arXiv:2105.06804v2 [cs.CL] 13 Jul 2021

Locate and Label: A Two-stage Identifier for Nested Named Entity Recognition

Yongliang Shen[1], **Xinyin Ma**[1], **Zeqi Tan**[1], **Shuai Zhang**[1], **Wen Wang**[2], **Weiming Lu**[1]*
[1]College of Computer Science and Technology, Zhejiang University
[2]University of Science and Technology of China
{syl, luwm}@zju.edu.cn

Abstract

Named entity recognition (NER) is a well-studied task in natural language processing. Traditional NER research only deals with flat entities and ignores nested entities. The span-based methods treat entity recognition as a span classification task. Although these methods have the innate ability to handle nested NER, they suffer from high computational cost, ignorance of boundary information, under-utilization of the spans that partially match with entities, and difficulties in long entity recognition. To tackle these issues, we propose a two-stage entity identifier. First we generate span proposals by filtering and boundary regression on the seed spans to locate the entities, and then label the boundary-adjusted span proposals with the corresponding categories. Our method effectively utilizes the boundary information of entities and partially matched spans during training. Through boundary regression, entities of any length can be covered theoretically, which improves the ability to recognize long entities. In addition, many low-quality seed spans are filtered out in the first stage, which reduces the time complexity of inference. Experiments on nested NER datasets demonstrate that our proposed method outperforms previous state-of-the-art models.

1 Introduction

Named entity recognition (NER) is a fundamental task in natural language processing, focusing on identifying the spans of text that refer to entities. NER is widely used in downstream tasks, such as entity linking (Ganea and Hofmann, 2017; Le and Titov, 2018) and relation extraction (Li and Ji, 2014; Miwa and Bansal, 2016).

Previous works usually treat NER as a sequence labeling task, assigning a single tag to each token in a sentence. Such models lack the ability to identify nested named entities. Various approaches for nested NER have been proposed in recent years. Some works revised sequence models to support nested entities using different strategies (Alex et al., 2007; Ju et al., 2018; Straková et al., 2019; Wang et al., 2020a) and some works adopt the hyper-graph to capture all possible entity mentions in a sentence (Lu and Roth, 2015; Katiyar and Cardie, 2018). We focus on the span-based methods (Sohrab and Miwa, 2018; Zheng et al., 2019; Tan et al., 2020), which treat named entity recognition as a classification task on a span with the innate ability to recognize nested named entities. For example, Sohrab and Miwa (2018) exhausts all possible spans in a text sequence and then predicts their categories. However, these methods suffer from some serious weaknesses. First, due to numerous low-quality candidate spans, these methods require high computational costs. Then, it is hard to identify long entities because the length of the span enumerated during training is not infinite. Next, boundary information is not fully utilized, while it is important for the model to locate entities.

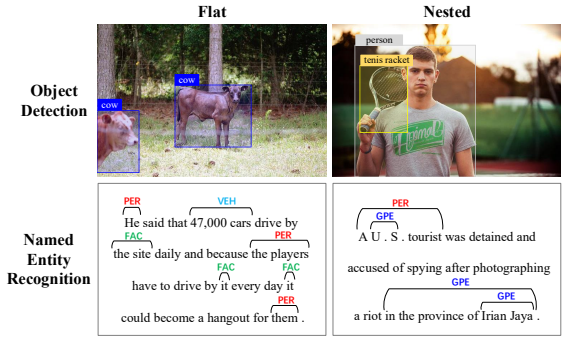

Figure 1: A Comparison of Named Entity Recognition and Object Detection. Examples of flat and nesetd entities or objects sampled from the COCO 2017 dataset and the ACE04 dataset, respectively.

* Corresponding author

Although some methods (Zheng et al., 2019; Tan et al., 2020) have used a sequence labeling model to predict boundaries, yet without dynamic adjustment, the boundary information is not fully utilized. Finally, the spans which partially match with entities are not effectively utilized. These methods simply treat the partially matched spans as negative examples, which can introduce noise into the model.

Different from the above studies, we observed that NER and object detection tasks in computer vision have a high degree of consistency. They both need to locate regions of interest (ROIs) in the context (image/text) and then assign corresponding categories to them. Furthermore, both flat NER and nested NER have corresponding structures in the object detection task, as shown in Figure 1. For the flat structure, there is no overlap between entities or between objects. While for nested structures, fine-grained entities are nested inside coarse-grained entities, and small objects are nested inside large objects correspondingly. In computer vision, the two-stage object detectors (Girshick et al., 2014; Girshick, 2015; Ren et al., 2017; Dai et al., 2016; He et al., 2017; Cai and Vasconcelos, 2018) are the most popular object detection algorithm. They divide the detection task into two stages, first generating candidate regions, and then classifying and fine-tuning the positions of the candidate regions.

Inspired by these, we propose a two-stage entity identifier and treat NER as a joint task of boundary regression and span classification to address the weaknesses mentioned above. In the first stage, we design a span proposal module, which contains two components: a filter and a regressor. The filter divides the seed spans into contextual spans and span proposals, and filters out the former to reduce the candidate spans. The regressor locates entities by adjusting the boundaries of span proposals to improve the quality of candidate spans. Then in the second stage, we use an entity classifier to label entity categories for the number-reduced and quality-improved span proposals. During training, to better utilize the spans that partially match with the entities, we construct soft examples by weighting the loss of the model based on the IoU. In addition, we apply the soft non-maximum suppression (Soft-NMS) (Bodla et al., 2017) algorithm to entity decoding for dropping the false positives.

Our main contributions are as follow:

- Inspired by the two-stage detector popular in object detection, we propose a novel two-stage identifier for NER of locating entities first and labeling them later. We treat NER as a joint task of boundary regression and span classification.
- We make effective use of boundary information. Taking the identification of entity boundaries a step further, our model can adjust the boundaries to accurately locate entities. And when training the boundary regressor, in addition to the boundary-level SmoothL1 loss, we also use a span-level loss, which measures the overlap between two spans.
- During training, instead of simply treating the partially matched spans as negative examples, we construct soft examples based on the IoU. This not only alleviates the imbalance between positive and negative examples, but also effectively utilizes the spans which partially match with the ground-truth entities.
- Experiments show that our model achieves state-of-the-art performance consistently on the KBP17, ACE04 and ACE05 datasets, and outperforms several competing baseline models on F1-score by +3.08% on KBP17, +0.71% on ACE04 and +1.27% on ACE05.

2 Model

Figure 2 illustrates an overview of the model structure. We first obtain the word representation through the encoder and generate seed spans. Among these seed spans, some with higher overlap with the entities are the proposal spans, and others with lower overlap are the contextual spans. In the span proposal module, we use a filter to keep the proposal spans and drop the contextual spans. Meanwhile, a regressor regresses the boundary of each span to locate the left and right boundaries of entities. Next, we adjust the boundaries of the span proposals based on the output of the regressor, and then feed them into the entity classifier module. Finally, the entity decoder decodes the entities using the Soft-NMS algorithm. We will cover our model in the following sections.

2.1 Token Representation

Consider the i-th word in a sentence with n words, we represent it by concatenating its word embedding x_i^w, contextualized word embedding x_i^{lm}, part-

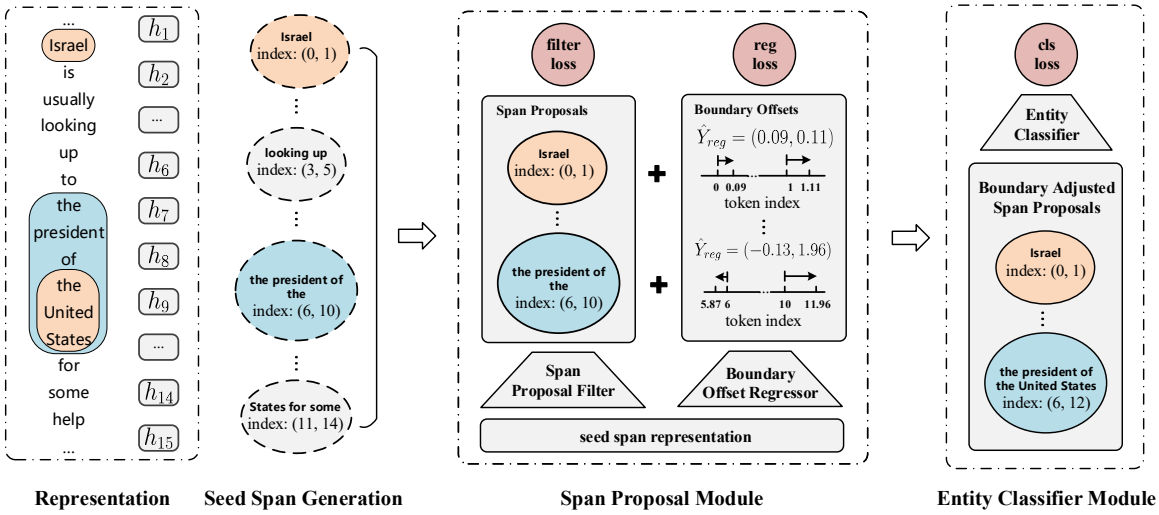

Figure 2: The overall architecture of the Two-stage Identifier.

of-speech(POS) embedding x_i^{pos} and character-level embedding x_i^{char} together. The character-level embedding is generated by a BiLSTM module with the same setting as (Ju et al., 2018). For the contextualized word embedding, we follow (Yu et al., 2020) to obtain the context-dependent embedding for a target token with one surrounding sentence on each side. Then, the concatenation of them is fed into another BiLSTM to obtain the hidden state as the final word representation $h_i \in \mathbb{R}^d$.

2.2 Seed Span Generation

Seed spans are subsequences sampled from a sequence of words. By filtering, adjusting boundaries, and classifying on them, we can extract entities from the sentence. Under the constraint of a pre-specified set of lengths, where the maximum does not exceed L, we enumerate all possible start and end positions to generate the seed spans. We denote the set of seed spans as $\mathcal{B} = \{b_0, \ldots, b_K\}$, where $b_i = (st_i, ed_i)$ denotes i-th seed span, K denotes the number of the generated seed spans, and st_i, ed_i denote the start and end positions of the span respectively.

For training the filter and the regressor, we need to assign a corresponding category and regression target to each seed span. Specifically, we pair each seed span in $\mathcal{B}$ and the ground-truth entity with which the span has the largest IoU. The IoU measure the overlap between spans, defined as $\text{IoU}(A, B) = \frac{A \cap B}{A \cup B}$, where A and B are two spans. Then we divide them into positive and negative spans based on the IoU between the pair. The spans whose IoU with the paired ground truth is above the threshold α_1 are classified as positive examples, and those less than threshold α_1 are classified as negative examples. For the positive span, we assign it the same category $\hat{y}$ with the paired ground truth and compute the boundary offset $\hat{t}$ between them. For the negative span, we only assign a NONE label. We downsample the negative examples such that the ratio of positive to negative is 1:5.

2.3 Span Proposal Module

The quality of the generated seed spans is variable. If we directly input them into the entity classifier, it will lead to a lot of computational waste. High-quality spans have higher overlap with entities, while low-quality spans have lower overlap. We denote them as **span proposals** and **contextual spans**, respectively. Our Span Proposal module consists of two components: Span Proposal Filter and Boundary Regressor. The former is used to drop the contextual spans and keep the span proposals, while the latter is used to adjust the boundaries of the span proposals to locate entities.

Span Proposal Filter For the seed span $b_i(st_i, ed_i)$, we concatenate the maximum pooled span representation h_i^p with the inner boundary word representations (h_{st_i}, h_{ed_i}) to obtain the span representation h_i^{filter}. Based on it we calculate the probability p_i^{filter} that the span b_i belongs to the span proposals, computed as follows:

$$h_i^p = \text{MaxPooling}(h_{st_i}, h_{st_i+1}, \ldots, h_{ed_i}) \quad (1)$$

$$h_i^{filter} = [h_i^p; h_{st_i}; h_{ed_i}] \tag{2}$$

$$p_i^{filter} = \text{Sigmoid}\left(\text{MLP}\left(h_i^{filter}\right)\right) \tag{3}$$

where $[;]$ denotes the concatenate operation, MLP consists of two linear layers and a GELU (Hendrycks and Gimpel, 2016) activation function.

Boundary Regressor Although the span proposal has a high overlap with the entity, it cannot hit the entity exactly. We design another boundary regression branch where a regressor locates entities by adjusting the left and right boundaries of the span proposals. The boundaries regression requires not only the information of span itself but also the outer boundary words. Thus we concatenate the maximum pooled span representation h_i^p with the outer boundary word representations (h_{st_i-1}, h_{ed_i+1}) to obtain the span representation h_i^{reg}. Then we calculate the offsets t_i of left and right boundaries:

$$h_i^{reg} = [h_i^p; h_{st_i-1}; h_{ed_i+1}] \tag{4}$$

$$t_i = W_2 \cdot \text{GELU}(W_1 h_i^{reg} + b_1) + b_2 \tag{5}$$

where $W_1 \in \mathbb{R}^{3d\times d}$, $W_2 \in \mathbb{R}^{d\times 2}$, $b_1 \in \mathbb{R}^d$ and $b_2 \in \mathbb{R}^2$ are learnable parameters.

2.4 Entity Classifier Module

With the boundary offsets t_i predicted by the boundary regressor, we adjust the boundaries of span proposals. The adjusted start postion $\widetilde{st}_i$ and end position $\widetilde{ed}_i$ of b_i are calculated as follow:

$$\widetilde{st}_i = \max(0, st_i + \left\lfloor t_i^l + \frac{1}{2} \right\rfloor) \tag{6}$$

$$\widetilde{ed}_i = \min(L-1, ed_i + \left\lfloor t_i^r + \frac{1}{2} \right\rfloor) \tag{7}$$

where t_i^l and t_i^r denote the left and right offsets, respectively. As in the filter above, we concatenate the maximum pooled span representation $\widetilde{h}_i^p$ with the inner boundary word representations $(h_{\widetilde{st}_i}, h_{\widetilde{ed}_i})$. Then we perform entity classification:

$$\widetilde{h}_i^p = \text{MaxPooling}(h_{\widetilde{st}_i}, h_{\widetilde{st}_i+1}, \ldots, h_{\widetilde{ed}_i}) \tag{8}$$

$$h_i^{cls} = [\widetilde{h}_i^p; h_{\widetilde{st}_i}; h_{\widetilde{ed}_i}] \tag{9}$$

$$p_i = \text{Softmax}\left(\text{MLP}\left(h_i^{cls})\right)\right) \tag{10}$$

where MLP consists of two linear layers and a GELU activation function, as in the filter above.

For training the entity classifier, we need to reassign the categories based on the IoU between the new adjusted span proposal and paired ground-truth entity. Specifically, if the IoU between a span and its corresponding entity is higher than the threshold α_2, we assign the span the same category with the entity, otherwise we assign it a NONE category and treat the span as a negative example.

2.5 Training Objective

The spans that partially match with the entities are very important, but previous span-based approaches simply treat them as negative examples. Such practice not only fails to take advantage of these spans but also introduces noise into the model. We treat partially matched spans as soft examples by weighting its loss based on its IoU with the corresponding ground truth. For the i-th span b_i, the weight w_i is calculated as follows:

$$\begin{cases} \text{IoU}(b_i, e_i)^\eta, & \text{IoU}(b_i, e_i) \geq \alpha \\ (1 - \text{IoU}(b_i, e_i))^\eta, & \text{IoU}(b_i, e_i) < \alpha \end{cases} \tag{11}$$

where $\alpha \in \{\alpha_1, \alpha_2\}$ denotes the IoU threshold used in the first or the second stage and e_i denotes corresponding ground-truth entity of b_i. η is a focusing parameter that can smoothly adjust the rate at which partially matched examples are down-weighted. We can find that if we set $\eta = 0$, the above formula degenerates to a hard one. Also, if a span does not overlap with any entity or match exactly with some entity, the loss weight $w_i = 1$.

Then, we calculate the losses for the span proposal filter, boundary regressor and entity classifier respectively. For the span proposal filter, we use focal loss (Lin et al., 2017) to solve the imbalance problem:

$$\begin{aligned} \mathcal{L}_{filter} = -\sum_i & w_i \mathbb{I}_{\hat{y}\neq 0}(1 - p_i^{filter})^\gamma \log(p_i^{filter}) \\ & + w_i \mathbb{I}_{\hat{y}=0}(p_i^{filter})^\gamma \log(1 - p_i^{filter}) \end{aligned} \tag{12}$$

where w_i is the weight of i-th example calculated at Equation 11 and γ denotes focusing parameter of focal loss. For the boundary regressor, the loss consists of two components, the smooth L1 loss at

the boundary level and the overlap loss at the span level, calculated as follows:

$$\mathcal{L}_{reg}\left(\hat{t}, t\right) = \mathcal{L}_{f1} + \mathcal{L}_{olp} \tag{13}$$

$$\mathcal{L}_{f1}\left(\hat{t}, t\right) = \sum_{i} \sum_{j \in \{l,r\}} \text{smoothL1}\left(\hat{t}_i^j, t_i^j\right) \tag{14}$$

$$\mathcal{L}_{olp} = \sum_{i} \left(1 - \frac{\min\left(d_i\right) - \max\left(e_i\right)}{\max\left(d_i\right) - \min\left(e_i\right)}\right) \tag{15}$$

where $d_i = \left\{\widetilde{ed}_i, \hat{ed}_i\right\}$, $e_i = \left\{\widetilde{st}_i, \hat{st}_i\right\}$. $\hat{st}_i$, $\hat{ed}_i$, $\hat{t}_i^l$ and $\hat{t}_i^r$ denote the ground-truth left boundary, right boundary, left offset and right offset, respectively. For the entity classifier, we simply use the cross-entropy loss:

$$\mathcal{L}_{cls} = \sum_{i} w_i \, \text{CELoss}(\hat{y}, p_i) \tag{16}$$

where w_i is the weight of i-th example calculated at Equation 11. We train the filter, regressor and classifier jointly, thus the total loss is computed as:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{filter} + \lambda_2 \mathcal{L}_{reg} + \lambda_3 \mathcal{L}_{cls} \tag{17}$$

where λ_1, λ_2 and λ_3 are the weights of filter, regressor and classifier losses respectively.

2.6 Entity Decoding

In the model prediction phase, after the above steps, we get the classification probability and boundary offset regression results for each span proposal. Based on them, we need to extract all entities in the sentence (i.e., find the exact start and end positions of the entities as well as their corresponding categories). We assign label $y_i = \text{argmax}(p_i)$ to span s_i and use $score_i = \max(p_i)$ as the confidence of span s_i belonging to the y_i category.

Now for each span proposal, our model has predicted the exact start and end positions, the entity class and the corresponding score, denoted as $s_i = (l_i, r_i, y_i, score_i)$. Given the score threshold δ and the set of span proposals $\mathcal{S} = \{s_1, \ldots, s_N\}$, where N denotes as the number of span proposals, we use the Soft-NMS (Bodla et al., 2017) algorithm to filter the false positives. As shown in Algorithm 1, we traverse the span proposals by the order of their score (the traversal term is denoted as s_i) and then adjust the scores of other span proposals s_j to $f(s_i, s_j)$, which is defined as:

$$\begin{cases} score_j * u, & \text{IoU}(s_i, s_j) \geq k \\ score_j, & \text{IoU}(s_i, s_j) < k \end{cases} \tag{18}$$

where $u \in (0, 1)$ denotes the decay coefficient of the score and k denotes is the IoU threshold. Then we keep all span proposals with a $score > \delta$ as the final extracted entities.

Algorithm 1: Soft-NMS Algorithm

Input: $\mathcal{S} = \{s_i, \ldots, s_N\}$, δ, where $s_i = (l_i, r_i, y_i, score_i)$
Output: $\mathcal{O}$

$\mathcal{O} \leftarrow \{\}$;
Sort($\mathcal{S}$) by the $score$ of each element in descend order;
for s_i *in* $\mathcal{S}$ **do**
  $\mathcal{O} \leftarrow \mathcal{O} \cup \{s_i\}$;
  for s_j *in* $\mathcal{S}\,[i : N]$ **do**
    $\mathcal{S} \leftarrow \mathcal{S} - \{s_j\}$;
    $s_j \leftarrow (l_j, r_j, y_j, f(s_i, s_j))$;
    Insert $(\mathcal{S}, k, s_j)$ where k denotes the insertion position of s_j in $\mathcal{S}$ ordered by $score$;
  end
end

3 Experiment Settings

3.1 Datasets

To provide empirical evidence for effectiveness of the proposed model, we conduct our experiments on four nested NER datasets: ACE04 [1], ACE05 [2], KBP17[3] and GENIA [4]. Please refer to Appendix A.1 for statistical information about the datasets.

ACE 2004 and ACE 2005 (Doddington et al., 2004; Christopher Walker and Maeda, 2006) are two nested datasets, each of them contains 7 entity categories. We follow the same setup as previous work Katiyar and Cardie (2018); Lin et al. (2019) split them into train, dev and test sets by 8:1:1.

[1] https://catalog.ldc.upenn.edu/LDC2005T09
[2] https://catalog.ldc.upenn.edu/LDC2006T06
[3] https://catalog.ldc.upenn.edu/LDC2019T02
[4] http://www.geniaproject.org/genia-corpus

KBP17 (Ji et al., 2017) has 5 entity categories, including GPE, ORG, PER, LOC, and FAC. We follow Lin et al. (2019) to split all documents into 866/20/167 documents for train/dev/test set.

GENIA (Ohta et al., 2002) is a biology nested named entity dataset and contains five entity types, including DNA, RNA, protein, cell line, and cell type categories. Following Yu et al. (2020), we use 90%/10% train/test split.

3.2 Evaluation Metrics

We use strict evaluation metrics that an entity is confirmed correct when the entity boundary and the entity label are correct simultaneously. We employ precision, recall and F1-score to evaluate the performance.

3.3 Parameter Settings

In most experiments, we use GloVE (Pennington et al., 2014) and BERT (Devlin et al., 2019) in our encoder. For the GENIA dataset, we replace GloVE with BioWordvec (Chiu et al., 2016), BERT with BioBERT (Lee et al., 2019). The dimensions for x_i^w, x_i^{lm}, x_i^{pos}, x_i^{char} and h_i are 100, 1024, 50, 50 and 1024, respectively. For all datasets, we train our model for 35 epochs and use the Adam Optimizer with a linear warmup-decay learning rate schedule, a dropout before the filter, regressor and entity classifier with a rate of 0.5. See Appendix A for more detailed parameter settings and baseline models we compared [5].

4 Results and Comparisons

4.1 Overall Evaluation

Table 1 illustrates the performance of the proposed model as well as baselines on ACE04, ACE05, GENIA and KBP17. Our model outperforms the state-of-the-art models consistently on three nested NER datasets. Specifically, the F1-scores of our model advance previous models by +3.08%, +0.71%, +1.27% on KBP17, ACE04 and ACE05 respectively. And on GENIA, we achieve comparable performance. We analyze the performance on entities of different lengths on ACE04, as shown in Table 2. We observe that the model works well on the entities whose lengths are not enumerated during training. For example, although entities of length 6 are not enumerated, while those of length 5 and 7 are enumerated, our model can achieve a comparable F1-score for entities of length 6. In particular, the entities whose lengths exceed the maximum length (15) enumerated during training, are still well recognized. This verifies that our model has the ability to identify length-uncovered entities and long entities by boundary regression. We also evaluated our model on two flat NER datasets, as shown in Appendix B.

Model	ACE04		
	Pr.	Rec.	F1
Katiyar and Cardie (2018)	73.60	71.80	72.70
Shibuya and Hovy (2020)	83.73	81.91	82.81
Straková et al. (2019)	-	-	84.40
Wang et al. (2020a)	86.08	86.48	86.28
Yu et al. (2020)	87.30	86.00	86.70
Ours	**87.44**	**87.38**	**87.41**

Model	ACE05		
	Pr.	Rec.	F1
Katiyar and Cardie (2018)	70.60	70.40	70.50
Lin et al. (2019)	76.20	73.60	74.90
Luo and Zhao (2020)	75.00	75.20	75.10
Straková et al. (2019)	-	-	84.33
Wang et al. (2020a)	83.95	85.39	84.66
Yu et al. (2020)	85.20	85.60	85.40
Ours	**86.09**	**87.27**	**86.67**

Model	KBP17		
	Pr.	Rec.	F1
Ji et al. (2017)	76.20	73.00	72.80
Lin et al. (2019)	77.70	71.80	74.60
Luo and Zhao (2020)	77.10	74.30	75.60
Li et al. (2020b)	80.97	81.12	80.97
Ours	**85.46**	**82.67**	**84.05**

Model	GENIA		
	Pr.	Rec.	F1
Lin et al. (2019)	75.80	73.90	74.80
Luo and Zhao (2020)	77.40	74.60	76.00
Wang et al. (2020b)	78.10	74.40	76.20
Straková et al. (2019)	-	-	78.31
Wang et al. (2020a)	79.45	78.94	79.19
Yu et al. (2020)	81.80	79.30	80.50
Ours	80.19	**80.89**	**80.54**

Table 1: Results for ***nested*** NER tasks

4.2 Ablation Study

We choose the ACE04 and KBP17 datasets to conduct several ablation experiments to elucidate the main components of our proposed model. To illustrate the performance of the model on entities of different lengths, we divide the entities into three groups according to their lengths. The re-

[5] Our code is available at https://github.com/tricktreat/locate-and-label.

Length	ACE04			
	Pr.	Rec.	F1	Support
1	89.62	90.98	90.30	1519
2	87.93	86.10	87.01	626
3	89.67	84.59	87.06	318
4	79.04	88.59	83.54	149
5	85.58	83.18	84.36	107
6	**84.62**	**86.84**	**85.71**	**76**
7	85.07	85.07	85.07	67
8	**79.31**	**79.31**	**79.31**	**29**
9	81.48	73.33	77.19	30
10	**76.47**	**76.47**	**76.47**	**17**
11	68.75	68.75	68.75	16
12	**66.67**	**80.00**	**72.73**	**15**
13	100.00	85.71	92.31	7
14	**55.56**	**83.33**	**66.67**	**6**
15	55.56	71.43	62.50	7
16	**80.00**	**57.14**	**66.67**	**7**
17	**66.67**	**80.00**	**72.73**	**5**
18	**83.33**	**83.33**	**83.33**	**6**
19	**33.33**	**33.33**	**33.33**	**3**
$\geq$ 20	**66.67**	**24.00**	**35.29**	**25**
All	87.46	87.35	87.41	3035

Table 2: A comparison of recognition F1-score on entities of different lengths. Regular rows indicate that the entity lengths are enumerated, while bold ones indicate that the entity lengths are not enumerated.

sults are shown in Table 3. Firstly, we observe that the boundary regressor is very effective for the identification of long entities. Lack of the boundary regressor leads to a decrease in F1-score for long entities ($L \geq 10$) on ACE04 by 36.73% and KBP17 by 30.54%. Then, compared with the *w/o filter* setting, the F1-scores of our full model on the two datasets improved by 0.52% and 0.75%, respectively. In addition, experimental results also demonstrate that the soft examples we constructed are effective. This allows the model to take full advantage of the information of partially matched spans in training, improving the F1-score by 0.87% on ACE04 and 0.16% on KBP17. However, Soft-NMS play a limited role and improve the model performance only a little. We believe that text is sparse data compared to images and the number of false positives predicted by our model is quite small, so the Soft-NMS can hardly perform the role of a filter.

4.3 Time Complexity

Theoretically, the number of possible spans of a sentence of length N is $\frac{N(N+1)}{2}$. Previous span-based methods need to classify almost all spans into corresponding categories, which leads to the high computational cost with $O(cN^2)$ time complexity where c is the number of categories. The words in a sentence can be divided into two categories: contextual words and entity words. Traditional approaches waste a lot of computation on the spans composed of contextual words. However, our approach retains only the span proposals containing entity words by the filter, and the time complexity is $O(N^2)$. Although in the worst case the model keeps all seed spans, generating $\frac{N(N+1)}{2}$ span proposals, we observe that we generate approximately three times as many span proposals as the entities in practice. Assuming that the number of entities in the sentence is k, the total time complexity of our model is $O(N^2 + ck)$ where $k << N^2$.

5 Case Study

Examples of model predictions are shown in Table 4. The first line illustrates that our model can recognize entities with multi-level nested structures. We can see that the three nested entities from inside to outside are *united nations secretary general kofi annan*, *united nations secretary general* and *united nations*, all of which can be accurately recognized by our model. The second line illustrates that our model can recognize long entities well, although trained without seed spans of the same length as it. The long entity *Aceh, which is rich in oil and gas and has a population of about 4.1 million people*, with a length of 20, exceeds the maximum length of generated seed spans, but can still be correctly located and classified. However, our model has difficulties in resolving ambiguous entity references. As shown in the third line, our model incorrectly classifies the reference phrase *both sides*, which refers to ORG, into the PER category.

6 Related Work

6.1 Nested Named Entity Recognition

NER is usually modeled as a sequence labeling task, and a sequence model (e.g., LSTM-CRF (Huang et al., 2015)) is employed to output the sequence of labels with maximum probability. However, traditional sequence labeling models cannot handle nested structures because they can only assign one label to each token. In recent years, several approaches have been proposed to solve the nested named entity recognition task, mainly including tagging-based (Alex et al., 2007; Wang et al., 2020a), hypergraph-based (Muis and Lu, 2017; Katiyar and Cardie, 2018), and span-based

Model	F1-score on ACE04				F1-score on KBP17			
	1≤L<5	5≤L<10	L≥10	ALL	1≤L<5	5≤L<10	L≥10	ALL
support	2612	309	114	3035	11594	756	250	12600
Full model	**88.73**	83.71	**66.06**	**87.41**	**85.52**	67.67	58.58	**84.05**
w/o regressor	88.63	66.41	29.33	85.18	83.99	50.50	28.04	82.54
w/o filter	88.35	83.87	60.55	86.89	84.77	67.04	59.06	83.30
w/o filter & regressor	88.59	65.65	31.08	85.12	85.28	51.76	26.03	82.85
w/o soft-NMS	88.66	83.50	65.16	87.28	85.49	67.62	58.77	84.02
w/o soft examples	88.39	80.39	55.96	86.54	85.27	68.95	60.85	83.89

Table 3: Ablation study on ACE04 and KBP17. To compare the performance of the model on entities of different lengths, we divided the entities into three groups: $1 \leq L < 5$, $5 \leq L < 10$ and $L \geq 10$.

[3[3[2[2[1[1united nations1]ORG1]ORG secretary general2]PER2]PER kofi annan3]PER3]PER today discussed plans for the summit with [1[1the host1]PER1]PER , [3[3[2[2[1[1egyptian1]GPE1]GPE president2]PER2]PER hosni mubarak3]PER3]PER .
[1[1Separatists1]PER1]PER have fought since 1975 for independence in [3[3Aceh , [1[1which1]GEP1]GEP is rich in oil and gas and has [2[2a population of [1about 4 . 1 million people1]PER2]PER2]PER3]GEP3]GEP .
[2[2The [1US1]GPE Supreme Court2]ORG2]ORG will hear arguments from [1[1both sides1]PER1]ORG on Friday and [2[2[1[1Florida1]GPE1]GPE ' s [1[1Leon County1]GPE1]GPE Circuit Court2]ORG2]ORG will consider the arguments on disputed [1[1state1]GPE1]GPE ballots on Saturday .

Table 4: Cases Study. Blue brackets indicate entities predicted by the model, red brackets indicate true entities, the labels in the lower right corner indicate the type of entity, and the superscripts indicate the level of the nesting.

(Sohrab and Miwa, 2018; Zheng et al., 2019) approaches. The tagging based nested NER model transforms the nested NER task into a special sequential tagging task by designing a suitable tagging schema. Layered-CRF (Alex et al., 2007) dynamically stacks flat NER layers to identify entities from inner to outer. Pyramid (Wang et al., 2020a) designs a pyramid structured tagging framework that uses CNN networks to identify entities from the bottom up. The hypergraph-based model constructs the hypergraph by the structure of nested NER and decodes the nested entities on the hypergraph. Lu and Roth (2015) is the first to propose the use of Mention Hypergraphs to solve the overlapping mentions recognition problem. Katiyar and Cardie (2018) proposed hypergraph representation for the nested NER task and learned the hypergraph structure in a greedy way by LSTM networks. The span-based nested NER model first extracts the subsequences (spans) in a sequence and then classifies these spans. Exhaustive Model (Sohrab and Miwa, 2018) exhausts all possible spans in a text sequence and then predicts their classes. Zheng et al. (2019); Tan et al. (2020) took a sequence labeling model to identify entity boundaries and then predicted the categories of boundary-relevant regions. Different from the above methods, some works adopt the methods from other tasks. For example, Yu et al. (2020) reformulated NER as a structured prediction task and adopted a biaffine model for nested and flat NER. While Li et al. (2020b) treated NER as a reading comprehension task, and constructed type-specific queries to extract entities from the context.

6.2 Object Detection

Object detection is a computer vision technique that can localize and identify objects in an image. With this identification and localization, object detection can determine the exact location of objects while assigning them categories. Neural-based object detection algorithms are divided into two main categories: one-stage and two-stage approach. The one-stage object detector densely proposes anchor boxes by covering the possible positions, scales, and aspect ratios, and then predicts the categories and accurate positions based on them in a single-shot way, such as OverFeat (Sermanet et al., 2013), YOLO (Redmon et al., 2016) and SSD (Liu et al., 2016). The two-stage object detector can be seen as an extension of the dense detector and has been the most dominant object detection algorithm for many years (Girshick et al., 2014; Girshick, 2015; Ren et al., 2017; Dai et al., 2016; He et al., 2017; Cai and Vasconcelos, 2018). It first obtains sparse proposal boxes containing objects from a dense set of region candidates, and then adjusts the position and predicts a category for each proposal.

7 Conclusion

In this paper, we treat NER as a joint task of boundary regression and span classification and propose a two-stage entity identifier. First we generate span proposals through a filter and regressor, then classify them into the corresponding categories. Our proposed model can make full use of the boundary information of entities and reduce the computational cost. Moreover, by constructing soft samples during training, our model can exploit the spans that partially match with the entities. Experiments illustrate that our method achieves state-of-the-art performance on several nested NER datasets. For future work, we will combine named entity recognition and object detection tasks, and try to use a unified framework to address joint identification on multimodal data.

Acknowledgments

This work is supported by the Key Research and Development Program of Zhejiang Province, China(No. 2021C01013), the National Key Research and Development Project of China (No. 2018AAA0101900), the Chinese Knowledge Center of Engineering Science and Technology (CKCEST) and MOE Engineering Research Center of Digital Library.

References

Beatrice Alex, Barry Haddow, and Claire Grover. 2007. Recognising nested named entities in biomedical text. In *Biological, translational, and clinical language processing*, pages 65–72, Prague, Czech Republic. Association for Computational Linguistics.

N. Bodla, B. Singh, R. Chellappa, and L. S. Davis. 2017. Soft-nms — improving object detection with one line of code. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5562–5570.

Z. Cai and N. Vasconcelos. 2018. Cascade r-cnn: Delving into high quality object detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6154–6162.

Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. 2016. How to train good word embeddings for biomedical NLP. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, pages 166–174, Berlin, Germany. Association for Computational Linguistics.

Stephanie Strassel Christopher Walker and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. linguistic. In *Linguistic Data Consortium, Philadelphia 57*.

Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. 2016. R-fcn: Object detection via region-based fully convolutional networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 379–387, Red Hook, NY, USA. Curran Associates Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2619–2629, Copenhagen, Denmark. Association for Computational Linguistics.

Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, page 1440–1448, USA. IEEE Computer Society.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, page 580–587, USA. IEEE Computer Society.

K. He, G. Gkioxari, P. Dollár, and R. Girshick. 2017. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415.

Dou Hu and Lingwei Wei. 2020. Slk-ner: Exploiting second-order lexicon knowledge for chinese ner. In *The 32nd International Conference on Software & Knowledge Engineering*.

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

Heng Ji, Xiaoman Pan, Boliang Zhang, Joel Nothman, James Mayfield, Paul McNamee, and Cash Costello. 2017. Overview of TAC-KBP2017 13 languages entity discovery and linking. In *Proceedings of the 2017 Text Analysis Conference, TAC 2017, Gaithersburg, Maryland, USA, November 13-14, 2017*. NIST.

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A neural layered model for nested named entity recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1446–1459, New Orleans, Louisiana. Association for Computational Linguistics.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 861–871, New Orleans, Louisiana. Association for Computational Linguistics.

Phong Le and Ivan Titov. 2018. Improving entity linking by modeling latent relations between mentions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1595–1604, Melbourne, Australia. Association for Computational Linguistics.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412, Baltimore, Maryland. Association for Computational Linguistics.

Xiaonan Li, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2020a. FLAT: Chinese NER using flat-lattice transformer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6836–6842, Online. Association for Computational Linguistics.

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 the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859, Online. Association for Computational Linguistics.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2019. Sequence-to-nuggets: Nested entity mention detection via anchor-region networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5182–5192, Florence, Italy. Association for Computational Linguistics.

T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. 2017. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 857–867, Lisbon, Portugal. Association for Computational Linguistics.

Ying Luo and Hai Zhao. 2020. Bipartite flat-graph network for nested named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6408–6418, Online. Association for Computational Linguistics.

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 *Advances in Neural Information Processing Systems*, volume 32, pages 2746–2757. Curran Associates, Inc.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany. Association for Computational Linguistics.

Aldrian Obaja Muis and Wei Lu. 2017. Labeling gaps between words: Recognizing overlapping mentions with mention separators. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2608–2618, Copenhagen, Denmark. Association for Computational Linguistics.

Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. 2002. The genia corpus: An annotated research abstract corpus in molecular biology domain. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, page 82–86, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for Chinese social media with jointly trained embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 548–554, Lisbon, Portugal. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. 2016. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2017. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6):1137–1149.

Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. 2013. Overfeat: Integrated recognition, localization and detection using convolutional networks. *2nd International Conference on Learning Representations*.

Takashi Shibuya and Eduard Hovy. 2020. Nested named entity recognition via second-best sequence learning and decoding. *Transactions of the Association for Computational Linguistics*, 8:605–620.

Mohammad Golam Sohrab and Makoto Miwa. 2018. Deep exhaustive model for nested named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2843–2849, Brussels, Belgium. Association for Computational Linguistics.

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested NER through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331, Florence, Italy. Association for Computational Linguistics.

Chuanqi Tan, Wei Qiu, Mosha Chen, Rui Wang, and Fei Huang. 2020. Boundary enhanced neural span classification for nested named entity recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9016–9023.

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 the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Jue Wang, Lidan Shou, Ke Chen, and Gang Chen. 2020a. Pyramid: A layered model for nested named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5918–5928, Online. Association for Computational Linguistics.

Yu Wang, Yun Li, Hanghang Tong, and Ziye Zhu. 2020b. HIT: Nested named entity recognition via head-tail pair and token interaction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6027–6036, Online. Association for Computational Linguistics.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics.

Changmeng Zheng, Yi Cai, Jingyun Xu, Ho-fung Leung, and Guandong Xu. 2019. A boundary-aware neural model for nested named entity recognition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 357–366, Hong Kong, China. Association for Computational Linguistics.

A Experiments on Nested NER

A.1 Statistics of Nested Datasets

In Table 5, We report the number of sentences, the number of sentences containing nested entities, the average sentence length, the total number of entities, the number of nested entities and the nesting ratio on the ACE04, ACE05, GENIA and KBP17 datasets.

A.2 Baseline Methods

We use the following models as baselines for nested NER:

- **Biaffine** (Yu et al., 2020) reformulates NER as a structured prediction task and adopts a dependency parsing approach for NER.
- **Pyramid** (Wang et al., 2020a) consists of a stack of inter-connected layers. Each layer predicts whether a text region of certain length is a complete entity mention.
- **BiFlaG** (Yu et al., 2020) designs a bipartite flat-graph network with two interacting subgraph modules for outermost entities and inner entities, respectively.
- **HIT** (Wang et al., 2020b) leverages the head-tail pair and token interaction to express the nested entities.
- **ARN** (Lin et al., 2019) designs a sequence-to-nuggets architecture by modeling and levraging the head-driven phrase structures of entity mentions.
- **Seq2seq** (Straková et al., 2019) views the nested NER as a sequence-to-sequence problem.
- **KBP17-Best** (Ji et al., 2017) gives an overview of the Entity Discovery task and reports previous best results for the task of nested NER.

We didn't compare our model with BERT-MRC (Li et al., 2020b), because it uses additional external resources to construct the questions, which essentially introduces descriptive information about the categories.

A.3 Detailed Parameter Settings

In our experiments, the detailed parameter settings for the model are shown in Table 6.

A.4 Analysis of Boundary Offset Regression

We analyzed the distribution of the boundary offsets predicted by the model on the ACE04 dataset, as shown in Figure 3. We can find that the numbers of offsets by 0, 1, 2, 3 and $\geq$ 4 are 2162, 2440, 888, 368 and 202, respectively. Most of the offsets are 1, indicating that most of the seed spans require slight boundary adjustments to accurately locate the entities. There are also many offsets of 0. This is because many entities in the dataset are short and the seed spans can cover them, and their boundaries do not need to be adjusted.

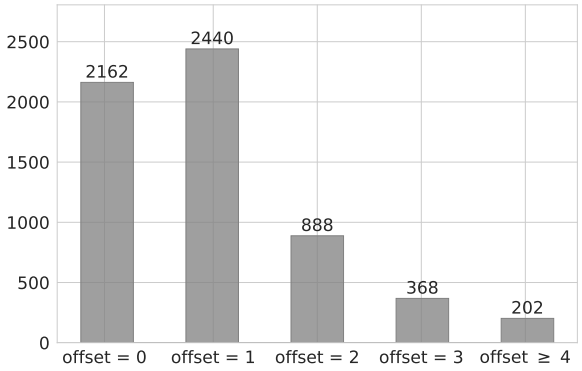

Figure 3: Boundary Offset Statistics

B Experiments on Flat NER

B.1 Datasets

We use two flat NER datasets to evaluate our model:

CoNLL03 English is an English dataset (Tjong Kim Sang and De Meulder, 2003) with four types of flat entities: Location, Organization, Person and Miscellaneous. Following Lin et al. (2019), we train our model on the concatenation of the train and dev set.

Weibo Chinese is a Chinese dataset (Peng and Dredze, 2015) sampled from Weibo with four types of flat entities, including Person, Organization, Location and Geo-political. And we evaluate our model using the same setting with Li et al. (2020a).

B.2 Baselines

For English flat NER, we use several taggers as baseline models, including **ELMO-Tagger** (Peters et al., 2018), **BERT-Tagger** (Peters et al., 2018), which using ELMO, BERT as encoder respectively. And for Chinese flat NER, we use **Glyce** (Meng

Dataset Statistics	ACE04			ACE05			KBP17			GENIA	
	Train	Dev	Test	Train	Dev	Test	Train	Dev	Test	Train	Test
# sentences	6200	745	812	7194	969	1047	10546	545	4267	16692	1854
# sent. nested entities	2712	294	388	2691	338	320	2809	182	1223	3522	446
avg sentence length	22.50	23.02	23.05	19.21	18.93	17.2	19.62	20.61	19.26	25.35	25.99
# total entities	22204	2514	3035	24441	3200	2993	31236	1879	12601	50509	5506
# nested entities	10149	1092	1417	9389	1112	1118	8773	605	3707	9064	1199
nested percentage (%)	45.71	46.69	45.61	38.41	34.75	37.35	28.09	32.20	29.42	17.95	21.78

Table 5: Statistics of the datasets used in the experiments.

P	**ACE04**	**ACE05**	**KBP17**	**GENIA**
lr	3e-05	3e-05	5e-5	5e-6
windows	[1-7, 9, 11, 13, 15]			[1-10]
batch size	8	8	4	6
γ	2.0			
α_1	0.7			
α_2	1.0			
η	1.0			
u	0.9	0.8	0.9	0.9
k	0.6	0.7	0.6	0.7
δ	0.55	0.5	0.5	0.45
$\lambda_1, \lambda_2, \lambda_3$	[1.0, 0.1 ,1.0]			

Table 6: Detailed Parameter(P) Settings

et al., 2019), **FLAT** (Li et al., 2020a) and **SLK-NER** (Hu and Wei, 2020) as baseline models. They incoprate glyph information, phrase embeddings and second-order lexicon knowledge for Chinese NER respectively.

B.3 Results

We evaluated our model on the flat NER dataset, as shown in Table 7. Our model outperforms the baseline models on Weibo Chinese, improving the F1-score by 0.61%. On CoNLL03, our model also achieves comparable results, with less than 1% performance drop compared to the (Yu et al., 2020).

Model	CoNLL03 English		
	Pr.	Rec.	F1
Peters et al. (2018)	-	-	92.22
Devlin et al. (2019)	-	-	92.80
Yu et al. (2020)	93.70	93.30	93.50
Ours	92.13	**93.73**	92.94

Model	Weibo Chinese		
	Pr.	Rec.	F1
Hu and Wei (2020)	61.80	66.30	64.00
Meng et al. (2019)	67.6	67.68	67.71
Li et al. (2020a)	-	-	68.55
Ours	**70.11**	**68.12**	**69.16**

Table 7: Results for ***flat*** NER tasks