A Supervised Multi-Head Self-Attention Network for Nested Named Entity Recognition

Yongxiu Xu,[1,2] **Heyan Huang,** [3*] **Chong Feng,** [3] **Yue Hu** [1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3] School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
xuyongxiu@iie.ac.cn, {hhy63,fengchong}@bit.edu.cn, huyue@iie.ac.cn

Abstract

In recent years, researchers have shown an increased interest in recognizing the overlapping entities that have nested structures. However, most existing models ignore the semantic correlation between words under different entity types. Considering words in sentence play different roles under different entity types, we argue that the correlation intensities of pairwise words in sentence for each entity type should be considered. In this paper, we treat named entity recognition as a multi-class classification of word pairs and design a simple neural model to handle this issue. Our model applies a supervised multi-head self-attention mechanism, where each head corresponds to one entity type, to construct the word-level correlations for each type. Our model can flexibly predict the span type by the correlation intensities of its head and tail under the corresponding type. In addition, we fuse entity boundary detection and entity classification by a multi-task learning framework, which can capture the dependencies between these two tasks. To verify the performance of our model, we conduct extensive experiments on both nested and flat datasets. The experimental results show that our model can outperform the previous state-of-the-art methods on multiple tasks without any extra NLP tools or human annotations.

Introduction

Named Entity Recognition (NER) is an important task in Natural Language Processing (NLP), which aims to identify text spans to specific entity types such as Person, Location, Organization, *etc*. Although various NER methods have been proposed, the widely-used models usually regard NER as a sequence labeling task, which implicitly assumes each token only has one tag. For example, (Lample et al. 2016; Huang, Xu, and Yu 2015) exploit a LSTM-CRF neural network which encodes tokens by a Long Short-Term Memory (LSTM) layer and predicts labels by a Conditional Random Fields (CRF) layer. However, entities with nested structures are common in various domains (Alex, Haddow, and Grover 2007; Byrne 2007; Villodre et al. 2007; Wang 2009). As shown in Figure 1, the entity *[mouse IL-2R alpha]* is nested in the entity *[mouse IL-2R alpha gene]*. Taking nested entities into consideration can facilitate many downstream NLP tasks, including relation extraction (Miwa and Bansal 2016; Liu et al. 2017), question answering (Wang et al. 2017), entity linking (Gupta, Singh, and Roth 2017; Martins, Marinho, and Martins 2019) and coreference resolution (Chang, Samdani, and Roth 2013; Fragkou 2017).

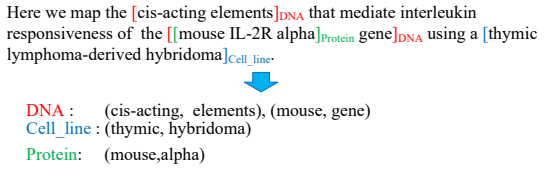

Figure 1: An example in the GENIA dataset. For different given entity types, we will generate the head-tail pair of entities with corresponding types.

Numerous approaches for nested NER have been proposed in recent years. One representative category is based on the sequence labeling (Liu et al. 2019; Xin et al. 2018) that convert the nested structures into flat structures by various transform operations, such as (Finkel and Manning 2009) construct a syntactic constituency tree, (Lu and Roth 2015; Muis and Lu 2017; Wang and Lu 2018; Clark et al. 2018) build hyper-graphs. However, they need a lot of human efforts to annotate the corpus for complex transformations, suffering from error propagation and recognition inaccuracy. Another appealing category is based on span classification that classify candidate spans based on span representations. (Xia et al. 2019) recently propose a Multi-Grained Named Entity Recognition (MGNER) model contains a Detector for detecting all possible word segments and a Classifier for classifying these segments into the pre-defined types.

Although the span-based model overcomes the drawbacks of the sequence labeling based models, it still has some obvious issues. First, learning the span representation for all possible spans in the sentence is computationally expensive. Second, they ignore the explicit boundary information when classify spans, leading to detecting boundaries of entities inaccurate. Observing the problems of span-based models, two recent works (Zheng et al. 2019; Tan et al. 2020) introduce boundary-aware neural models to solve these problems by classifying entity types following entity boundary identification.

Despite the significant progress that has been made by

*Corresponding author
Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

the boundary-aware models, they independently predict the boundary of the entities ignoring the relevance between the head and tail of entities. In addition, for span classification, they still rely on span representation ignoring the correlation between entity spans and entity types. We suppose that if we consider a head-tail pair of an entity span as a point in a type space, there is no overlap between points, as shown in Figure 1. Further more, considering words in the sentence play different roles under different types, we argue that the correlation intensities of the head-tail pair of candidate spans in the sentence are different under different entity types. For example, the head-tail pair *(mouse, alpha)* of span *mouse IL-2R alpha* has higher correlation intensities in the space of type "Protein" than the type "DNA", while the head-tail pair *(mouse, gene)* of span *mouse IL-2R alpha gene* has higher correlation intensities in the space of type "DNA" than the type "Protein".

To alleviate the aforementioned problems, we propose a simple but effective multi-task learning model which mainly contains an entity boundary detection module and an entity classification module. For entity boundary detection, we treat it as a word-level multi-class classification task, and for entity classification, we treat it as a word-pair-level multi-class classification task. The key of our model is presenting a **S**upervised **M**ulti-**H**ead **S**elf-**A**ttention neural method (**SMHSA**) for entity classification. Concretely, we apply two single-layer linear fully connection layer to map each word in sentence into head and tail representation space which can learn to identify the head/tail token of spans. To learn the correlation between words in sentence, we exploit self-attention mechanism. With the consideration that the correlation intensities of a word pair are usually different under different entity types, we map each type into a subspace of the multiple heads and apply self-attention operation over the sentence in each type space. Besides, we introduce a multi-task learning framework for capturing the dependencies between entity boundary detection and entity classification, further improving the performance of our model. The whole architecture of our model is illustrated in Figure 2.

To evaluate the effectiveness of our model, we conduct comprehensive experiment on five benchmark datasets, among which three are nested datasets and other two are flat datasets. For nested datasets, our model achieves 86.3%, 85.4% and 79.6% F1 scores on the ACE2004, ACE2005 and GENIA datasets, respectively. For flat datasets, our model obtains 79.7% and 93.6% F1 scores on the JNLPBA and CoNLL03-English datasets, respectively. Besides, our model dose not depend on any external knowledge resource or complex human annotation, thus it can be easily adapted to domain-specific data. The code of our model will be released for future research[1].

Related Work

Research into named entity recognition has a long history and majority of models regard the NER as a sequence labeling task. Traditional sequence labeling models are based on Hidden Markov Models (HMM) (Zhou and Su 2002) or probabilistic graph models such as CRF (Ratinov and Roth 2009). With the success of deep learning, neural-based models have attracted lots of attention of researchers. (Collobert et al. 2011) exploits Convolutional Neural Network (CNN) to encode tokens paired with a CRF layer for predicting token labels. Later, (Huang, Xu, and Yu 2015) replaces the CNN with a bidirectional LSTM network. Beside word-level word representation, several methods incorporate character-level word embeddings with word-level embeddings, such as, (Lample et al. 2016) utilizes a character LSTM to represent character-level features, (Chiu and Nichols 2016; Ma and Hovy 2016) employs a character CNN instead. However, the aforementioned methods implicitly assumes each token only has one tag, which incapable to extract the overlapping entities with nested structures.

Various approaches for nested NER have been proposed in recent years, which mainly can be divided into two categories: sequence labeling-based models and span-based models. For **sequence labeling-based models**, the motivation is trying to transform the nested NER problem into a standard sequence labeling task inspired by the great success of sequence labeling in flat NER. (Finkel and Manning 2009) proposes a parsing-based method which transform each entity as a constituent in the parsing tree, and predict the entity types using a CRF-based method. (Lu and Roth 2015; Muis and Lu 2017) employ hypergraph transformation for recognizing the nested entities. However, they need a lot human efforts for designing unambiguous hypergraph. (Katiyar and Cardie 2018) also utilizes a hypergraph transformation but the structures of hypergraph are learned by an LSTM network. One issue of the hypergraph-based models is that they design the hypergraph specially for nested NER and are usually not suitable for flat NER. (Ju, Miwa, and Ananiadou 2018) presents a stacked LSTM-CRF NER recognizer which predicts entities from inner to outer iteratively. (Wang et al. 2018) proposes a transition-based model to transform the nested NER into an action sequence labeling task. (Straková, Straka, and Hajic 2019) proposes a sequence-to-sequence labeling model for nested NER. Although the sequence labeling-based models have achieved good performance on nested NER, most of them need extra annotation and complex feature engineering.

Instead of tagging each token by sequence labeling, **span-based models** classify spans based on span representation. This model can effectively avoid the disadvantages of the sequence labeling-based models and achieve decent results on both flat and nested NER tasks. Recently, (Xia et al. 2019) proposes a novel neural framework which consists of a detection component for recognizing all possible entity spans and a classification component for predicting the types of spans extracted. (Sohrab and Miwa 2018; Luan et al. 2019) are other works based on span classification. Although span-based models are simple and effective, its drawbacks are also obvious. For example, they ignore the boundary information and suffer from expensive computation, leading to detecting the entity spans inaccurate and long training. To alleviate these problems, (Tan et al. 2020; Zheng et al. 2019) propose a boundary-ware model to split NER task into two subtasks: boundary detection and span classification. Although

[1]https://github.com/xyxAda/Attention_NER

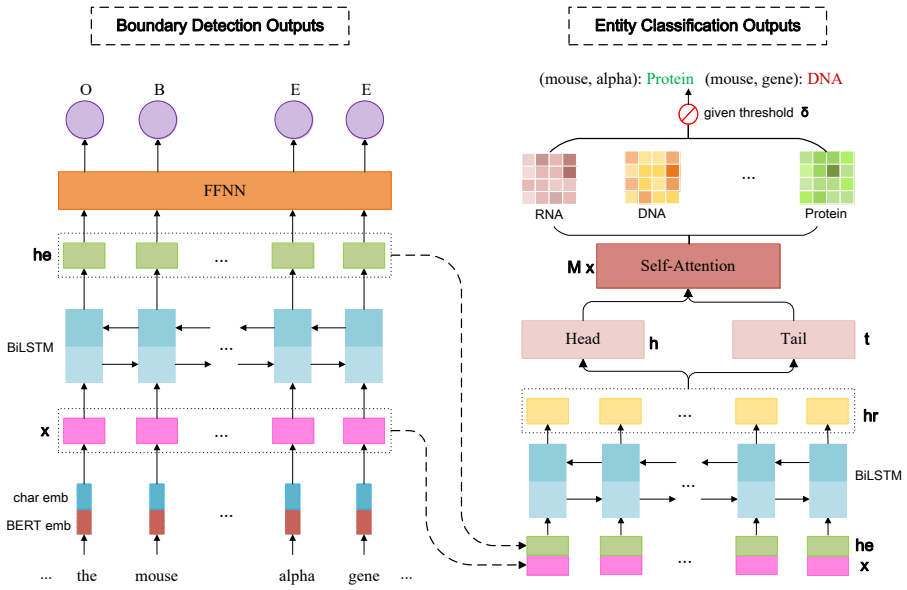

Figure 2: The whole framework mainly consists of two task modules: entity Boundary Recognition module to recognize the boundary of entities by word-level multi-class classification method and Entity Classification module to classify the entity spans into predefined types by word-pair-level multi-class classification method.

they joint two subtasks by sharing parameters in multi-task framework, they essentially extract boundaries and types separately and independently, generating error propagation and redundant information.

Different from the aforementioned works, we consider the correlations between words based on a supervised multi-head self-attention mechanism to learn the correlation intensities of the head-tail pair of candidate spans for each entity type. Simultaneously, we also employ a multi-task learning framework to joint entity boundary detection and entity classification, which can further improve the performance. In addition, with the successful application of pre-trained language models to various tasks, many researchers incorporate pre-trained contextual embeddings into their NER models. Such as (Wadden et al. 2019) exploits BERT (Devlin et al. 2019), (Luan et al. 2019) uses ELMo (Peters et al. 2018) and (Fisher and Vlachos 2019; Straková, Straka, and Hajic 2019) utilize both. In our model, we also apply the pre-trained language model BERT to learn more expressive word embeddings.

Model

We suppose the input is a sequence $S = (w_1, w_2, ..., w_N)$ with N words. The entity boundary detection module aims to classify each word into four classes belong to a set $L_b = \{B, I, E, O\}$, where B, I, E and O represent Beginning, Inside, Ending and Outside, respectively. We use a notation $s = (w_i, w_j)\ i, j \in [1, N]$ to represent the head-tail pair of a candidate span indicates its region from i-th word to j-th word in a sentence. The entity classification module aims to classify all possible head-tail pairs into multiple classes in a predefined set $R = \{r_1, r_2, ..., r_M\}$ with M entity types. Next, we will detail each component of our model.

Word Embedding Module

In this module, each word is represented by concatenating two type embeddings: a pre-trained word embedding $\boldsymbol{e}_{w_i}^{(w)}$, and a character-based word embedding $\boldsymbol{e}_{w_i}^{(c)}$. We use $\boldsymbol{x}_i$ to represent the i-th word w_i, as follows.

$$\boldsymbol{x}_i = [\boldsymbol{e}_{w_i}^{(w)}; \boldsymbol{e}_{w_i}^{(c)}], i \in [1, N] \quad (1)$$

Here, we use the pre-trained language model BERT and a convolutional neural network to obtain $\boldsymbol{e}_{w_i}^{(w)} \in \mathbb{R}^{d_w}$ and $\boldsymbol{e}_{w_i}^{(c)} \in \mathbb{R}^{d_c}$, respectively. Note that, we replace the BERT pre-trained embeddings with Glove word embeddimgs (Pennington, Socher, and Manning 2014) in our model without pre-trained language models.

Boundary Detection (BD) Module

This module mainly contains two layers: a BiLSTM encoder layer and a Feed Forward Neural Network (FFNN) classification layer. First, BiLSTM encoder takes the sequence of embeddings $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N)$ as input to capture the dependencies of words. And the contextualized word representation sequence $\boldsymbol{h_e} = (\boldsymbol{he}_1, \boldsymbol{he}_2, ..., \boldsymbol{he}_N)$ is obtained by concatenating the forward and backward LSTM hidden state, as follows:

$$\boldsymbol{he}_i = [\overrightarrow{LSTM}(\boldsymbol{x}_i, \overrightarrow{\boldsymbol{he}}_{i-1}); \overleftarrow{LSTM}(\boldsymbol{x}_i, \overleftarrow{\boldsymbol{he}}_{i+1})] \quad (2)$$

where $\boldsymbol{he}_i \in \mathbb{R}^{2d_{he}}$ and d_{he} denotes the dimension of the LSTM hidden state.

Then, H_e is fed into FFNN layer for modeling word classification. For detecting the boundaries of entities, we use the BIEO (Beginning, Inside, Ending, Outside) as tagging scheme. We suppose $\phi(l, w_i)$ measures how likely the i-th

word w_i has a boundary label $l \in L_b$, $P(w_i = l|S)$ indicates the probability of a w_i with boundary l over all possible boundary labels for the input sequence S. Let $\boldsymbol{he}_i$ be the contextual word embedding for word w_i, the $\phi(l, w_i)$ and $P(w_i = l|S)$ are computed as follows:

$$\begin{aligned} \phi(l, w_i) &= \boldsymbol{W_x} \cdot FFNN(\boldsymbol{he}_i) \\ p(w_i = l|S) &= \frac{exp(\phi(l, w_i))}{\sum_{l' \in L_b} exp(\phi(l', w_i))} \end{aligned} \tag{3}$$

where $\boldsymbol{W_x} \in \mathbb{R}^{d_f \times 4}$ is a neural network parameters to be learned and d_f is the width of FFNN layer. During training, we maximize the likelihood probability and choose the negative log-likelihood loss function as the loss $\mathcal{L}_{bd}$:

$$\mathcal{L}_{bd} = -\sum_{w_i \in S} \sum_{l \in L_b} \hat{p}(w_i = l|S) log p(w_i = l|S) \tag{4}$$

where $\hat{p}(w_i = l|S)$ represents the gold distribution of word w_i with label l.

Note, while CRF is considered good at modeling sequence labeling, choosing CRF as the output layer of BD module doesn't work very well because our sequence labels are different from flat NER models (e.g. the label sequence *BIIBIE* is also valid). And, the CRF-based model is time-consuming, about 2-4 times slower than the softmax-based model in inference speed.

Entity Classification (EC) Module

We assume the input of this module as $\boldsymbol{e} = (\boldsymbol{e}_1, \boldsymbol{e}_2, ..., \boldsymbol{e}_N)$. For each $\boldsymbol{e}_i$, it concatenates two component: the contextual representation $\boldsymbol{he}_i$ generated by the BiLSTM encoder in BD Module and the word embeddings $\boldsymbol{x}_i$ from the Embedding Module:

$$\boldsymbol{e}_i = [\boldsymbol{x}_i; \boldsymbol{he}_i], i \in [1, N] \tag{5}$$

where $\boldsymbol{e}_i \in \mathbb{R}^{d_w + d_c + 2d_{he}}$.

By sharing the input word embeddings and the contextual representations, the interaction and hierarchical semantic structure between two tasks can be captured. Then we run another BiLSTM encoder which takes $\boldsymbol{e}$ as input to learn another the contextual word representation sequence $\boldsymbol{hr} = (\boldsymbol{hr}_1, \boldsymbol{hr}_2, ..., \boldsymbol{hr}_N)$ for entity type classification:

$$\boldsymbol{hr}_i = [\overrightarrow{LSTM}(\boldsymbol{e}_i, \overrightarrow{\boldsymbol{hr}}_{i-1}); \overleftarrow{LSTM}(\boldsymbol{e}_i, \overleftarrow{\boldsymbol{hr}}_{i+1})] \tag{6}$$

where $\boldsymbol{hr}_i \in \mathbb{R}^{2d_{hr}}$, and d_{hr} is the hidden size of the LSTM in EC Module.

To represent the head and tail of entity spans, we first apply two single-layer linear fully connection layer to map each hr_i into head and tail vector representation space which correspond to whether the i-th word serves as the head or tail of entities, as follows.

$$\begin{aligned} \boldsymbol{h}(i) &= \boldsymbol{W_h hr}_i + \boldsymbol{b_h} \\ \boldsymbol{t}(i) &= \boldsymbol{W_t hr}_i + \boldsymbol{b_t} \\ i &\in [1, N] \end{aligned} \tag{7}$$

where $W_h \in \mathbb{R}^{l \times 2d_{hr}}$, $W_t \in \mathbb{R}^{l \times 2d_{hr}}$, $b_h \in \mathbb{R}^l$, $b_t \in \mathbb{R}^l$ and l is the width of the linear full connection layer.

To learn the correlation intensities of pairwise words for each entity type. We employ a multi-head self-attention mechanism which applies self-attention multiple times over the same inputs using separate attention heads, which can focus on different relevant words for each entity type. Here, we set the number of heads as the size of the entity types and utilize additive attention operation. We denote $s_k(i, j)$ as the correlation intensities of the head-tail pair (w_i, w_j) under the k-th entity type:

$$\begin{aligned} &s_k(i, j) = \boldsymbol{w}_k(tanh(\boldsymbol{h}(i) + \boldsymbol{t}(j))) \\ &i, j \in [1, N];\ k \in [1, M] \end{aligned} \tag{8}$$

where $\boldsymbol{w}_k$ is a vector of parameters with size l. For simplicity of computation, we extend the $\boldsymbol{w}_k$ into a matrix with size $M \times l$, note it as $\boldsymbol{W_r}$, the sequence $(\boldsymbol{h}(1), \boldsymbol{h}(2), ..., \boldsymbol{h}(N))$ as $\boldsymbol{h}$, and the sequence $(\boldsymbol{t}(1), \boldsymbol{t}(2), ..., \boldsymbol{t}(N))$ as $\boldsymbol{t}$. The correlation score tensor is computed as follows:

$$\boldsymbol{S} = \boldsymbol{W_r}(tanh(\boldsymbol{h} + \boldsymbol{t})) \tag{9}$$

where $\boldsymbol{S}$ is a $M \times N \times N$ tensor.

We guide this attention module to predict entity types by introducing a supervise information $r_k^{(i,j)}$. Given the word pair (w_i, w_j), we compute the probability of $r_k^{(i,j)} = 1$ and $r_k^{(i,j)} = 0$, as follows:

$$p(r_k|w_i, w_j) = \frac{exp(s_k(i, j))}{\sum_{k'=1}^{k'=M} exp(s'_k(i, j))} \tag{10}$$

$$p(r_k^{(i,j)}|w_i, w_j) = \begin{cases} p(r_k|w_i, w_j), r_k^{(i,j)} = 1 \\ 1 - p(r_k|w_i, w_j), r_k^{(i,j)} = 0 \end{cases} \tag{11}$$

where $p(r_k|w_i, w_j)$ indicates the probability of type r_k to be predicted as the entity type for word pair (w_i, w_j) over all possible entity types, $r_k^{(i,j)} = 1$ denotes the fact that the word pair (w_i, w_j) have entity type r_k, and vice versa. During training, we convert the gold annotation to a one-hot matrix and select the binary cross-entropy loss as the loss $\mathcal{L}_{ec}$:

$$\begin{aligned} \mathcal{L}_{ec} = -\sum_{k=1}^{M} \sum_{i=1}^{N} \sum_{j=1}^{N} [&\hat{p}(r_k^{(i,j)}|w_i, w_j) log p(r_k^{(i,j)}|w_i, w_j) + \\ &(1 - \hat{p}(r_k^{(i,j)}|w_i, w_j)) log(1 - p(r_k^{(i,j)}|w_i, w_j))] \end{aligned} \tag{12}$$

where $\hat{p}(r_k^{(i,j)}|w_i, w_j)$ represents the gold distribution of word pair (w_i, w_j) with relation r_k.

Joint Learning

To train entity boundary detection and entity classification simultaneously, we employ a multitask loss which a sum of loss of two tasks, as follows:

$$\mathcal{L}_{loss} = \mathcal{L}_{bd} + \mathcal{L}_{ec} \tag{13}$$

During the inference, given the head-tail pair (w_i, w_j) of a candidate span, the r_k is predicted as the entity type for it if $p(r_k|w_i, w_j)$ exceeds a given threshold δ.

Experiments

In this section, we conducted comprehensive experiments on five datasets among which three are nested and two are flat. And the results demonstrate that our proposed method outperforms all baselines on both nested NER and flat NER.

Datasets and Metrics

We performed nested NER task on the widely-used ACE2004, ACE2005 and GENIA datasets and performed flat NER task on JNLPBA and CoNLL03-English datasets. The details of data statistics are summarized in Table 1.

ACE2004[2] **and ACE2005**[3] (Doddington et al. 2004) are two English corpus and each contains 7 entity types as "PER", "ORG", "LOC", "GEP", "VEH", "WEA" and "FAC". We reuse the same train/dev/test splits following the previous works (Lu and Roth 2015; Wang and Lu 2018).

GENIA[4] is a nested dataset in biomedicine domain and is constructed based on GENIA v3.0.2 corpus. We processed the dataset following (Finkel and Manning 2009; Lu and Roth 2015) by keeping only 5 entity categories as "DNA", "RNA", "Protein", "Cell-type", and "Cell-line". We split dataset into training, development and testing with the ratio 8.1:0.9:1.

JNLPBA[5] is a corpus for flat NER and contains training and testing set. This corpus is originally from GENIA but only flat and topmost entities are kept. Like GENIA corpus, we only kept the 5 entity types. Following (Ju, Miwa, and Ananiadou 2018), we randomly select 10% sentences from training set as our development set.

CoNLL03-English (Sang and Meulder 2003) is a flat NER dataset in the domain of news and contains 4 entity categories: "PER", "LOC", "ORG" and "MISC". We split dataset into training, development and testing with the ratio 8:1:1.

Metrics In all cases, we use the training set to train our model and use the development set to tune the hyperparameters of our model. We exploit the same evaluation metrics used by previous works: a predicted entity is considered correct only when its span and type match with the gold entity. For comparison, we reported precision (P), recall (R) and micro F1 (F1) scores of our best performing models.

Implementation Details

We use AllenNLP which is a friendly NLP framework to implement our model. For word-level word embeddings, while exploiting pre-trianed language model, we apply SciBERT$_{base}$ on GENIA and JNLPBA datasets and apply BERT$_{base}$ on other datasets and their detailed model size are referred to (Beltagy, Lo, and Cohan 2019) and (Devlin et al. 2019), respectively; while without pre-trained language models, we use the pretrained embeddings trained on MEDLINE abstract (Chiu et al. 2016) for GENIA and JNLPBA datasets and use Glove embeddings with 300 dimensions for other datasets. For character-level word embeddings, we apply 32 filters with window [3,4,5] on a CNN layer. For BiLSTM encoder in both BD and EC module, the dimension of the hidden state is 200. Each linear full connection layer has one layer with 150-dimensions. To avoid over fitting, we apply 0.5 dropout for the input embeddings and 0.4 dropout for BiLSTM encoder. During training, we use Adam optimizer with the learning rates of 5.0×10^{-5} and perform linear decay of the learning rate. For hyperparameters, we select δ as 0.5 with the best developing results.

Baselines

We compare our model with several state-of-the-art models on both nested NER and flat NER tasks, including sequence labeling based and span-based models .

For **sequence labeling-based models**, (Lample et al. 2016) is a classical neural model for flat NER, which utilize a LSTM-CRF neural network. (Ju, Miwa, and Ananiadou 2018) is a CRF-based model which dynamically stack flat NER layers for nested NER based on a deep neural network. (Lu and Roth 2015; Katiyar and Cardie 2018; Wang and Lu 2018; Wang et al. 2018) are transformation-based nested NER models, where (Lu and Roth 2015; Katiyar and Cardie 2018; Wang and Lu 2018) use hyper-graph transformation while (Wang et al. 2018) use shift-reduce transformation. (Straková, Straka, and Hajic 2019) propose two models for nested NER task, where one is a multi-label classification model based on LSTM-CRF, another is a seq2seq model, meanwhile, they exploit pre-trained language models to enhance the performance. (Fisher and Vlachos 2019) present a merge and label approach for nested NER and apply both BERT and EMLo contextual embeddings respectively.

For **span-based models**, (Xia et al. 2019) and (Sohrab and Miwa 2018) utilize spans enumeration and span classification for both nested and flat NER. But (Xia et al. 2019) employ a pre-trained language model ELMo to improve the performance. (Luan et al. 2019) is also based on span classification and they apply a multi-task method to joint entities, relations and coreference links extraction for further improvement. Recently, (Tan et al. 2020) also introduce a muti-task framework for extract nested entities, but they joint two subtasks (span boundaries detection and span type classification) of NER instead of different application tasks.

Nested NER Task

We evaluate our model on three nested datasets and the results are presented in Tabel 2. The results demonstrate our model can effectively improve the performance in nested NER task. Compared with the state-of-the-art baselines without pre-trained language models, our model achieves best results in two out of three corpora. We can see that our model is slightly worse than (Straková, Straka, and Hajic 2019)(seq2seq), but the main reason is that (Straková, Straka, and Hajic 2019)(seq2seq) needs more complex network structure and dependents on extra features such as word lemmas and POS tags, while our model is simpler and without any extra features and annotations.

[2]https://catalog.ldc.upenn.edu/ LDC2005T09

[3]3https://catalog.ldc.upenn.edu/ LDC2006T06

[4]http://www.geniaproject.org/genia-corpus/term-corpus

[5]http://www.nactem.ac.uk/tsujii/GENIA/ ERtask/report.html

Statistics	ACE2004	ACE2005	GENIA	JNLPBA	CoNLL03-English
#Sentences	8488	9311	18546	22402	22131
Split ratio	8:1:1	8:1:1	81:9:10	81:9:10	8:1:1
#Entities	27747	30944	56870	60158	34920
Nested entities	45.7%	39.8%	21.6%	0%	0%
Ave. entity length	2.64	2.21	2.9	2.14	1.45
#Ave. entity per sentence	3.27	3.32	3.07	2.69	1.58

Table 1: The statistical comparison between the datasets.

Model	ACE2004			ACE2005			GENIA		
	P	R	F1	P	R	F1	P	R	F1
(Lu and Roth 2015)	70.0	56.9	62.8	66.3	59.2	62.5	72.5	65.2	68.7
(Ju, Miwa, and Ananiadou 2018)	-	-	-	74.2	70.3	72.2	78.5	71.3	74.7
(Katiyar and Cardie 2018)	73.6	71.8	72.7	70.6	70.4	70.5	79.8	68.2	73.6
(Wang and Lu 2018)	78.0	72.4	75.1	76.8	72.3	74.5	77.0	73.3	75.1
(Sohrab and Miwa 2018)	77.8*	70.8*	74.1*	77.2*	70.0*	73.4*	93.2	64.0	77.1
(Straková, Straka, and Hajic 2019)(LSTM-CRF)	-	-	72.3	-	-	71.6	-	-	76.2
(Straková, Straka, and Hajic 2019)(Seq2seq)	-	-	**77.8**	-	-	75.4	-	-	76.4
(Tan et al. 2020)	78.1	72.8	75.3	77.1	74.2	75.6	78.9	72.7	75.7
MHSA(ours)	79.5	74.8	77.1	80.1	73.3	**76.5**	77.9	76.7	**77.3**
with Pre-trained Language Model									
(Xia et al. 2019)(ELMo)	81.7	77.4	79.5	79.0	77.3	78.2	-	-	-
(Luan et al. 2019)(ELMo)	-	-	84.7	-	-	82.9	-	-	76.2
(Straková, Straka, and Hajic 2019)(seq2seq-BERT)	-	-	84.3	-	-	83.4	-	-	78.2
(Tan et al. 2020)(BERT)	85.8	84.8	85.3	83.8	83.9	83.9	79.2	77.4	78.3
MHSA-BERT (ours)	86.9	85.8	**86.3**	85.7	85.2	**85.4**	80.3	78.9	**79.6**

Table 2: Nested NER results for ACE2004, ACE2005 and GENIA datasets, and the scores marked with $*$ are reported using our self-implemented results.

Compared with state-of-art the models using contextual embeddings, we can see that our model outperforms the best baseline (Tan et al. 2020) by 1.0%, 1.5% and 1.3% absolute F1 scores on ACE2004, ACE2005 and GENIA datasets, respectively. We think that although (Tan et al. 2020) utilize multi-task model for joint learning, they still suffer from error cascading between tasks because of working in pipeline manner. Another, we can observe that there are substantial increases in all performance when applying pre-trained language models, which indicates that learning the expressive word representations is important for named entity recognition.

In addition, we can also see that the span-based baseline (Sohrab and Miwa 2018) perform on GENIA dataset with worst recall. We attribute the reason into two folds. First, the length of spans would limit to generate all possible entity spans. Second, lacking of explicit boundary information leading to poor performance on span detection. Another, we can observe that (Lu and Roth 2015) has the worst performance on three nested datasets, caused by the spurious structure of hyper-graphs designed for their model.

Flat NER Task

We also compare our model with state-of-the-art models on flat NER task to verify our model also preform well in identifying flat entities. Table 3 shows the performance of our model on two flat datasets JNLPBA and CoNLL03-English. We can see that our MHSA-BERT model outperforms the best baseline model (Tan et al. 2020)(BERT) by 1.1% and 1.7% on JNLPBA and CoNLL03-English, respectively. Further more, by comparing with the baselines without pre-trained language models, our MHSA model also obtains comparable results on both datasets. We think the main reason is that our model is a united type-aware model which maps each pairwise words as points in multiple entity type spaces, without external dependencies and error cascading.

Impact of Model Components

Tabel 4 presents the results of an ablation experiment on GENIA testing set showing each component of our model have various degrees of contributes to the effectiveness of our model. We can see that ablation on pre-trained language model BERT significantly decrease the F1 scores by 2.3 percentage points. Moreover, the performance is also impaired large when replace the SciBERT$_{base}$ with BERT$_{base}$, which indicates that pre-training the model on the corpus in similar domains with training data can significantly improve the performance. In addition, adding entity boundaries detection module and character-level embeddings can also improve the performance of our model. Because the character-level embeedings has been contained in BERT, so ablation on it has small decrease on F1 scores.

Entity Boundary Detection

We also conduct experiments on boundary detection task to illustrate that our model can extract decent entity boundaries by multi-task framework, as shown in Table 5. (Sohrab and Miwa 2018) and Only BD Module do not make full use

Model	JNLPBA	CoNLL03-English
	F1	F1
(Lample et al. 2016)	-	90.9
(Straková, Straka, and Hajic 2019)(LSTM-CRF)	-	90.7
(Straková, Straka, and Hajic 2019)(Seq2seq)	-	90.8
(Straková, Straka, and Hajic 2019)(seq2seq-BERT)	-	93.0
(Xia et al. 2019)(ELMo)	-	92.3
(Ju, Miwa, and Ananiadou 2018)	75.6	-
(Sohrab and Miwa 2018)	78.4	-
(Tan et al. 2020)	73.2*	90.7*
(Tan et al. 2020)(BERT)	78.6*	91.9*
MHSA(ours)	78.3	91.0
MHSA-BERT (ours)	**79.7**	**93.6**

Table 3: flat NER results on JNLPBA and CoNLL03 datasets, and the scores marked with $*$ are reported using our self-implemented results.

Model	GENIA		
	P	R	F1
HMTIE-BERT(sciber_base)	80.3	78.9	79.6
HMTIE-BERT(base)	78.1	77.6	77.8(↓1.8)
-BD	79.4	77.8	78.6(↓1.0)
-BERT	77.9	76.7	77.3(↓2.3)
-character-level embeddings	79.8	78.5	79.1 (↓0.5)

Table 4: Ablation test on GENIA testing set.

Model	ACE2004	ACE2005	GENIA
(Sohrab and Miwa 2018)	81.5	80.3	72.7
(Tan et al. 2020)(BERT)	83.9	83.0	78.3
Only BD Module	83.2	82.6	77.8
MHSA-BERT (ours)	**84.7**	**84.1**	**79.1**

Table 5: Performance of entity boundary detection on ACE2004, ACE2005 and GENIA testing set.

of the dependencies between boundary detection and type classification, so they have worse performance than other two models. Compared with (Tan et al. 2020)(BERT), our model obtains 0.8%, 1.1% and 0.8% absolute F1 gains on ACE2004, ACE2005 and GENIA, respectively. We think it is because that our model not only consider the dependencies between entity boundary detection and entity type classification, but also consider the correlation between words when classifying entity types. In addition, because the entities contained in GENIA are longer and more complex than those contained in ACE2004 and ACE2005, the models perform worse on GENIA.

Visualization

Each slice of tensor S is a correlation score matrix $S_i, i \in [1, M]$ which indicates the correlation intensities of each pairwise words in sentence under i-th entity type. For each correlation score matrix S_i, we sum up over all the column vectors to get a weight vector and then normalizing. So we can get M normalized weight vectors that yields a general view of what words the M entity types mostly focus on respectively. For interpreting the correlation between words in sentence under different entity types, we draw a heap map of

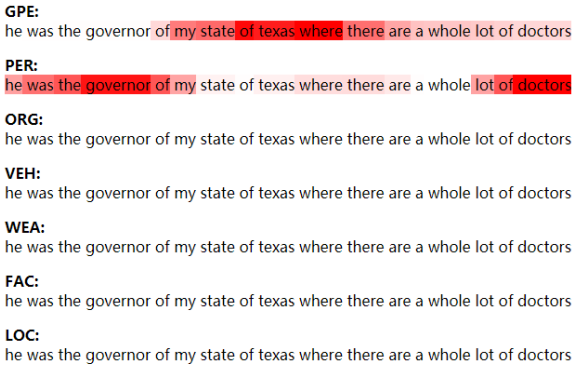

Figure 3: Attention of entity types on words in an example sentence randomly selected from ACE2005. The darker the color, the greater the attention weight.

M normalized weight vectors for an example sentence randomly selected from ACE2005 corpus, as shown in Figure 3. From the figure, we can see that "GEP" and "PER" entity types can focus on relevant words properly and other entity types have no words to focus on due to no entities with these types. We can conclude that the correlations between words for each entity type leaned by our model are meaningful and in accord with common sense.

Conclusion and Future Work

In this paper, we address a task of recognizing entities with nested structures. We consider words in the sentence play different roles under different entity types and treat overlapping entities extraction as a multi-class classification of pairwise words. For this end, we design a simple but effective model based on supervised multi-head self-attention mechanism. Simultaneously, we introduce a multi-task framework to capture the dependencies between entity boundary detection and type classification, generating further improvement.

For future work, we want to introduce a type embedding to explicitly model the semantic relevance between words and types.

Acknowledgments

This work was supported by in part by the National Key Research and Development Program of China under Grant No.2016YFB0801003.

References

Alex, B.; Haddow, B.; and Grover, C. 2007. Recognising Nested Named Entities in Biomedical Text. In *BioNLP@ACL*, 65–72.

Beltagy, I.; Lo, K.; and Cohan, A. 2019. SciBERT: A Pretrained Language Model for Scientific Text. In *EMNLP/IJCNLP*, 3615–3620.

Byrne, K. 2007. Nested Named Entity Recognition in Historical Archive Text. *International Conference on Semantic Computing (ICSC 2007)* 589–596.

Chang, K.-W.; Samdani, R.; and Roth, D. 2013. A Constrained Latent Variable Model for Coreference Resolution. In *EMNLP*, 601–612.

Chiu, B.; Crichton, G. K. O.; Korhonen, A.; and Pyysalo, S. 2016. How to Train good Word Embeddings for Biomedical NLP. In *BioNLP@ACL*, 166–174.

Chiu, J. P. C.; and Nichols, E. 2016. Named Entity Recognition with Bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics* 4: 357–370.

Clark, K.; Luong, M.-T.; Manning, C. D.; and Le, Q. V. 2018. Semi-Supervised Sequence Modeling with Cross-View Training. In *EMNLP*, 1914–1925.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural Language Processing (Almost) from Scratch. *Journal of machine learning research* 12: 2493–2537.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*, 4171–4186.

Doddington, G.; Mitchell, A.; Przybocki, M. A.; Ramshaw, L.; Strassel, S.; and Weischedel, R. 2004. The Automatic Content Extraction (ACE) Program - Tasks, Data, and Evaluation. In *LREC*.

Finkel, J. R.; and Manning, C. D. 2009. Nested Named Entity Recognition. In *EMNLP*, 141–150.

Fisher, J.; and Vlachos, A. 2019. Merge and Label: A novel neural network architecture for nested NER. In *ACL*, 5840–5850.

Fragkou, P. 2017. Applying named entity recognition and co-reference resolution for segmenting English texts. *Progress in Artificial Intelligence* 6: 325–346.

Gupta, N.; Singh, S.; and Roth, D. 2017. Entity Linking via Joint Encoding of Types, Descriptions, and Context. In *EMNLP*, 2681–2690.

Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *ArXiv* abs/1508.01991.

Ju, M.; Miwa, M.; and Ananiadou, S. 2018. A Neural Layered Model for Nested Named Entity Recognition. In *NAACL-HLT*, 1446–1459.

Katiyar, A.; and Cardie, C. 2018. Nested Named Entity Recognition Revisited. In *NAACL-HLT*, 861–871.

Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 260–270.

Liu, L.; Ren, X.; Zhu, Q.; Zhi, S.; Gui, H.; Ji, H.; and Han, J. 2017. Heterogeneous Supervision for Relation Extraction: A Representation Learning Approach. In *EMNLP*, 46–56.

Liu, Y.; Meng, F.; Zhang, J.; Xu, J.; Chen, Y.; and Zhou, J. 2019. GCDT: A Global Context Enhanced Deep Transition Architecture for Sequence Labeling. In *ACL*, 2431–2441.

Lu, W.; and Roth, D. 2015. Joint Mention Extraction and Classification with Mention Hypergraphs. In *EMNLP*, 857–867.

Luan, Y.; Wadden, D.; He, L.; Shah, A.; Ostendorf, M.; and Hajishirzi, H. 2019. A General Framework for Information Extraction using Dynamic Span Graphs. In *NAACL-HLT*, 3036–3046.

Ma, X.; and Hovy, E. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 1064–1074.

Martins, P. H.; Marinho, Z.; and Martins, A. F. T. 2019. Joint Learning of Named Entity Recognition and Entity Linking. In *ACL*, 190–196.

Miwa, M.; and Bansal, M. 2016. End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. In *ACL*, 1105–1116.

Muis, A. O.; and Lu, W. 2017. Labeling Gaps Between Words: Recognizing Overlapping Mentions with Mention Separators. In *EMNLP*, 2608–2618.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*, 1532–1543.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. In *ACL*, 2227–2237.

Ratinov, L.-A.; and Roth, D. 2009. Design Challenges and Misconceptions in Named Entity Recognition. In *CoNLL*, 147–155.

Sang, E. T. K.; and Meulder, F. D. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *CONLIG*.

Sohrab, M. G.; and Miwa, M. 2018. Deep Exhaustive Model for Nested Named Entity Recognition. In *EMNLP*, 2843–2849.

Straková, J.; Straka, M.; and Hajic, J. 2019. Neural Architectures for Nested NER through Linearization. In *ACL*, 5326–5331.

Tan, C.; Qiu, W.; Chen, M.; Wang, R.; and Huang, F. 2020. Boundary Enhanced Neural Span Classification for Nested Named Entity Recognition. In *AAAI*.

Villodre, L. M.; Villarejo, L.; Martí, M.; and Taulé, M. 2007. SemEval-2007 Task 09: Multilevel Semantic Annotation of Catalan and Spanish. In *SemEval@ACL*, 42–47.

Wadden, D.; Wennberg, U.; Luan, Y.; and Hajishirzi, H. 2019. Entity, Relation, and Event Extraction with Contextualized Span Representations. In *EMNLP/IJCNLP*, 5784–5789.

Wang, B.; and Lu, W. 2018. Neural Segmental Hypergraphs for Overlapping Mention Recognition. In *EMNLP*, 204–214.

Wang, B.; Lu, W.; Wang, Y.; and Jin, H. 2018. A Neural Transition-based Model for Nested Mention Recognition. In *EMNLP*, 1011–1017.

Wang, F.; Wu, W.; Li, Z.; and Zhou, M. 2017. Named entity disambiguation for questions in community question answering. *Knowl. Based Syst.* 126: 68–77.

Wang, Y. 2009. Annotating and Recognising Named Entities in Clinical Notes. In *ACL/IJCNLP*, 18–26.

Xia, C.; Zhang, C.; Yang, T.; Li, Y.; Du, N.; Wu, X.; Fan, W.; Ma, F.; and Yu, P. S. 2019. Multi-Grained Named Entity Recognition. In *ACL*, 1430–1440.

Xin, Y.; Hart, E.; Mahajan, V.; and Ruvini, J.-D. 2018. Learning Better Internal Structure of Words for Sequence Labeling. In *ACL*, 2584–2593.

Zheng, C.; Cai, Y.; Xu, J.; fung Leung, H.; and Xu, G. 2019. A Boundary-aware Neural Model for Nested Named Entity Recognition. In *EMNLP/IJCNLP*, 357–366.

Zhou, G.; and Su, J. 2002. Named Entity Recognition using an HMM-based Chunk Tagger. In *ACL*, 473–480.