



# Joint Learning of Token Context and Span Feature for Span-Based Nested NER

Lin Sun , Member, IEEE, Yuxuan Sun , Fule Ji, and Chi Wang 

**Abstract**—Nested named entity recognition (NER) is a linguistic phenomenon that has received increasing attention in the NLP community. In this article, we propose a novel joint learning network of the token context, and span feature (TCSF) for nested NER. Our model is a combination of token context network (TCN) for token learning, and deep residual convolutional neural network (CNN), and span relation network (SRN) for span learning. Span features are represented at both the token level, and span relation level. The span relation representation of SRN is trained on the similarity of span, and its positional features by attention weights. The *IoU* metric is employed for span filtering, and analysis of the overlapping adjacent spans. Moreover, we propose a novel head-inner-tail ( $HIT$ ) operation to extend the inner features in spans for a fixed-length representation. TCSF is a fully end-to-end entity recognition model. We perform a set of comprehensive experiments, including an ablation study on the ACE, and GENIA datasets, and show state-of-the-art performance without, and with the pretrained language models.

**Index Terms**—Contextual network, deep residual CNN, nested named entity recognition, span representation.

## I. INTRODUCTION

NAMED entity recognition (NER) is a fundamental task in natural language processing (NLP). It is the first step for higher-level NLP tasks, such as coreference resolution, relation extraction, and question answering. In recent years, nested NER has become an active topic in the information extraction (IE) field [1]–[4]. In particular, nested entity structures frequently exist in broadcast transcripts [5] and biomedical documents [6]. Nested named entities account for 17% of all entities in the GENIA corpus, and the sentences containing nested named entities account for 30% of the ACE corpora. Nested NER can be used to capture fine-grained semantic information. For example, “University of Washington” is an organization with a nested location. Nested NER can help us to recognize that the University of Washington is located in Washington. Although

Manuscript received March 5, 2020; revised June 18, 2020 and August 5, 2020; accepted August 18, 2020. Date of publication September 18, 2020; date of current version October 14, 2020. This work was supported in part by the National Innovation and Entrepreneurship Training Program for College Students under Grant 202013021005 and in part by the Natural Science Foundation of Zhejiang Province under Grant LY17F020008. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Taro Watanabe. (Corresponding author: Lin Sun.)

Lin Sun, Yuxuan Sun, and Fule Ji are with the Department of Computer Science, Zhejiang University City College, 310015 Hangzhou, Zhejiang, China (e-mail: sunl@zucc.edu.cn; sunyuxuan866@163.com; jifule523@gmail.com).

Chi Wang is with the College of Computer Science and Technology, Zhejiang University, 310007 Hangzhou, Zhejiang, China (e-mail: cwangup@zju.edu.cn). Digital Object Identifier 10.1109/TASLP.2020.3024944

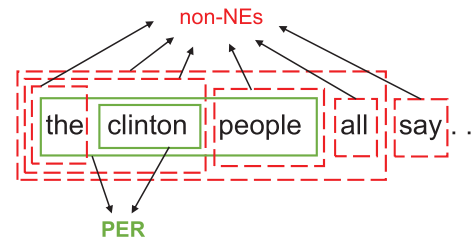


Fig. 1. An illustrative example from the ACE05 dataset for span-based nested NER. The green solid spans are classified as NEs of type Person (PER) and the red dashed spans are classified as non-NEs. The red dashed spans are a portion of non-NEs.

flat NER has achieved excellent performance in terms of accuracy [7], improving the performance of nested NER remains a great challenge [8].

Previous nested NER methods used parse tree [9], stacking flat layers [1], and hypergraphs [2], [10] to model the hierarchical structure of named entities (NEs). Recently, some researchers investigated the span-based approach [11], [12] for nested NER. The definition of span-based nested NER is to classify all candidate spans into either NE types or non-NE. An illustrative example for span-based nested NER is shown in Fig. 1. Compared with the stacked LSTM layer or hypergraph, the advantage of the span-based approach is that there is no need to specifically model the nested level between entities. However, there are still some issues that should be addressed. For example, usually all possible spans are used to be classified to detect mentions. Consequently, this approach may cause overlapping but not nested mentions. In addition, post-processing is needed, and therefore the network is not fully end-to-end. Second, the features of spans are always the concatenation of token embeddings. There is no feature learning for individual spans. Third, the overlapping adjacent spans, such as “ $u_1u_2u_3$ ” and “ $u_1u_2$ ,” or “ $u_1u_2u_3$ ” and “ $u_2u_3u_4$ ,” share the same words, resulting in the representations of these spans are relatively close in the feature space and increasing difficulty in span classification.

In this paper, we propose a novel span-based model for nested NER and perform a comprehensive evaluation. Our model mainly focuses on the improvement of span feature extraction, span representation, and span classification. It is a fully end-to-end entity detection model and no further post-processing is required. The results show that our model achieves state-of-the-art performance not only without any extra resource but also with pretrained language models (LMs).

The contributions of this paper can be summarized as follows:

- We propose a joint network for token context and span feature learning. Span features are represented at both the token level and span relation level. The results show the importance of span feature learning through ablation study and visualization of the feature space.
- To improve the performance of span classification, we design a span relation network (SRN) with a positional feature. SRN makes the span features more discriminative, especially for the overlapping adjacent spans.
- We propose a novel  $HI^pT$  operation for converting spans of various lengths to fixed-length vectors. This operation can extract scalable inner features in addition to boundary information. The results show that the extended information within spans is beneficial for performance improvement.

## II. RELATED WORK

Several methods have been proposed for nested NER. In the early years, Finkel and Manning [9] proposed a constituency parser with constituents for each named entity in a sentence. The model employed handcrafted features that were local and pairwise. This method was time-consuming, and its complexity was cubic in the number of words in the sentence. Word-level classification by conditional random field (CRF) is popular for NER [13]. However, in nested NER, one word might belong to two different NEs. Therefore, researchers employed some techniques to solve the nested NER problem. Gu [14] built separate binary SVM classifiers for the innermost and outermost entities. Li *et al.* [15] used the BIOESD [16] encoding scheme and proposed a classification bi-LSTM network to decide whether a non-head component should be combined with its nearest head component or not. In Yaseen *et al.* [17], the outermost entities detected by a level-1 NER model were fed to a level-2 NER model to detect nested entities. It could only handle two-layer nested NEs. Ju *et al.* [1] and Nguyen *et al.* [18] designed stacking multiple layers of LSTM-CRF to detect nested NEs. A flat layer consisting of a bi-LSTM and CRF layers predicted the label sequences in the corresponding nested level. Lu and Roth [19] proposed a mention hypergraph model for recognizing overlapping mentions. Then, they incorporated mention separators to tag the gap between two words [20]. Recently, Wang and Lu [10] proposed the segmental hypergraph representation to model nested entity mentions with pretrained embeddings. Katiyar *et al.* [2] presented a standard LSTM-based sequence labeling model to learn the nested entity hypergraph structure for an input sentence.

Span- or region-based approaches were recently presented [11], [12], [21]. Sarawagi *et al.* [22] had earlier attempted at segment-based information extraction by semi-CRF. The best path for segmentation is searched using a lattice structure [23], [24]. In fact, one word in the segment on the best path could only belong to one of the NEs, which is the same as word-level CRF. Both “span” and “segment” are literally a word chunk. However, the span-based approach enumerates all possible spans and classifies the spans one by one into either NE types or non-NE. The strong advantage of the span-based approach

for nested NER is to naturally allow one word to belong to two or more NEs. One challenge of the span-based approach is partially overlapped mentions in the detection results. A post-processing method, such as non-maximum suppression, is sometimes needed to remove partially overlapped mentions. Xu *et al.* [11] used the highest-first and longest-first strategies. The authors checked every word in a sentence. If the word was contained by more than one mention, only the highest score or the longest length mention would be retained. The problem that such post-processing methods introduce is that an NE would undergo loss when its score is not the highest or its length is not the longest, suggesting that post-processing is not necessarily effective. In addition, to reduce the size of all possible spans, Lin *et al.* [25] proposed anchor-region networks (ARNs) for nested mention detection. ARNs identify head words of mentions and recognize mention regions by exploiting the regular phrase structure. Zheng *et al.* [26] proposed a boundary-aware neural model, which first detected the boundary of the entity region. Although region and boundary detection could decrease the number of spans to be classified, the accuracy of detecting the region or boundary, approximately 80%, would limit the upper bound of the overall performance. Sohrab and Miwa [12] and Sun *et al.* [21] employed bi-LSTM and self-attention to learn token features, respectively. They both used START, END, and an average of tokens in a span as span representation.

In 2019, pretrained LMs were used in nested NER. Luan *et al.* [27] proposed a multi-task learning framework with ELMo embeddings [28], called DyGIE, to perform named entity recognition, relation extraction, and coreference resolution through shared span representations. Later, DyGIE++ [29] used BERT [30] encodings and fed each sentence to BERT together with surrounding sentences to capture a wider cross-sentence context. Strakova *et al.* [3] viewed nested NER as a sequence-to-sequence problem, tagged the labels of a sequence using an LSTM decoder, and improved the performance using pretrained LMs such as BERT and Flair [31]. Fisher *et al.* [4] merged tokens into entities forming nested structures and labeled them independently. The authors also improved F1 score using BERT.

## III. THE PROPOSED MODEL

In this work, we propose a novel nested NER neural architecture of the token context and span feature (TCSF), as shown in Fig. 2. The TCSF model contains five major parts: 1) token context network (TCN), 2) deep residual CNN, 3) span relation network (SRN), 4)  $HI^pT$  operation, and 5) entity classification network (ECN). We input the sequence of token embeddings  $T$  into TCN, which is based on a self-attention mechanism, to generate a token representation sequence  $U$  under a global context. In our model, the spans are represented as the token-based level and span relation level. The token-based span is the concatenation of token features. We enumerate all token-based spans  $S$  from the contextual token sequence  $U$ . To extract the features of spans, we employ a deep CNN that allows shortcut connections to cross convolutional layers [32]. The deep CNN focuses on the local features within the span region. It has the following advantages: 1) The deep CNN can convert contextual

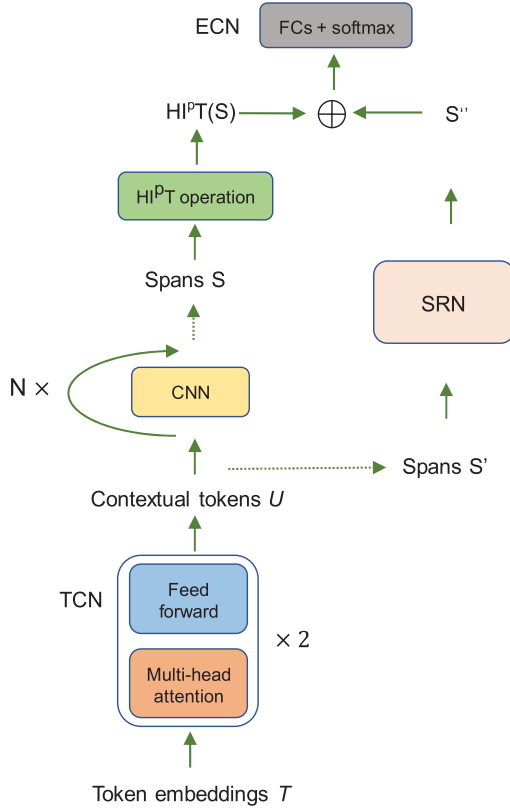


Fig. 2. Illustration of TCSF.

token representations to task-specific representations and extract more effective features of spans than one convolutional layer. 2) In the network implementation, all spans  $S$  can share feature maps and the computation of CNN in the forward and backward passes. The HIPT operation converts the spans  $S$  with various lengths into the fixed-length vectors HIPT( $S$ ) for connection to a fully connected classification network.

The flaw of the token-based span is that the appearances of the adjacent spans with a large overlap are similar. Span relation level representations  $S''$  are constructed by SRN to exploit relations between the spans from the set  $S'$ . The aim of SRN is to learn a discriminative feature transform for the spans with similar appearances and improve the performance of span classification. Finally, we concatenate ( $\oplus$ ) HIPT( $S$ ) with  $S''$  and feed them into ECN, which has two fully connected layers (FCs) for softmax classification. The softmax layer produces the probabilities over  $K$  categories plus non-NE, where  $K$  is the number of entity types. We show the details of the proposed model in the following sections.

#### A. Token Context Network

Contextualized token representation has demonstrated strong empirical performance [28], [33], [34]. We first review a basic self-attention module, called “Scaled Dot-Product Attention” [33]. The input matrices  $Q'$ ,  $K'$  and  $V'$  consist of queries and keys of dimension  $d_k$ , and values of dimension  $d_v$ . The scaled dot product is performed between the query in  $Q'$  and all keys in  $K'$  with a scaling factor of  $\frac{1}{\sqrt{d_k}}$  to obtain their similarities.

Then, a softmax function is applied to obtain the weights on the values. The output value of the attention is a weighted average over input values  $V'$ , and defined as follows:

$$Att(Q', K', V') = softmax\left(\frac{Q'K'^T}{\sqrt{d_k}}\right)V', \quad (1)$$

Single attention is extended to multi-head attention, which always performs better. Multi-head attention is the concatenation  $\oplus$  of single attentions, defined as follows [33],

$$MHA(Q, K, V) = head_1 \oplus \dots \oplus head_h, \quad (2)$$

where  $head_i = Att(QW_i^Q, KW_i^K, VW_i^V)$ ,  $W_i^Q \in \mathbb{R}^{d_{input} \times d_k}$ ,  $W_i^K \in \mathbb{R}^{d_{input} \times d_k}$ ,  $W_i^V \in \mathbb{R}^{d_{input} \times d_v}$ .

In this work, the TCN contains two stacked layers and each layer consists of multi-head attention and a feed-forward network, followed by layer normalization. Unlike the Transformer [33], we do not add any positional encoding because some evidence [35] showed that positional information could be implicitly encoded by deep CNN. Each token in the input sequence  $T$  is the concatenation of word embeddings [36] and character-level word embeddings. We use a bi-LSTM to construct character-level word embeddings as in [37]. In Eq. 2,  $Q = K = V = T$  and  $d_k = d_v = d_{input}/h$ . In addition, the feed-forward network has two linear transformations and a ReLU activation in between. The output of TCN has two branches: one is directly fed into deep residual CNN, and the other one is converted to a span set  $S'$  by averaging the contextual token representations in each span and fed into SRN.

Recently, pretrained LMs such as BERT obtained new state-of-the-art results for several major NLP tasks. When applying pretrained LMs in our model, we concatenate the outputs of deep residual CNN to that of pretrained LMs.

#### B. Deep Residual CNN

The deep residual CNN is based on the work of [32]. It consists of  $N$  CNN layers with identity shortcut connections that skip one layer. A CNN block is  $m$  convolutions of size  $k$  if the dimension of the token representation is  $m$  so that the input and output of a CNN layer have the same dimension. One residual CNN layer is defined as follows:

$$y = ReLU(x + Conv_{k,m}(x)), \quad (3)$$

where  $Conv_{k,m}$  is a convolutional network,  $ReLU$  is a rectified linear unit activation, and  $x$  is a token sequence.

#### C. Span Relation Network

In this section, we describe the computation of SRN, as shown in Fig. 3. Given an input set of  $M$  spans  $S' = \{s_n\} = \{(f_n, g_n)\}_{n=1}^M$ ,  $f = \frac{1}{j-i+1} \sum_{k=i}^j u_k$  is the average of the contextual token sequence  $\{u_k\}_{k=i}^j$  of a span and  $g$  is its positional feature. In a relation module, the relations  $r_n \in \mathbb{R}^{1 \times M}$  between span  $s_n$  and all spans are defined as follows:

$$r_n = softmax\left(\frac{((f_n \oplus g_n) \cdot W^{Q_r})(K_r \cdot W^{K_r})^T}{\sqrt{d_{k_r}}}\right), \quad (4)$$

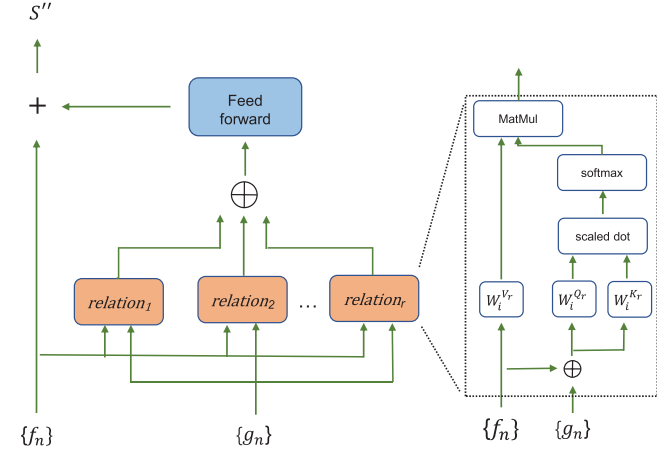


Fig. 3. Span relation representation computation. ‘+’ and  $\oplus$  denote sum and concatenation operation respectively.

and the relation representation of  $s_n$  is a relation-weighted sum over  $M$  spans:

$$r_n \cdot \begin{pmatrix} f_1 \cdot W^{V_r} \\ \vdots \\ f_M \cdot W^{V_r} \end{pmatrix}, \quad (5)$$

where  $K_r = \{f_n \oplus g_n\}_{n=1}^M$ .  $W^{Q_r}$ ,  $W^{K_r}$ , and  $W^{V_r}$  are linear projections, the output dimension of which is equal to  $d_{k_r}$ .

We extend a relation module to multiple relation modules, i.e.,  $relation_i$  ( $i = 1, \dots, r$ ) in Fig. 3 by the linear projections  $W_i^{Q_r}$ ,  $W_i^{K_r}$ , and  $W_i^{V_r}$ . The computation can be simply written as  $relation_i = Att(Q_r W_i^{Q_r}, K_r W_i^{K_r}, V_r W_i^{V_r})$  by Eq. (1), where  $Q_r = K_r = \{f_n \oplus g_n\}_{n=1}^M$ ,  $V_r = \{f_n\}_{n=1}^M$ . We concatenate the outputs of  $r$  relation modules and pass them through a feed-forward network. The final output of SRN is the sum of  $\{f_n\}$  and the output of the feed-forward network.

We explain the reason for adding the positional feature in the relation module. For example, the distance between “ $u_1 u_2 u_3$ ” and “ $u_2 u_3 u_4$ ” is close in feature space after averaging the token representations. To improve the discrimination between spans, we embed an additional positional feature  $\{g_n\}$  in computation of the relation weights to address the classification conflict caused by the overlapping adjacent spans. To make the positional feature irrelevant to the sentence length, we use the relative positional vector,  $pos = (i/l, j/l)$ , where  $i$  and  $j$  denote the start and end of a span respectively, and  $l$  is the length of the sentence. In terms of the positional vector, spans “ $u_1 u_2 u_3$ ” and “ $u_2 u_3 u_4$ ” can be better distinguished. Then, we encode the relative positional vector  $pos$  to a high-dimensional feature  $g = W^g \cdot pos$ , where  $W^g \in \mathbb{R}^{d_g \times 2}$ .

#### D. $H^pT$ Operation

The goal of  $H^pT$  operation is to convert the spans of different lengths to the fixed-length vectors and connect to a fully connected layer for classification. Several fixed-length span

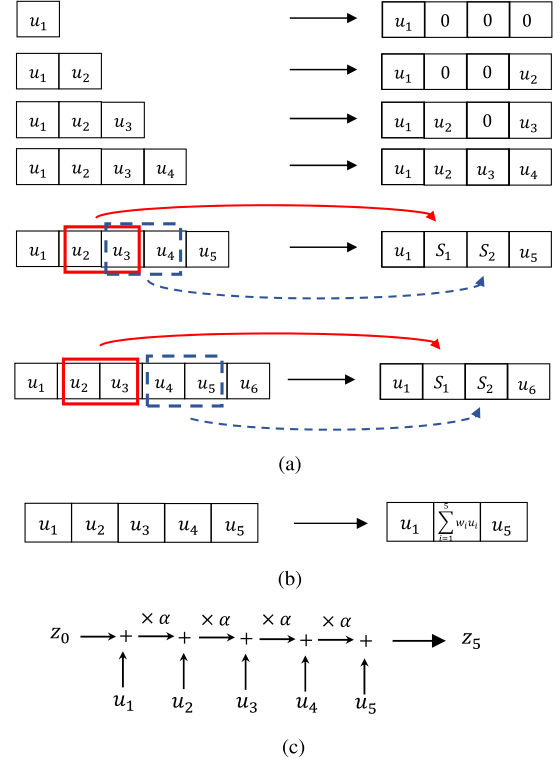


Fig. 4. Illustration of four span representations. (a)  $H^pT$  ( $p = 2$ ). The left side is the spans that have lengths from 1 to 6, and the right side is the fixed-length representations, (b) Attention weighted and HAT (if  $w_i = 1/5$ ), and (c) FOFE where  $z_0 = 0$ ,  $\alpha \in (0, 1)$  is a constant forgetting factor, and  $z_5$  is the FOFE code.

representation methods were proposed. Zheng *et al.* [38] proposed a fixed-size ordinally-forgetting encoding (FOFE) which was a recursive encoding method, as shown in Fig. 4(c). Lee *et al.* [39] presented a fixed-length representation consisting of boundary information (START and END) and an attention weighted sum of word vectors, as shown in Fig. 4(b). The variant of [39] with a low computational cost is to replace the attention-weighted sum with an average [12], [25], and called HAT operation in [21]. The weakness of current span representation methods is the lack of local inner information. In this work, we design a new fixed-length representation regarding the boundary and inner features, called  $H^pT$  (head-inner-tail). The advantage of the  $H^pT$  operation over the previous methods is that the span representation contains more inner segment information. The HAT operation averages all tokens of each span, but the  $H^pT$  operation divides the inner part of each span to  $p$  segments and represents the features of each inner segment (e.g.,  $\{S_1, S_2\}$  in Fig. 4(a)). Moreover, the output length of the  $H^pT$  operation is  $2+p$ , while that of HAT is 3.

Let  $l$  be the length of a span. The size of  $p$  segments is computed by  $\lfloor \frac{l-2}{p} \rfloor + ((l-2) \bmod p)$ . The number of overlapping tokens between two neighboring segments is  $(l-2) \bmod p$ . For example, in Fig. 4(a), a span “ $u_1 u_2 u_3 u_4 u_5$ ” of length 5 can be divided into  $p = 2$  segments of  $S_1 = \{u_2, u_3\}$  and  $S_2 = \{u_3, u_4\}$ , where  $u_3$  is the overlapping token between  $S_1$  and  $S_2$ . Finally, the  $H^pT$  operation is defined



**Algorithm 1:** H<sup>IP</sup>T Operation.

**Input:** Token sequence:  $u \in \mathbb{R}^{N_s \times m}$ ; start position of spans:  $start \in \mathbb{R}^{N_t}$ ; end position of spans:  $end \in \mathbb{R}^{N_t}$ ; segment size:  $p$ .  $N_s$  is the length of the token sequence.  $N_t$  is the number of spans.

**Output:**  $hit \in \mathbb{R}^{N_t \times (p+2) \times m}$

```

1: Initialize:  $hit = \mathbf{0}_{N_t \times (p+2) \times m}$ ;
    $length \leftarrow end - start$ ;  $u_c \leftarrow Cumsum(u)$ .
2:  $hit[:, 1, :] \leftarrow u[start, :]$ 
3:  $id \leftarrow Where(length > 1)$ 
4:  $hit[id, p+2, :] \leftarrow u[end[id], :]$ 
5: for  $span\_length = 3$  to  $p+2$  do
6:    $id \leftarrow Where(length = span\_length)$ 
7:    $hit[id, 2 : span\_length - 1, :] \leftarrow u[start[id] + 1 : start[id] + span\_length - 2, :]$ 
8: end for
9:  $id \leftarrow Where(length > (p+2))$ 
10:  $v \leftarrow \lfloor \frac{length[id]-2}{p} \rfloor$ ;  $o \leftarrow (length[id] - 2) \bmod p$ ;
    $s \leftarrow v + o$ 
11: for  $segment = 1$  to  $p$  do
12:    $hit[id, segment + 1, :] \leftarrow \frac{1}{s}(u_c[start[id] + segment \times v + o, :] - u_c[start[id] + (segment - 1) \times v, :])$ 
13: end for

```

as follows:

H<sup>IP</sup>T

$$= \begin{cases} [u_k; \mathbf{0}_{m \times 1}], & \text{if } l = 1 \\ [u_k; \dots; u_{k+l-2}; \mathbf{0}_{m \times (p-l+2)}; u_{k+l-1}], & \text{if } l \leq p+2 \\ [u_k; average(S_1); \dots; average(S_p); u_{k+l-1}], & \text{if } l > p+2, \end{cases} \quad (6)$$

where  $S_i$  is the  $i$ -th segment of the inner part of a span,  $\mathbf{0}$  is a zero matrix, and  $average(S)$  is the average value of token representations in segment  $S$ . If  $l \leq p+2$ , we pad the remaining slots with zeros  $\mathbf{0}_{m \times (p-l+2)}$ . For example, if  $l = 3$  and  $p = 2$ , the pooled output is  $\{u_1, u_2, 0, u_3\}$ . If  $p = 0$ , only the head and tail of spans are pooled out.

The H<sup>IP</sup>T operation is shown in Algorithm 1. To accelerate the operation, the algorithm operates the spans of the same length in parallel. In line 2 to 8 of the algorithm, when the length of spans is from 1 to  $p+2$ , we directly copy the results from the sequence  $u$ . If the length of spans is larger than  $p+2$ , in line 9 to 13, we calculate the average sum of segment 1 to  $p$ . In fact, we calculate the difference of the cumulative sum  $u_c[j, :] - u_c[i-1, :]$  instead of the sum of the segment  $\sum_{k=i}^j u[k, :]$ , thus the time complexity is reduced to  $\mathcal{O}(1)$  and all spans can be computed in parallel. In Algorithm 1, the function  $Where()$  returns the indices of elements where the given condition is satisfied, and the function  $Cumsum()$  returns the cumulative sum of array elements.

### E. Training Set

In training, we input sentences and their respective span set consisting of NEs and non-NE negatives. The non-NE negatives

are always from all possible spans of a sentence except NEs in the previous works [11], [12]. In this work, we use the Intersection over Union ( $IoU$ ) metric, also known as the Jaccard index [40], to filter non-NE spans. The advantages of non-NE span filtering are as follows: first, the number of negatives is reduced by approximately 14% to accelerate training, and second, the optimal value of  $IoU$  can prevent overlearning on the spans with high  $IoUs$ , e.g.,  $\geq 0.85$ .

The function  $IoU(a, b)$  measures how much overlap occurs between text strings  $a$  and  $b$ , and is defined as follows:

$$IoU(a, b) = \frac{length(a \cap b)}{length(a \cup b)}. \quad (7)$$

A span  $neg$  that could be considered as non-NE negative should satisfy

$$IoU(neg, g^{max}) < IoU_b, \quad (8)$$

where  $IoU_b$  is a threshold.  $g^{max}$  is an NE whose  $IoU$  with  $neg$  is the maximum for all NEs in a sentence, and defined as follows:

$$g^{max} = \arg \max_{g \in G} IoU(neg, g), \quad (9)$$

where  $G$  is a set of NEs in a sentence.

Here we give an example of filtering negative spans with parameter  $IoU_b$ . Given a named entity  $e = "t_1t_2t_3t_4"$ , non-NE spans  $sp_1 = "t_1t_2"$  and  $sp_2 = "t_1t_2t_3"$ , we can obtain  $IoU(sp_1, e) = 2/4 = 1/2$  and  $IoU(sp_2, e) = 3/4$ . If we set  $IoU_b = 1/2$ ,  $sp_1$  will be added to the collection of non-NE negatives, but  $sp_2$  will not because  $IoU(sp_2, e) > IoU_b$ . In the experiment, we also limit the length of spans by a parameter  $L_s$ .

## IV. EXPERIMENTS

### A. Datasets

We perform nested NER experiments on ACE corpora<sup>1</sup> and GENIA [6]. These datasets and Train/Dev/Test settings are briefly introduced as follows:

- **ACE:** The data of ACE are taken from newswire and broadcast news. The data contain 7 fine-grained entity categories, which are Person (PER), Organization (ORG), Geographical Entities (GPE), Location (LOC), Facility (FAC), Weapon (WEA) and Vehicle (VEH). Testing is performed on the English portions of ACE04 and ACE05. We split ACE04 and ACE05 into Train/Dev/Test sets randomly with the ratio of 8:1:1, respectively, as in previous work [2], [20].
- **GENIA:** This dataset is for biomedical NER. It involves 36 fine-grained entity categories of a total of 2000 MEDLINE abstracts. We follow the same task as previous work [9] and split the dataset into Train/Dev/Test sets sequentially with the ratio of 8:1:1. We use the Dev set to tune parameters. In evaluating the Test set, we concatenate Train+Dev portions as in [2], [3], [10], [12] for a direct comparison.

### B. Settings

Table I shows the hyperparameter values in TCSF. Each mini-batch has 8 sentences of the same length and their respective

<sup>1</sup>[Online]. Available: <https://catalog.ldc.upenn.edu/LDC2005T09> (ACE04) and [Online]. Available: <https://catalog.ldc.upenn.edu/LDC2006T06> (ACE05)

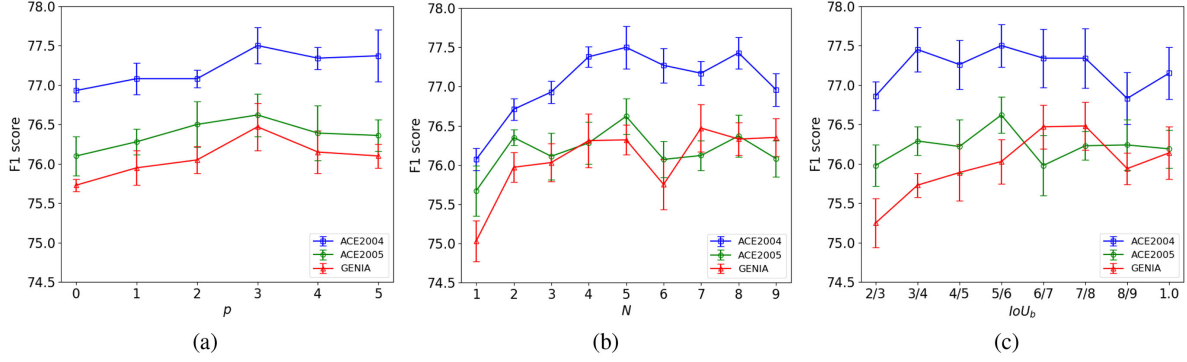


Fig. 5. Performance comparison with different parameter values on TCSF. (a) segment number  $p$ , (b) CNN depth  $N$ , and (c) filtering threshold  $IoU_b$ .

TABLE I  
HYPERPARAMETERS OF THE TCSF MODEL

| Hyperparameter                          | Value |
|---|-------|
| char embedding dimension                | 28    |
| positional feature dimension $d_g$      | 8     |
| number of heads in TCN $h$              | 4     |
| number of relations in SRN $r$          | 4     |
| CNN kernel size                         | 3     |
| output dimension of the first FC        | 256   |
| # if concat BERT                        | 1024  |
| segment number $p$ in $HI^pT$ operation | 3     |
| dropout rate for TCN and SRN            | 0.1   |
| dropout rate for others                 | 0.5   |
| learning rate                           | 3-e4  |
| max span length $L_s$                   | 10    |

training spans. We set  $IoU_b = 5/6$ ,  $N = 5$  on the ACE datasets, and  $IoU_b = 7/8$ ,  $N = 7$  on the GENIA dataset for optimal performance.

We use GloVe [36] 100-dimensional word embeddings for ACE and 200-dimensional biomedical word embeddings<sup>2</sup> for GENIA. BERT (large-uncased) and SciBERT (scivocab-uncased) [41] are employed in the pretrained LM testing. Inspired by the weighted sum of hidden layers in ELMo [28], we use the weighted sum of the last four hidden layers  $h_j$  of BERT, i.e.,  $\sum_{j=1}^4 w_j h_j$ .  $w_j$  are softmax-normalized weights and tuned in training. To align with the tokens from TCN, we average the embeddings of BERT-tokenized subwords to generate an approximate vector for out-of-vocabulary (OOV) words.

The TCSF model is implemented in PyTorch. We use dropout regularization to prevent neural networks from overfitting. We train the model using Adam [42] optimizer with default settings. The code and trained model of TCSF are available on GitHub.<sup>3</sup>

Since the NE detection results of our model do not contain any partially overlapped mentions, we do not employ any post-processing method for further processing after the classification of all possible spans. The reasons why no partially overlapped mentions are detected are: 1) The model uses partially overlapped spans as non-NE negatives in training. For example, assume a sentence “ab[cde]<sub>NE</sub>f” with an entity “cde,” the partial overlaps {“abc,” “bcd,” “def,” “ef”} are included in the collection of non-NE negatives. 2) NE and non-NE classification

capability are well improved by span feature learning of the deep CNN and SRN, leading to the correct recognition of partially overlapped spans. In the following Figs. 6, 7, and 8, we depict some evidence of improved classification performance by the deep CNN and SRN.

We train the models on one machine with an NVIDIA Tesla K80 (GPU) and Intel Xeon Silver 4114 Processor 2.2 GHz (CPU). The TCSF model takes an average of 96 seconds per epoch on the ACE dataset and 226 seconds per epoch on the GENIA dataset. To achieve the overall optimal performance, it takes approximately 2.4 hours on the ACE dataset and 4.4 hours on the GENIA dataset.

### C. Optimal Value of Model Parameters

In this section, we tune the performance of three parameters of TCSF on the Dev set. We report F1 scores by the means and 95% confidence intervals from eight runs in Fig. 5.

- **segment number  $p$  in  $HI^pT$ :** Fig. 5(a) shows the performance of different segment numbers in the  $HI^pT$  operation. The optimal segment number is 3 for the ACE and GENIA datasets. The  $HI^3T$  operation increases the mean F1 scores by 0.7%, 0.6%, and 0.9% compared to those of the  $HI^0T$  operation on the ACE04, ACE05, and GENIA datasets, respectively. The results show that local inner information of spans is beneficial for span classification.
- **depth  $N$  of CNNs:** We have explored nine depths for the deep residual CNN: 1-9. The results in Fig. 5(b) show that the deep residual CNN can improve the performance. F1 score increases, starting from depth 1. The optimal depth for the ACE and GENIA datasets is 5 and 7, respectively. The mean F1 scores increase by 1.5%, 1.0%, and 1.4% on the ACE04, ACE05, and GENIA datasets respectively, compared to those at a depth of 1.
- **$IoU_b$  in span filtering:** Fig. 5(c) illustrates the performance of TCSF while varying  $IoU_b$  on the Dev set. We set  $IoU_b = \{2/3, 3/4, 4/5, 5/6, 6/7, 7/8, 8/9, 1.0\}$  in the form of a fractional number. From the results, we find that F1 scores achieve the optimal values when  $IoU_b = 5/6$  for ACE and  $IoU_b = 7/8$  for GENIA. The mean F1 scores of the optimal value are 0.4% on average higher than those of all possible spans (i.e.,  $IoU_b = 1.0$ ).

<sup>2</sup>wikipedia-pubmed-and-PMC-w2v in the URL of <http://bio.nlpab.org/>

<sup>3</sup>[Online]. Available: <https://github.com/Nested-NER/TCSF>



TABLE VI  
ABLATION STUDY ON THE TEST SET. T IS TCN, C IS DEEP RESIDUAL CNN, AND S IS SRN. ALL NUMBERS ARE F1 SCORES (%)

|              | ACE04                  | ACE05                  | GENIA                  |
|--------------|------------------------|------------------------|------------------------|
| No T+C+S     | 55.1                   | 56.0                   | 62.5                   |
| T            | 63.9                   | 64.4                   | 66.4                   |
| C            | 68.8                   | 67.2                   | 74.6                   |
| S            | 55.8                   | 56.5                   | 67.8                   |
| T+C          | 77.3 (T 13.4↑, C 8.5↑) | 76.8 (T 12.4↑, C 9.6↑) | 76.7 (T 10.3↑, C 2.1↑) |
| T+S          | 69.9 (T 6.0↑, S 14.1↑) | 68.8 (T 4.4↑, S 13.3↑) | 74.0 (T 7.6↑, S 6.2↑)  |
| C+S          | 69.4 (C 0.6↑, S 12.9↑) | 67.6 (C 0.4↑, S 11.1↑) | 74.9 (C 0.3↑, S 7.1↑)  |
| T+C+S (TCSF) | <b>78.6</b>            | <b>77.5</b>            | <b>77.3</b>            |

TABLE VII  
COMPARISON OF SPAN REPRESENTATION METHODS

|                             | ACE04              | ACE05              | GENIA              |
|-----------------------------|--------------------|--------------------|--------------------|
| FOFE [11]                   | 75.89±0.22 / 3e-11 | 75.25±0.35 / 4e-7  | 75.18±0.36 / 7e-8  |
| HAT [12, 21]                | 77.49±0.30 / 0.001 | 76.58±0.18 / 0.003 | 76.44±0.29 / 0.010 |
| Attention weighted [39]     | 77.41±0.31 / 0.001 | 76.46±0.27 / 0.002 | 76.26±0.25 / 0.005 |
| HI <sup>P</sup> T ( $p=3$ ) | <b>78.24±0.25</b>  | <b>77.20±0.28</b>  | <b>76.95±0.32</b>  |

is achieved compared with T. From the results, we find that the performance of the deep residual CNN is better than that of the single TCN and SRN. In the two-component combination, T+C is the best. Adding the span feature learning by C and S to the token contextual learning by T achieves an average increase of 12% and 6%, respectively. In addition, F1 scores of T+C+S (TCSF) increase by an average of 1.0% on ACE and 0.6% on GENIA, compared to that of T+C (TCSF w/o SRN).

Furthermore, we visualize two examples in a 2D plot for input token embeddings and output representations of the TCN and CNN in Fig. 6 and 7 respectively. The token features are reduced to 2 dimensions by principal components analysis (PCA). These two figures show the effects of TCN and deep residual CNN on token classification. We find that the clustering of tokens of NEs has improved from the input embeddings to TCN, then to deep residual CNN. On the output representations of deep residual CNN, the dashed line separates two classes correctly.

Table VII shows the performance comparison of four span representation methods on the Test set. We replace the HI<sup>P</sup>T operation part of the TCSF in Fig. 2 with the other three methods. Fixed-size ordinaly forgetting encoding (FOFE) [38] encodes a word sequence by a recursive formula with a forgetting factor  $\alpha$ . Xu *et al.* [11] additionally generated left and right context FOFE codes for span features. We conduct a comparison with the FOFE span representation, which is the concatenation of five FOFE codes, consisting of the span, the left and right context including the span, and the left and right context excluding the span. The standard [START, Weighted\_sum, END] [39] consists of the boundary and sum of words by attention weights. HAT [12], [21] simplifies the attention weights by an average operation. We employ Welch's t-test [43] which assesses whether there is a significant difference between the means of two sets, to perform significance testing. We train models eight times for each span representation method and calculate  $p$ -values by comparing HI<sup>P</sup>T ( $p = 3$ ) with the other three methods. The results are represented as F1 score mean  $\pm$  standard deviation/ $p$ -value in Table VII. The mean F1 scores achieved by HI<sup>P</sup>T( $p = 3$ ) outperform 0.68% on average over Attention weighted and HAT, showing the effects of inner information of the spans for representation. Although the improvement is not large, the

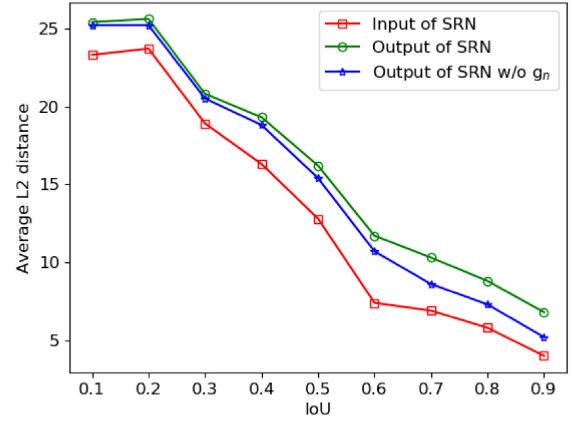


Fig. 8. Comparison of the input and output of SRN on the distance between NEs and their overlapping negatives on ACE05.

$p$ -values show that the differences between HI<sup>P</sup>T ( $p = 3$ ) and the other three methods are statistically significant.

#### F. Analysis of SRN

In the ablation study, the results show that the use of the SRN produces a good effect for span representation. Here we illustrate the role of SRN in improving span classification. Fig. 8 shows the comparison between the input  $S'$  and output  $S''$  of SRN in the L2 or Euclidean distance of span features with varying  $IoUs$ . We normalize span features and compute the L2 distance between an entity  $e$  and a negative span  $neg$ . Then, we count the average distance regarding different  $IoU(e, neg)$  values. Obviously, the distance is smaller as the  $IoU$  increases because the overlapping adjacent spans are more similar in appearance. The average distance is from 23.3 to 4.0 when the  $IoU$  increases from 0.1 to 0.9 in the line of “input of SRN”. Compared to “input of SRN,” the line of “Output of SRN” enlarges the distance between classified samples, increasing by 9.0%, 26.6%, and 70.0% when the  $IoU$  is equal to 0.1, 0.5, and 0.9, respectively. It also shows that SRN can especially increase the distance between the spans with large overlappings ( $IoU > 0.5$ ). A larger classification margin results in better classification performance,



TABLE VIII  
EFFECTS OF SRN AND POSITIONAL FEATURE IN SRN

|                       | ACE04              | ACE05              | GENIA              |
|-----------------------|--------------------|--------------------|--------------------|
| TCSF w/o SRN          | 76.91±0.24 / 3e-6  | 76.54±0.32 / 0.002 | 76.31±0.37 / 0.006 |
| TCSF (SRN w/o $g_n$ ) | 77.64±0.34 / 0.007 | 76.85±0.30 / 0.022 | 76.59±0.29 / 0.037 |
| TCSF                  | <b>78.24±0.25</b>  | <b>77.20±0.28</b>  | <b>76.95±0.32</b>  |

TABLE IX  
COMPARISON WITH STATE-OF-THE-ART MODELS ON THE ACE AND GENIA DATASETS. (†) INDICATES THE RESULTS OF THE ORIGINAL PAPERS' MISSING DATA SETS USING THE AUTHORS' SOURCE CODES

|                        | Method type   | ACE04 |      |             | ACE05 |      |             | GENIA |      |             |
|------------------------|---------------|-------|------|-------------|-------|------|-------------|-------|------|-------------|
|                        |               | P(%)  | R(%) | F1(%)       | P(%)  | R(%) | F1(%)       | P(%)  | R(%) | F1(%)       |
| Ju et al. [1]          | stacked layer | 76.6  | 71.0 | 73.6 (†)    | 74.2  | 70.3 | 72.2        | 78.5  | 71.3 | 74.7        |
| Katiyar et al. [2]     | hypergraph    | 73.6  | 71.8 | 72.7        | 70.6  | 70.4 | 70.5        | 76.7  | 71.1 | 73.8        |
| Wang and Lu [10]       | hypergraph    | 78.0  | 72.4 | 75.1        | 76.8  | 72.3 | 74.5        | 77.0  | 73.3 | 75.1        |
| Lin et al. [25]        | span-based    | 74.8  | 71.5 | 73.1 (†)    | 76.2  | 73.6 | 74.9        | 75.8  | 73.9 | 74.8        |
| Zheng et al. [26]      | span-based    | 72.0  | 71.2 | 71.6 (†)    | 71.8  | 69.8 | 70.8 (†)    | 75.8  | 73.6 | 74.7        |
| Sun et al. [21]        | span-based    | 79.2  | 73.3 | 76.1        | 79.7  | 72.0 | 75.6        | 75.6  | 74.9 | 76.2        |
| Fisher and Vlachos [4] | merge tokens  | 74.4  | 71.5 | 72.9 (†)    | 75.1  | 74.1 | 74.6        | 72.9  | 69.6 | 71.2 (†)    |
| seq2seq [3]            | hypergraph    | -     | -    | 77.1        | -     | -    | 75.4        | -     | -    | 76.4        |
| TCSF                   | span-based    | 79.7  | 77.6 | <b>78.6</b> | 77.4  | 77.6 | <b>77.5</b> | 78.2  | 76.5 | <b>77.3</b> |

i.e., an increase of 1.33% in the mean F1 score on ACE04, 0.66% on ACE05, and 0.64% on GENIA, compared to TCSF w/o SRN, as shown in Table VIII. The figure also presents the line of “Output of SRN w/o  $g_n$ ” when the positional feature  $g_n$  is ablated from SRN. With the increase of the  $IoU$ , especially in [0.7, 0.9], the average L2 distance of “Output of SRN w/o  $g_n$ ” drops more than that of “Output of SRN”. That is, without the positional feature, the performance of SRN is weakened in the discrimination task of the overlapping adjacent spans. F1 score of SRN w/o  $g_n$  also confirms this, as shown in Table VIII.

Table VIII shows F1 score mean±standard deviation and  $p$ -value in t-test for comparison between the mean F1 scores of TCSF and other two settings, which are w/o SRN and SRN w/o positional feature  $g_n$  respectively. The  $p$ -values ( $<0.05$ ) show that the performance difference is statistically significant on the ablation of  $g_n$ , although the improvements of positional feature  $g_n$  are small, approximately 0.44% on average.

### G. Comparison With State-of-the-Art Models

Table IX shows the comparison with state-of-the-art models without any extra resource or pretrained LM. The source codes of all comparison methods are available online. The performance results are retrieved from the original paper. We also use the source codes of Ju *et al.* [1], Lin *et al.* [25], Zheng *et al.* [26], and Fisher and Vlachos [4], to test the missing data sets and show the results marked by (†). In the seq2seq model [3], the authors trained the model on Train+Dev portions for all datasets. Actually, for the scenario under the same data settings, the TCSF model outperforms the current best results by more than 2% in F1 score on the ACE datasets.

Table X shows the comparison of F1 scores with the latest published results using BERT on the Test sets of the ACE and GENIA datasets. F1 score of TCSF+BERT increases by 8.8% and 9.4% on the ACE04 and ACE05 datasets, respectively. BERT and SciBERT achieve an increase of 1.5% and 2.8%, respectively, on the GENIA dataset, compared to TCSF. Here we clarify some of the differences between DyGIE++ and our model as follows: 1) DyGIE++ is a multi-task framework for

TABLE X  
F1 SCORE (%) COMPARISON WITH OTHER MODELS USING PRETRAINED LMS

|                          | ACE04       | ACE05       | GENIA       |
|--------------------------|-------------|-------------|-------------|
| seq2seq [3]              |             |             |             |
| +BERT                    | 84.3        | 83.4        | 78.2        |
| +BERT+Flair              | <b>84.4</b> | <b>84.3</b> | <b>78.4</b> |
| DyGIE++ [29]             |             |             |             |
| (BERT, Best, w=3)        | 86.5        | 84.3        | 76.7        |
| (BERT, NER task, w=3)    | <b>86.7</b> | <b>84.6</b> | 76.4        |
| (BERT, NER task, w=1)    | 86.2        | 83.8        | 76.1        |
| (SciBERT, Best, w=3)     | 82.9        | 80.5        | <b>78.7</b> |
| (SciBERT, NER task, w=3) | 83.0        | 80.9        | 78.0        |
| (SciBERT, NER task, w=1) | 82.4        | 79.8        | 77.4        |
| TCSF                     |             |             |             |
| +BERT                    | <b>87.4</b> | <b>86.9</b> | 78.8        |
| +SciBERT                 | 83.2        | 82.1        | <b>80.1</b> |

three IE tasks: named entity recognition, relation extraction, and event extraction. The approach also performed coreference resolution as an auxiliary task to improve the performance. In the ACE05 training, the coreference annotations were from a large corpus OntoNotes [44]. DyGIE++ aimed to improve the performance of single tasks by co-training the multiple IE tasks in a combined manner. The authors of DyGIE++ provided the best co-training configurations on ACE05 and GENIA, see the website of DyGIE++.<sup>4</sup> 2) In terms of preprocessing of ACE05, there is a significant difference between DyGIE++ and many previous nested NER works. The preprocessing of ACE05 in DyGIE++ followed the code from Miwa and Bansal [45]. Miwa and Bansal used the head spans for entities and a 351/80/80 train/dev/test split. Doddington *et al.* [5] annotated both the maximal extent of the string that represented the entity and the head of each mention. The head spans contained few nested NEs, e.g., 5 nested NEs on ACE05, therefore we should use the extent spans to evaluate the nested NER models. We follow the same preprocessing and an 8:1:1 split as Lu and Roth [19]. 3) DyGIE++ fed each sentence to BERT together with surrounding

<sup>4</sup>The best NER configuration file of ACE05: [https://github.com/dwadden/dygiepp/blob/master/training\\_config/ace05\\_best\\_ner\\_bert.jsonnet](https://github.com/dwadden/dygiepp/blob/master/training_config/ace05_best_ner_bert.jsonnet). The best NER configuration file of GENIA: [https://github.com/dwadden/dygiepp/blob/master/training\\_config/genia\\_working\\_example.jsonnet](https://github.com/dwadden/dygiepp/blob/master/training_config/genia_working_example.jsonnet).

TABLE XI  
NESTED ENTITY DETECTION FAILURE CASES

|   |
|---|
| <b>ACE dataset:</b>   |
| So [they] <sub>PER</sub> → <b>ORG</b> want to look for alternative sources for [planes] <sub>VEH</sub> , [helicopters] <sub>VEH</sub> and [other hardware] <sub>WEA</sub> .   |
| [the popular band] <sub>ORG</sub> has been criticized within [[country] <sub>ORG</sub> → <b>non-NE</b> music circles] <sub>ORG</sub> → <b>PER</b> for comments by [[lead singer] <sub>PER</sub> nathalie maines] <sub>PER</sub> . |
| And, currently, [both] <sub>PER</sub> → <b>GPE</b> have opposed the [U.S.] <sub>GPE</sub> - led war on [Iraq] <sub>GPE</sub> .  |
| <b>GENIA dataset:</b>   |
| We used [mouse embryonic stem ( ES ) cells] <sub>cell_line</sub> → <b>cell_type</b> to study [globin gene] <sub>DNA</sub> expression and switching in vitro.  |
| The formation of this complex also requires both [[AP - 1] <sub>protein</sub> → <b>non-NE</b> sites] <sub>DNA</sub> and correlates with maximal [enhancer] <sub>DNA</sub> activity .  |

sentences to capture cross-sentence context. The authors showed that cross-sentence context (w=3) improved the NER performance. Our model studies a single NER task and does not use cross-sentence context, similar to many previous NER works. This setting is more adaptable to many other NER datasets.

To compare our TCSF with DyGIE++, we use the available code<sup>5</sup> to test DyGIE++ in the configurations of Best (“NER+relation” tasks for ACE, “NER+coreference” tasks for GENIA), NER task with w=3, and NER task with w=1. NER task with w=1 is the same setting as many previous NER works and ours. Our model performs the best on all nested NER datasets in Table X. The best co-training configuration of DyGIE++ increases F1 score by an average of 0.5% on GENIA, but does not help improve the performance of NER task on ACE. F1 score of w=3 increases by an average of 0.7% on ACE and 0.5% on GENIA, compared to w=1. When DyGIE++ is implemented under the same experimental setting as our model, F1 scores of TCSF+BERT are 1.2% and 3.1% higher than that of DyGIE++(BERT, NER task, w=1) on ACE04 and ACE05 respectively, and TCSF+SciBERT is 2.7% higher than that of DyGIE++(SciBERT, NER task, w=1) on GENIA. Additionally, F1 score of TCSF without BERT is 77.3% on GENIA, which is comparable to that of DyGIE++ (SciBERT, NER task, w=1), and 1.2% higher than that of DyGIE++(BERT, NER task, w=1).

Several entity detection failure cases in the best results of our model are shown in Table XI, where the red font represents the correct label if the detected result is wrong. Most of the detected mentions in the failure cases are on the correct location, and errors are mainly incurred by NE classification. The pronouns such as “they” and “both” are often recognized as PER, since the number of pronouns of PER is large in the training set. Increasing the number of pronouns of other NE types or applying sampling techniques to handle imbalanced data is a potential remedy. The polysemous words such as “circles” introduce more difficulty to NER, although state-of-the-art contextual pretrained models are used. In addition, allowing nesting between NEs increases the possibility of misdetection such as “country” and “AP - 1”. The classification of biomedical terms, such as

“mouse embryonic stem (ES) cells” and “enhancer,” particularly depends on context. Pretraining language models on biomedical vocabulary might be a solution to learn embeddings of better quality regarding biomedical terms.

## V. CONCLUSION

This paper has presented a novel span-based TCSF model for nested NER. Our model is fully end-to-end and highly parallelizable on GPU. TCSF has implemented a joint learning framework for token and span features. The ablation study shows that the span features learned by the deep residual CNN are more important than the token features learned by the TCN. The combination of TCN for token learning and deep residual CNN and SRN for span learning achieves the best performance. The span relation level representation by SRN with the positional feature can increase the classification margin between NEs and non-NE spans, especially with a large overlap. Moreover, the HI<sup>P</sup>T operation for the fixed-length representation provides a flexible way to extract the inner features of spans. The experimental results show that our model achieves state-of-the-art performance not only without any extra resource but also with pretrained LMs.

## ACKNOWLEDGMENT

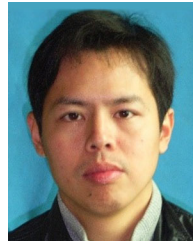
We would like to thank the anonymous reviewers for their valuable comments.

## REFERENCES

- [1] M. Ju, M. Miwa, and S. Ananiadou, “A neural layered model for nested named entity recognition,” in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist. Human Lang. Technol., Volume 1 (Long Papers)*, vol. 1, 2018, pp. 1446–1459.
- [2] A. Katiyar and C. Cardie, “Nested named entity recognition revisited,” in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.-Assoc. Comput. Linguist., Volume 1 (Long Papers)*, vol. 1, 2018, pp. 861–871.
- [3] J. Straková, M. Straka, and J. Hajic, “Neural architectures for nested ner through linearization,” in *Proc. 57th Conf. Assoc. Comput. Linguist.*, 2019, pp. 5326–5331.
- [4] J. Fisher and A. Vlachos, “Merge and label: A novel neural network architecture for nested ner,” in *Proc. 57th Annu. Meet. Assoc. Comput. Linguist.*, 2019, pp. 5840–5850.
- [5] G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. M. Strassel, and R. M. Weischedel, “The automatic content extraction (ACE) program-tasks, data, and evaluation,” in *Proc. Int. Conf. Lang. Res. Eval.*, vol. 2, no. 1., 2004, pp. 837–840.
- [6] J.-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii, “Genia corpora semantically annotated corpus for bio-textmining,” *Bioinformatics*, vol. 19, no. suppl\_1, pp. i180–i182, 2003.
- [7] J. P. Chiu and E. Nichols, “Named entity recognition with bidirectional LSTM-CNNs,” *Trans. Assoc. Comput. Linguist.*, vol. 4, pp. 357–370, 2016.
- [8] V. Yadav and S. Bethard, “A survey on recent advances in named entity recognition from deep learning models,” in *Proc. 27th Int. Conf. Comput. Linguist.*, 2018, pp. 2145–2158.
- [9] J. R. Finkel and C. D. Manning, “Nested named entity recognition,” in *Proc. Conf. EMNLP: Volume 1. Assoc. for Comput. Linguist.*, 2009, pp. 141–150.
- [10] B. Wang and W. Lu, “Neural segmental hypergraphs for overlapping mention recognition,” in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 204–214.
- [11] M. Xu, H. Jiang, and S. Watcharawittayakul, “A local detection approach for named entity recognition and mention detection,” in *Proc. 55th Annu. Meet. Assoc. Comput. Linguist. (Volume 1: Long Papers)*. Vancouver, Canada, Jul. 2017, pp. 1237–1247.
- [12] M. G. Sohrab and M. Miwa, “Deep exhaustive model for nested named entity recognition,” in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 2843–2849.

<sup>5</sup>[Online]. Available: <https://github.com/dwadden/dygiepp/>

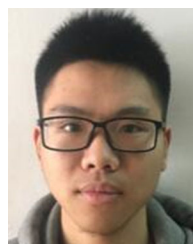
- [13] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF," in *Proc. 54th Annu. Meet. Assoc. Comput. Linguist. (Volume 1: Long Papers)*, 2016, pp. 1064–1074.
- [14] B. Gu, "Recognizing nested named entities in genia corpus," in *Proc. Workshop Linking Natural Lang. Process. Biol. Towards Deeper Biol. Literature Anal.*, 2006, pp. 112–113.
- [15] F. Li, M. Zhang, B. Tian, B. Chen, G. Fu, and D. Ji, "Recognizing irregular entities in biomedical text via deep neural networks," *Pattern Recognit. Lett.*, vol. 105, pp. 105–113, 2018.
- [16] B. Tang, Y. Wu, M. Jiang, J. C. Denny, and H. Xu, "Recognizing and encoding disorder concepts in clinical text using machine learning and vector space model." *CLEF (Work. Notes)*, vol. 665, 2013.
- [17] U. Yaseen, P. Gupta, and H. Schütze, "Linguistically informed relation extraction and neural architectures for nested named entity recognition in bionlp-ost 2019," in *Proc. The 5th Workshop BioNLP Open Shared Tasks*, 2019, pp. 132–142.
- [18] T.-S. Nguyen and L.-M. Nguyen, "Nested named entity recognition using multilayer recurrent neural networks," in *Proc. Int. Conf. Pacific Assoc. Comput. Linguist.*, Springer, 2017, pp. 233–246.
- [19] W. Lu and D. Roth, "Joint mention extraction and classification with mention hypergraphs," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 857–867.
- [20] A. O. Muis and W. Lu, "Labeling gaps between words: Recognizing overlapping mentions with mention separators," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 2608–2618.
- [21] L. Sun, F. Ji, K. Zhang, and C. Wang, "Multilayer toi detection approach for nested ner," *IEEE Access*, vol. 7, pp. 186 600–186 608, 2019.
- [22] S. Sarawagi and W. W. Cohen, "Semi-Markov conditional random fields for information extraction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 1185–1192.
- [23] M. Sano, H. Shindo, I. Yamada, and Y. Matsumoto, "Segment-level neural conditional random fields for named entity recognition," in *Proc. 8th Int. Joint Conf. Natural Lang. Process. (Volume 2: Short Papers)*, 2017, pp. 97–102.
- [24] Y. Zhang and J. Yang, "Chinese ner using lattice LSTM," in *Proc. 56th Annu. Meet. Assoc. Comput. Linguist. (Volume 1: Long Papers)*, 2018, pp. 1554–1564.
- [25] H. Lin, Y. Lu, X. Han, and L. Sun, "Sequence-to-nuggets: Nested entity mention detection via anchor-region networks," in *Proc. 57th Annu. Meet. Assoc. Comput. Linguist.*, Florence, Italy, Jul. 2019, pp. 5182–5192.
- [26] C. Zheng, Y. Cai, J. Xu, H.-f. Leung, and G. Xu, "A boundary-aware neural model for nested named entity recognition," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, Hong Kong, China, Nov. 2019, pp. 357–366.
- [27] Y. Luan, D. Wadden, L. He, A. Shah, M. Ostendorf, and H. Hajishirzi, "A general framework for information extraction using dynamic span graphs," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.: Human Lang. Technol., Volume 1 (Long and Short Papers)*, 2019, pp. 3036–3046.
- [28] M. E. Peters *et al.*, "Deep contextualized word representations," in *Proc. North Amer. Chapter Assoc. Comput. Linguist.-Assoc. Comput. Linguist.*, 2018, pp. 2227–2237.
- [29] D. Wadden, U. Wennberg, Y. Luan, and H. Hajishirzi, "Entity, relation, and event extraction with contextualized span representations," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process.*, 2019, pp. 5788–5793.
- [30] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.: Human Lang. Technol., Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.
- [31] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in *Proc. 27th Int. Conf. Comput. Linguist.*, 2018, pp. 1638–1649.
- [32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
- [33] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
- [34] B. McCann, J. Bradbury, C. Xiong, and R. Socher, "Learned in translation: Contextualized word vectors," in *Proc. Adv. Neural Inform. Process. Syst.*, 2017, pp. 6294–6305.
- [35] M. A. Islam, S. Jia, and N. D. Bruce, "How much position information do convolutional neural networks encode?" in *Proc. Int. Conf. Learn. Representations*, 2019.
- [36] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1532–1543.
- [37] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.-Assoc. Comput. Linguist.*, 2016, pp. 260–270.
- [38] S. Zhang, H. Jiang, M. Xu, J. Hou, and L. Dai, "The fixed-size ordinally-forgetting encoding method for neural network language models," in *Proc. 53rd Annu. Meet. Assoc. Comput. Linguist. 7th Int. Joint Conf. Natural Lang. Process.*, Beijing, China, vol. 2, Jul. 2015, pp. 495–500.
- [39] K. Lee, L. He, M. Lewis, and L. Zettlemoyer, "End-to-end neural coreference resolution," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 188–197.
- [40] M. Levandowsky and D. Winter, "Distance between sets," *Nature*, vol. 234, no. 5323, pp. 34–35, 1971.
- [41] I. Beltagy, A. Cohan, and K. Lo, "Scibert: Pretrained contextualized embeddings for scientific text," 2019, *arXiv:1903.10676*.
- [42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
- [43] R. Dror, G. Baumer, S. Shlomov, and R. Reichart, "The Hitchhiker's guide to testing statistical significance in natural language processing," in *Proc. 56th Annu. Meet. Assoc. for Comput. Linguist.*, Melbourne, Australia, vol. 1, Jul. 2018, pp. 1383–1392.
- [44] S. Pradhan, A. Moschitti, N. Xue, O. Uryupina, and Y. Zhang, "Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes," in *Proc. Joint Conf. Empirical Methods Natural Lang. Process. CoNLL-Shared Task*, 2012, pp. 1–40.
- [45] M. Miwa and M. Bansal, "End-to-end relation extraction using LSTMs on sequences and tree structures," in *Proc. 54th Annu. Meet. Assoc. Comput. Linguist.*, Berlin, Germany, vol. 1, Aug. 2016, pp. 1105–1116.



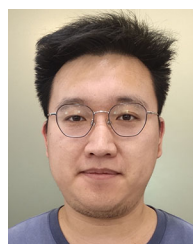
**Lin Sun** (Member, IEEE) received the B.S. degree in communication engineering and the M.S. degree in computer science from the East China University of Science and Technology, China, in 2001 and 2004, respectively. He received Ph.D. degree in computer science from Zhejiang University, China, in 2010. He is currently an Associate Professor with the department of computer science of Zhejiang University City College. His research interests include natural language processing and computer vision.



**Yuxuan Sun** is currently an undergraduate with the department of computer science of Zhejiang University City College. His research interests include deep learning and data mining.



**Fule Ji** is currently an undergraduate with the department of computer science of Zhejiang University City College. His research interests include deep learning and big data.



**Chi Wang** received the B.S. degree in computer science from the University of Electronic Science and Technology of China, in 2017. He is currently a Postgraduate with the College of Computer Science and Technology of Zhejiang University. His research interests include natural language processing and Internet of Things.