A Sequence-to-Set Network for Nested Named Entity Recognition

Zeqi Tan[*] , **Yongliang Shen**[*] , **Shuai Zhang** , **Weiming Lu**[†] , **Yueting Zhuang**

College of Computer Science and Technology, Zhejiang University

{zqtan, syl, zsss, luwm, yzhuang}@zju.edu.cn

Abstract

Named entity recognition (NER) is a widely studied task in natural language processing. Recently, a growing number of studies have focused on the nested NER. The span-based methods, considering the entity recognition as a span classification task, can deal with nested entities naturally. But they suffer from the huge search space and the lack of interactions between entities. To address these issues, we propose a novel sequence-to-set neural network for nested NER. Instead of specifying candidate spans in advance, we provide a fixed set of learnable vectors to learn the patterns of the valuable spans. We utilize a non-autoregressive decoder to predict the final set of entities in one pass, in which we are able to capture dependencies between entities. Compared with the sequence-to-sequence method, our model is more suitable for such unordered recognition task as it is insensitive to the label order. In addition, we utilize the loss function based on bipartite matching to compute the overall training loss. Experimental results show that our proposed model achieves state-of-the-art on three nested NER corpora: ACE 2004, ACE 2005 and KBP 2017. The code is available at https://github.com/zqtan1024/sequence-to-set.

1 Introduction

Named entity recognition (NER) is a fundamental task in natural language processing because named entities often contain the key information [Lample *et al.*, 2016]. Nested named entities refer to that internal named entities are contained in external named entities as in Figure 1, in which "*nyu*" and "*a professor of psychiatry at nyu*" are both named entities. Such nested entities are common in practice. For example, about 38% of the entities in the ACE 2005 training dataset are nested. Traditional work [Huang *et al.*, 2015; Lample *et al.*, 2016; Chiu and Nichols, 2016] models named entity recognition as a sequence labeling task, thus enabling to leverage RNN-based methods to recognize flat entities.

---

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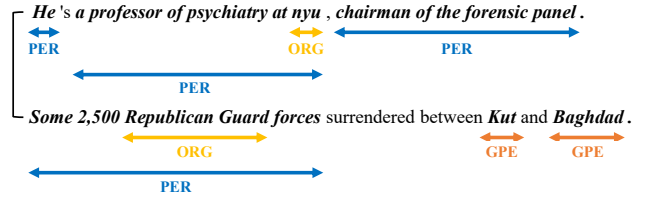
[†]Corresponding author.

Figure 1: Examples for *nested* entities from ACE05 corpora.

However, since tokens in nested entities may belong to multiple labels, the traditional sequence labeling approaches cannot meet the demand.

Various methods have been proposed to deal with the nested NER task, including the sequence-to-sequence methods [Ju *et al.*, 2018; Straková *et al.*, 2019; Jue *et al.*, 2020] and the span-based methods [Sohrab and Miwa, 2018; Zheng *et al.*, 2019; Yu *et al.*, 2020]. The sequence-to-sequence method treats nested NER as a sequence generation task in which labels are decoded one by one in order. However, in the named entity recognition task, the output labels are essentially an unordered set. As shown in Figure 1, there is apparently no ordinal relationship between the two GPE entities ("*Kut*" and "*Baghdad*"). Straková *et al.* [2019] generates the output labels sequentially which is sensitive to the label order. In this manner, even if the model predicts all correct labels, it may cause an unreasonable training loss as a result of inconsistent order. The span-based models classify the candidate spans which are extracted from a text sequence through various approaches. Sohrab and Miwa [2018] enumerates all possible spans within a limit length and then predicts their classes. Unlike the exhaustive method, Zheng *et al.* [2019] first identifies the left and right boundaries separately and then matches them to form candidate spans. Similarly, ARN [Lin *et al.*, 2019] recognizes the spans of interest based on previously identified anchor words. Recently, Yu *et al.* [2020] scores candidate spans via the biaffine model [Dozat and Manning, 2016] and reaches the state-of-the-art. Although the above span-based methods achieve good performances, they also face some difficulties. The span-selecting methods face the problem of error propagation, because the boundaries or anchors are tend to be identified incorrectly, while the span-enumerating methods need to search for all possible regions. Besides, the candidate spans in these methods do not interact

directly with each other and thus inappropriately ignore the dependencies between named entities.

To address the above issues, we propose a novel model that treats the named entity recognition as a sequence-to-set task. Considering that the entities in a sequence are inherently unordered, we expect to predict the entire set of entities in one pass. Therefore our model is no longer as sensitive to the label order as the sequence-to-sequence model. Compared to the span-based methods, instead of specifying candidate spans in advance, we provide a fixed set of learnable vectors that go through the entire train data to learn the patterns of the valuable spans. Hence we do not need to search for all possible spans anymore. Specifically, our model consists of three components: a sequence encoder, an entity set decoder and a loss function based on bipartite matching. We utilize BERT [Devlin *et al.*, 2019] and BiLSTM [Huang *et al.*, 2015] to construct the sequence encoder. In order to predict all entities of the sequence in one pass, we design a non-autoregressive entity set decoder. It receives the sequence encoding and a set of learnable vectors, which are called entity queries, to decode the final entity set. Furthermore, our decoder is able to capture the dependencies between entities quite naturally through a self-attention mechanism between entity queries. At the end, we utilize the loss function based on bipartite matching to compute the overall training loss.

Our main contributions are as follow:

- We propose a novel sequence-to-set network for the prediction of the entity set. To the best of our knowledge, we are the first to consider named entity recognition as a sequence-to-set task. In addition, we predict the final entity set in one pass, while the sequence-to-sequence model predicts entities one by one. Since entities are inherently unordered, our model which is insensitive to the label order achieves a better performance.

- We provide a fixed set of entity queries to replace the explicit candidate spans, thus eliminating the need to search for all possible spans. Besides, we are able to capture the dependencies between entities by using a self-attention mechanism which performs direct interactions between entity queries.

- Experimental results show that our model achieves state-of-the-art on three widely used datasets, and outperforms several competing baseline models on F1 score by 0.56% on ACE 2004, 1.65% on ACE 2005 and 2.99% on KBP 2017.

2 Model

In order to transform the input sequence to the entity set, our model has three necessary components: a sequence encoder, an entity set decoder and a loss function based on bipartite matching. Firstly, the sequence encoder obtains rich contextual information from the input sentence. The entity set decoder then receives the sequence representation from the encoder and a set of entity queries to decode the left and right boundaries and categories of the predicted entity set. The bipartite matching is utilized to assign a unique prediction to each target entity so that the prediction loss of the entire

model can be computed. The overview of our sequence-to-set model is shown in Figure 2. In the following subsections, we will introduce them in more detail.

2.1 Sequence Encoder

We use BERT [Devlin *et al.*, 2019] and BiLSTM [Huang *et al.*, 2015] to construct our sequence encoder. Given a input sequence, we initially represent its i-th token by concatenating the BERT contextualized embeddings x_i^{bert}, the GloVE [Pennington *et al.*, 2014] embeddings x_i^{glove}, part-of-speech (POS) embeddings x_i^{pos} and character-level embeddings x_i^{char} together. We follow Yu *et al.* [2020] to obtain the contextualized embeddings x_i^{bert} by encoding a target token with one surrounding tokens each side. The character-level embeddings are produced by a BiLSTM module which is the same as Lample *et al.* [2016]. Then, the token representations $\{x_i\}$ are feed into another BiLSTM to get the final sequence representation $H \in \mathbb{R}^{l \times d}$ as:

$$x_i = x_i^{bert} \oplus x_i^{glove} \oplus x_i^{pos} \oplus x_i^{char}, \tag{1}$$

$$\overrightarrow{H}_i = \text{LSTM}_f\left(x_i, \overrightarrow{H}_{i-1}; \theta_f\right), \tag{2}$$

$$\overleftarrow{H}_i = \text{LSTM}_b\left(x_i, \overleftarrow{H}_{i+1}; \theta_b\right), \tag{3}$$

$$H_i = \overrightarrow{H}_i \oplus \overleftarrow{H}_i, \tag{4}$$

where l is the sequence length and d is twice the hidden size of LSTM, $\oplus$ denotes the concatenation operation and θ_f and θ_b denote the parameters of the forward and backward LSTM. The $\overrightarrow{H}_i$ and $\overleftarrow{H}_i$ are the hidden states at the position i of the forward and backward LSTM.

2.2 Entity Set Decoder

The entity set decoder follows the classical transformer framework and uses self-attention as well as cross-attention mechanisms to transform N entity queries. Different from the sequence-to-sequence method [Straková *et al.*, 2019] to predict entities one by one, we aim to decode all entities of a sequence in parallel due to the inherent disorder of entities. Similar to [Gu *et al.*, 2018], we use a non-autoregressive decoder thereby allowing us to obtain N predictions in one pass. Through the self-attention mechanism between entity queries, the decoder is able to capture the dependencies between entities. The decoder can also effectively acquire contextual information via the cross-attention mechanism. Because the decoding manner is non-autoregressive, we do not need to adopt the masking mechanism to prevent information leakage and thus can obtain the complete contextual semantic information. In addition, since the set of entity queries is used as the input for each sequence during the training process, we have an access to the global view of the entire dataset.

The N entity queries denoted by $\mathbf{Q}_{span}$ are transformed into output embeddings by the entity set decoder of M layers. This process mainly involves the multi-head attention
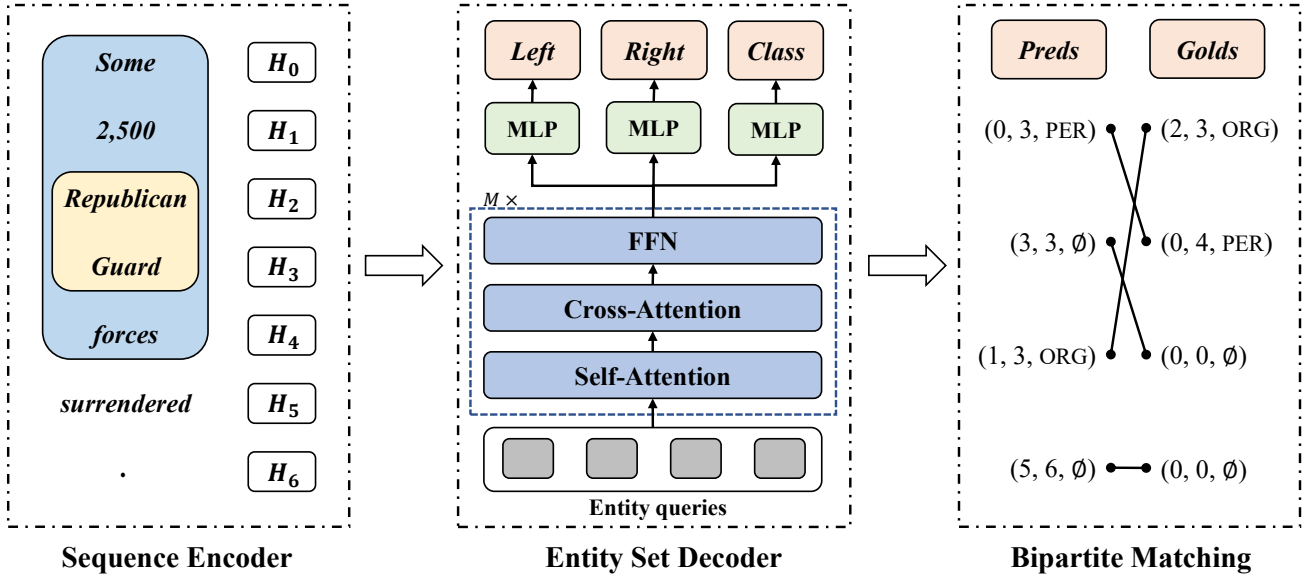
Figure 2: The architecture of our Sequence-to-Set Network. The representation of each token in sentence "*Some 2,500 Republican Guard forces surrendered.*" is fed into the entity set decoder along with the entity queries. Then the decoder transforms the queries to predicted entities. Finally, we score them by the loss function based on bipartite matching.

mechanism [Vaswani *et al.*, 2017]. For simplicity, we denote the attention as the following equation:

$$\text{Attention} (\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax} \left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} \right) \mathbf{V}, \quad (5)$$

where $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ are the query matrix, key matrix and value matrix respectively, and the $1/\sqrt{d_k}$ is the scaling factor. In the self-attention, $\mathbf{Q} = \mathbf{K} = \mathbf{V} = \mathbf{Q}_{span}$. And in the cross-attention, $\mathbf{Q} = \mathbf{Q}_{span}$ while $\mathbf{K} = \mathbf{V} = \mathbf{H}$. The multi-head attention can be formulated as follows:

$$\text{head}_i = \text{Attention} \left(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V \right), \quad (6)$$

$$\mathbf{R} = \text{Concat} (\text{head}_1, \dots, \text{head}_h) \mathbf{W}^O, \quad (7)$$

where $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, $\mathbf{W}_i^V$ and $\mathbf{W}^O$ are trainable projection parameters, and h is the number of head. We take $\mathbf{R}$ computed by the cross-attention module as the input of FFN.

The FFN module is composed of a 3-layer perceptron with ReLU activation function and a linear projection layer. Through FFN, we denote the final output embeddings as $\mathbf{U} \in \mathbb{R}^{N \times d}$. We add an additional label $\varnothing$ to indicate that no entity is recognized because we predict a fixed-size set of N entities, where N is set larger than the actual number of entities in a sequence. Given an entity query $u \in \mathbb{R}^d$ in U, the classification process can be defined by:

$$p^c = \text{MLP}_c(u), \quad (8)$$

$$\text{H}_{fuse} = \text{dup}(u, l) \oplus \text{H}, \quad (9)$$

$$p^l = \text{MLP}_l(\text{H}_{fuse}), \quad (10)$$

$$p^r = \text{MLP}_r(\text{H}_{fuse}), \quad (11)$$

where p^c, p^l and p^r denote the probability of classification for classes, left boundaries and right boundaries respectively. $\oplus$ denotes the concatenation operation and MLP denotes the multilayer perceptron with softmax function in the last layer. The function dup will duplicate u for l times into shape $\mathbb{R}^{l \times d}$.

2.3 Bipartite Matching

After decoding the N predictions, the main difficulty in training is to score the predicted entities (*left*, *right*, *class*) according to the golden entities. To cope with the problem, we design a loss function based on bipartite matching. Before calculating the training loss, we first need to find a optimal matching between the predicted entity set and the golden entity set. We denote the golden set of entities by y, and the set of N predictions by $\hat{y} = \{\hat{y}_i\}_{i=1}^N$. We also pad y to the size of N with $\varnothing$. To find the optimal matching we search for a permutation of N elements $\beta \in \mathcal{O}_N$ with the lowest cost:

$$\hat{\beta} = \arg\min_{\beta \in \mathcal{O}_N} \sum_i^N \mathcal{L}_{\text{match}} \left(y_i, \hat{y}_{\beta(i)} \right), \quad (12)$$

where $\mathcal{L}_{\text{match}} \left(y_i, \hat{y}_{\beta(i)} \right)$ is a pair matching cost between the golden entity y_i and a prediction with index $\beta(i)$. We compute this optimal assignment efficiently by using the Hungarian algorithm [Kuhn, 1955]. Considering the left boundaries, the right boundaries and the classes of entities, each element i of the golden entity set can be seen as a $y_i = (l_i, r_i, c_i)$. We define $\mathcal{L}_{\text{match}} \left(y_i, \hat{y}_{\beta(i)} \right)$ as:

Dataset Statistics	ACE 2004			ACE 2005			KBP 2017			GENIA	
	Train	Dev	Test	Train	Dev	Test	Train	Dev	Test	Train	Test
# sentences	6200	745	812	7194	969	1047	10546	545	4267	16692	1854
# with 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 1: Statistics of the datasets used in the experiments.

$$\mathcal{L}_{\text{match}}\left(y_i, \hat{y}_{\beta(i)}\right) = -\mathbb{1}_{\{c_i \neq \varnothing\}} \left[p^c_{\beta(i)}\left(c_i\right) \right.$$
$$+ p^l_{\beta(i)}\left(l_i\right) \qquad (13)$$
$$\left. + p^r_{\beta(i)}\left(r_i\right) \right].$$

After we get the optimal matching $\hat{\beta}(i)$, we define the final loss $\mathcal{L}(y, \hat{y})$ as:

$$\mathcal{L}(y, \hat{y}) = \sum_{i=1}^{N} \left\{ -\log p^c_{\hat{\beta}(i)}\left(c_i\right) \right.$$
$$+ \mathbb{1}_{\{c_i \neq \varnothing\}} \left[-\log p^l_{\hat{\beta}(i)}\left(l_i\right) \right. \qquad (14)$$
$$\left. \left. -\log p^r_{\hat{\beta}(i)}\left(r_i\right) \right] \right\}.$$

3 Experiments

3.1 Experimental Setup

In the experimental setup, we utilize the widely used ACE 2004, ACE 2005, KBP 2017 and GENIA datasets. For ACE 2004 and ACE 2005, we follow the previous work [Katiyar and Cardie, 2018; Lin *et al.*, 2019] to keep files from bn, nw and wl and divide these files into train, dev and test sets in the ratio of 8:1:1, respectively. For GENIA, we use genia-corpus3.02 as in Katiyar and Cardie [2018]. For KBP 2017, we use the 2017 English evaluation dataset and the same split strategy in Lin *et al.* [2019]. Table 1 shows the statistical results in detail for all the above datasets.

Standard precision, recall and F1-measure are employed as evaluation metrics. An entity is considered correct only if both the entity boundary and the entity label are correct. The checkpoint that has the best F1 score in the development set is chosen to evaluate the test set.

For comparison with Jue *et al.* [2020] and Yu *et al.* [2020], we use the cased large version of BERT in most experiments. We also adopt the GloVe [Pennington *et al.*, 2014] 100-dimension pre-trained word vectors. The character-level BiLSTM layer is set to 1 and the token-level BiLSTM layer is set to 3. The hidden size of the former is 50, and the hidden size of the latter is half of the BERT hidden size. Specifically, on the GENIA dataset, we replace BERT with BioBERT [Lee *et al.*, 2019], GloVE with BioWordvec [Chiu *et al.*, 2016]. The number of entity queries N is set to 60 and the vectors are randomly initialized with the normal distribution $\mathcal{N}(0.0, 0.02)$.

Model	ACE 2004		
	P	R	F1
Katiyar and Cardie [2018]	73.60	71.80	72.70
Shibuya and Hovy [2019]	83.73	81.91	82.81
Straková *et al.* [2019]	-	-	84.40
Jue *et al.* [2020]	86.08	86.48	86.28
Yu *et al.* [2020]	87.30	86.00	86.70
Ours	88.46	86.10	**87.26**

Model	ACE 2005		
	P	R	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
Shibuya and Hovy [2019]	82.98	82.42	82.70
Straková *et al.* [2019]	-	-	84.33
Jue *et al.* [2020]	83.95	85.39	84.66
Yu *et al.* [2020]	85.20	85.60	85.40
Ours	87.48	86.63	**87.05**

Model	KBP 2017		
	P	R	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.* [2020]	80.97	81.12	80.97
Ours	84.91	83.04	**83.96**

Model	GENIA		
	P	R	F1
Lin *et al.* [2019]	75.80	73.90	74.80
Luo and Zhao [2020]	77.40	74.60	76.00
Wang *et al.* [2020]	78.10	74.40	76.20
Shibuya and Hovy [2019]	78.07	76.45	77.25
Straková *et al.* [2019]	-	-	78.31
Jue *et al.* [2020]	79.45	78.94	79.19
Yu *et al.* [2020]	81.80	79.30	**80.50**
Ours	82.31	78.66	80.44

Table 2: The overall performances for ***nested*** NER tasks.

The number of our decoder layer M is set to 3. The number of the attention heads is set to 8. The number of the MLP layer is set to 1. We use the AdamW [Loshchilov and Hutter, 2017] optimizer with a linear warmup-decay learning rate schedule (with peak learning rate of 2e-5 and epoch of 100), a dropout with the rate of 0.1 and a batch size of 8.

3.2 Baselines

We compare our model with several start-of-the-art methods on ACE 2004, ACE 2005, KBP 2017 and GENIA datasets:

- **Biaffine:** Yu *et al.* [2020] utilizes the biaffine model to score pairs of start and end tokens in a sequence.
- **Pyramid:** Jue *et al.* [2020] designs the normal and inverse pyramidal structures to identify entities through bidirectional interactions.
- **BiFlaG:** Luo and Zhao [2020] proposes a bipartite flat-graph network with two interacting subgraph modules.
- **HIT:** Wang *et al.* [2020] designs head-tail detector and token-interaction tagger to express the nested structure.
- **Seq2seq:** Straková *et al.* [2019] considers the nested NER as a sequence-to-sequence problem. The encoded labels are predicted one by one by the decoder.
- **Second-best:** Shibuya and Hovy [2019] searches a span of each extracted entity for nested entities with second-best sequence decoding.
- **ARN:** Lin *et al.* [2019] leverages the head-driven phrase structures to predict nested named entities.
- **Hyper-Graph:** Katiyar and Cardie [2018] makes use of the BILOU tagging scheme to learn the hypergraph representation.
- **KBP17-Best:** Ji *et al.* [2017] reports the best previous results for nested NER tasks.

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

3.3 Overall Performances

The overall performances of the proposed model against baselines for nested NER tasks on ACE 2004, ACE 2005, KBP 2017 and GENIA datasets are shown in Table 2. Our proposed model outperforms the state-of-the-art model on ACE 2004, ACE 2005 and KBP 2017 datasets. To be specific, the F1 scores of our model improve by +0.56%, +1.65% and +2.99% over previous SOTA performances on ACE 2004, ACE 2005 and KBP 2017, respectively. Meanwhile, we achieve the comparable result with the SOTA model on GENIA dataset. The experimental results demonstrate the ability of our model to recognize named entities in the sequence.

The main improvement of the proposed model comes from the fact that we treat the nested NER as a sequence-to-set problem, which is consistent with the inherent disorder of entities. Compared with the sequence-to-sequence model, our model is insensitive to the label order and achieves a better performance.

# Entity Query	ACE 2005		
	P	R	F1
40	86.84	86.93	86.88
60	87.48	86.63	**87.05**
80	87.13	86.23	86.68
100	86.28	86.60	86.44
200	86.63	85.53	86.08

Table 3: The performances with different number of entity queries.

# Layer	Interaction	ACE 2005		
		P	R	F1
1	✗	82.14	82.85	82.49
1	✓	86.64	84.12	85.36
2	✗	83.57	83.46	83.51
2	✓	85.80	86.63	86.21
3	✗	85.26	85.23	85.24
3	✓	87.48	86.63	**87.05**

Table 4: Ablation studies for decoder layer number and interaction between entity queries.

In addition, in order to choose an appropriate number of entity queries, we design a comparison experiment on ACE 2005 with a maximum number of entities of 27. The evaluation results are shown in Table 3. We observe that when the number is obviously larger than the ground truth, the performance of the model does decrease. Eventually, the query number N in our experiments is set to 60.

3.4 Ablation Studies

To demonstrate the effectiveness of the several modules in our method, we conduct a series of ablation studies on ACE 2005. First, according to our intuition, the more decoder layers we stack, the more expressive the model is. Table 4 verifies this point: when we reduce the number of decoder layer from 3 to 2 and from 2 to 1, the model performance decreases by 0.84% and 0.85%, respectively. We also conduct experiments with 6 layers, but the model performance does not improve obviously. Furthermore, from Table 4 we also observe that regardless of the number of layers, the performance of the model consistently decreases with the absence of the interaction (*self attention*) between entity queries. When the number of decoder layer is 3, 2 and 1, the results decrease by 1.81%, 2.70% and 2.87% respectively. This is because the decoder can capture the dependencies between entities by using a self-attention mechanism which performs direct interactions between entity queries.

Likewise, we explore the learning ability of entity queries and the effectiveness of bipartite matching loss. In Table 5, Freeze Queries means we keep the parameters of the entity queries unchanged, and CE Loss refers to replace bipartite matching loss with cross-entropy loss. We can see that freezing the entity queries results in the performance drops of 0.40% and 0.81% on ACE 2004 and ACE 2005, respectively. This indicates that the entity queries do learn the patterns

Settings	ACE 2004		
	P	R	F1
Freeze Queries	87.37	86.36	86.86
CE Loss	85.55	83.66	84.59
Full Model	88.46	86.10	**87.26**

Settings	ACE 2005		
	P	R	F1
Freeze Queries	85.78	86.70	86.24
CE Loss	85.24	86.66	85.94
Full Model	87.48	86.63	**87.05**

Table 5: Ablation studies for the learnable entity queries and bipartite matching loss.

of the valuable regions. Besides, we can gain improvement of 2.67% and 1.11% on ACE 2004 and ACE 2005 respectively by replacing cross-entropy loss with bipartite matching loss, which shows the power of bipartite matching loss on the unordered prediction task. Compared with the sequence-to-sequence model, we can not only predict all named entities in one pass, but are able to utilize the more suitable bipartite matching loss for the unordered entity recognition task.

4   Related Work

Recent nested named entity recognition methods can be categorized as sequence-based [Ju *et al.*, 2018; Jue *et al.*, 2020], hypergraph-based [Lu and Roth, 2015; Wang and Lu, 2018; Katiyar and Cardie, 2018], and span-based [Xu *et al.*, 2017; Sohrab and Miwa, 2018; Zheng *et al.*, 2019; Yu *et al.*, 2020] approaches. Ju *et al.* [2018] proposed a layered model to recognize entities from inside to outside sequentially. Pyramid [Jue *et al.*, 2020] were designed with normal and inverse pyramidal structures to identify entities through bidirectional interactions. Lu and Roth [2015] were the first to propose a hypergraph-based approach to cope with the nested entity mention detection problem. Wang and Lu [2018] utilized segmental hypergraph to model arbitrary combinations of mentions. Katiyar and Cardie [2018] made use of the BILOU tagging scheme to learn the hypergraph representation. The span-based models classify the candidate spans which are extracted from a text sequence through various approaches. Exhaustive Model [Sohrab and Miwa, 2018] enumerated all possible spans in a sentence within a limited length and afterwards to predict their categories. Luan *et al.* [2019] used a dynamic span graphs to select the most confident entity spans. Unlike the above methods, BERT-MRC [Li *et al.*, 2020] formulated the named entity recognition problem as a machine reading comprehension task so that the flat and nested cases can be handled uniformly. Other works [Lin *et al.*, 2019; Fisher and Vlachos, 2019; Straková *et al.*, 2019] also presented various approaches to solve the nested NER problem.

We consider the nested named entity recognition as a sequence-to-set task, thus avoiding the error propagation problem of the sequence-based methods [Ju *et al.*, 2018]. Compared to the span-based methods, we provide a fixed set

of entity queries to learn the patterns of the valuable spans. Thus we do not need to search for all possible regions anymore. Besides, we predict the final set of entities in one pass, while the sequence-to-sequence method [Straková *et al.*, 2019] predicts entities one by one. Since the entities are inherently unordered, our model which is insensitive to the label order achieves a better performance. Similarly, DETR [Carion *et al.*, 2020] treats the object detection as a set prediction task. This end-to-end approach effectively eliminates the need for many manually designed components in the two-stage methods such as Faster R-CNN [Ren *et al.*, 2016].

5   Conclusion

In this paper, we propose a novel sequence-to-set network for nested NER. Due to the fact that the entities in a sequence are inherently unordered, we design a non-autoregressive decoder to generate the set of entities in one pass. To score the predicted entities respect to the golden entities, we design a bipartite matching loss which is more suitable for the unordered entity recognition task. Compared with the sequence-to-sequence model, our model is insensitive to the label order and achieves a better performance. Besides, we can capture the dependencies between entities by using a self-attention mechanism which performs direct interactions between entity queries. Experiments show that our model achieves state-of-the-art on ACE 2004, ACE 2005 and KBP 2017 datasets. In the future, how to represent the dependencies between entities better will be considered.

References

[Carion *et al.*, 2020] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *arXiv preprint arXiv:2005.12872*, 2020.

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

[Chiu *et al.*, 2016] Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. 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, August 2016.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL 2019*, pages 4171–4186, Minneapolis, Minnesota, June 2019.

[Dozat and Manning, 2016] Timothy Dozat and Christopher D. Manning. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734, 2016.

[Fisher and Vlachos, 2019] Joseph Fisher and Andreas Vlachos. Merge and label: A novel neural network architecture for nested ner. In *Proceedings of ACL 2019*, pages 5840–5850, 2019.

[Gu *et al.*, 2018] Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. In *ICLR*, 2018.

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

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

[Ju *et al.*, 2018] Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. A neural layered model for nested named entity recognition. In *Proceedings of NAACL 2018*, pages 1446–1459, 2018.

[Jue *et al.*, 2020] Wang Jue, Lidan Shou, Ke Chen, and Gang Chen. Pyramid: A layered model for nested named entity recognition. In *Proceedings of ACL 2020*, pages 5918–5928, 2020.

[Katiyar and Cardie, 2018] Arzoo Katiyar and Claire Cardie. Nested named entity recognition revisited. In *Proceedings of ACL 2018*, pages 861–871, 2018.

[Kuhn, 1955] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

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

[Lee *et al.*, 2019] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 09 2019.

[Li *et al.*, 2020] Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. A unified MRC framework for named entity recognition. In *Proceedings of ACL 2020*, pages 5849–5859, Online, July 2020.

[Lin *et al.*, 2019] Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. Sequence-to-nuggets: Nested entity mention detection via anchor-region networks. *arXiv preprint arXiv:1906.03783*, 2019.

[Loshchilov and Hutter, 2017] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017.

[Lu and Roth, 2015] Wei Lu and Dan Roth. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of EMNLP 2015*, pages 857–867, 2015.

[Luan *et al.*, 2019] Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. A general framework for information extraction using dynamic span graphs. In *Proceedings of NAACL 2019*, pages 3036–3046, 2019.

[Luo and Zhao, 2020] Ying Luo and Hai Zhao. Bipartite flat-graph network for nested named entity recognition. In *Proceedings of ACL 2020*, pages 6408–6418, Online, July 2020. Association for Computational Linguistics.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543, Doha, Qatar, October 2014.

[Ren *et al.*, 2016] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

[Shibuya and Hovy, 2019] Takashi Shibuya and Eduard Hovy. Nested named entity recognition via second-best sequence learning and decoding. *arXiv preprint arXiv:1909.02250*, 2019.

[Sohrab and Miwa, 2018] Mohammad Golam Sohrab and Makoto Miwa. Deep exhaustive model for nested named entity recognition. In *Proceedings of EMNLP 2018*, pages 2843–2849, 2018.

[Straková *et al.*, 2019] Jana Straková, Milan Straka, and Jan Hajic. Neural architectures for nested ner through linearization. In *Proceedings of ACL 2019*, pages 5326–5331, 2019.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[Wang and Lu, 2018] Bailin Wang and Wei Lu. Neural segmental hypergraphs for overlapping mention recognition. In *Proceedings of EMNLP 2018*, pages 204–214, 2018.

[Wang *et al.*, 2020] Yu Wang, Yun Li, Hanghang Tong, and Ziye Zhu. HIT: Nested named entity recognition via head-tail pair and token interaction. In *Proceedings of EMNLP 2020*, pages 6027–6036, Online, November 2020.

[Xu *et al.*, 2017] Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. A local detection approach for named entity recognition and mention detection. In *Proceedings of ACL 2017*, pages 1237–1247, 2017.

[Yu *et al.*, 2020] Juntao Yu, Bernd Bohnet, and Massimo Poesio. Named entity recognition as dependency parsing. *arXiv preprint arXiv:2005.07150*, 2020.

[Zheng *et al.*, 2019] Changmeng Zheng, Yi Cai, Jingyun Xu, Ho-fung Leung, and Guandong Xu. A boundary-aware neural model for nested named entity recognition. In *Proceedings of EMNLP-IJCNLP*, pages 357–366, 2019.