 15-20, 2018*, pages 235–240.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of ACL, July 5-10, 2020*, pages 6470–6476.

Xiangji Zeng, Yunliang Li, Yuchen Zhai, and Yin Zhang. 2020. Counterfactual generator: A weakly-supervised method for named entity recognition. In *Proceedings of EMNLP, November 16-20, 2020*, pages 7270–7280.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *Proceedings of ICLR, April 30 - May 3, 2018*.

Rongzhi Zhang, Yue Yu, and Chao Zhang. 2020a. Seqmix: Augmenting active sequence labeling via sequence mixup. In *Proceedings of EMNLP, November 16-20, 2020*, pages 8566–8579.

Yi Zhang, Tao Ge, and Xu Sun. 2020b. Parallel data augmentation for formality style transfer. In *Proceedings of ACL*, pages 3221–3228.

Yue Zhang and Jie Yang. 2018. Chinese NER using lattice LSTM. In *Proceedings of ACL, July 15-20, 2018*, pages 1554–1564.