Boundary Enhanced Neural Span Classification for Nested Named Entity Recognition

Chuanqi Tan[†] **Wei Qiu**[†] **Mosha Chen**[†] **Rui Wang**[†] **Fei Huang**[†]
[†]Alibaba Group
[†] {chuanqi.tcq, qiuwei.cw, chenmosha.cms, masi.wr, f.huang}@alibaba-inc.com

Abstract

Named entity recognition (NER) is a well-studied task in natural language processing. However, the widely-used sequence labeling framework is usually difficult to detect entities with nested structures. The span-based method that can easily detect nested entities in different subsequences is naturally suitable for the nested NER problem. However, previous span-based methods have two main issues. First, classifying all subsequences is computationally expensive and very inefficient at inference. Second, the span-based methods mainly focus on learning span representations but lack of explicit boundary supervision. To tackle the above two issues, we propose a boundary enhanced neural span classification model. In addition to classifying the span, we propose incorporating an additional boundary detection task to predict those words that are boundaries of entities. The two tasks are jointly trained under a multitask learning framework, which enhances the span representation with additional boundary supervision. In addition, the boundary detection model has the ability to generate high-quality candidate spans, which greatly reduces the time complexity during inference. Experiments show that our approach outperforms all existing methods and achieves 85.3, 83.9, and 78.3 scores in terms of F_1 on the ACE2004, ACE2005, and GENIA datasets, respectively.

Introduction

Named entity recognition (NER) is a fundamental task in the field of natural language processing. It aims to identify text spans to specific entity types such as Person, Organization, and Location, which benefits many downstream NLP applications. Previous works usually treat NER as a sequence labeling task. For example, Lample et al. (2016) propose the LSTM-CRF model, which achieves promising results by combining deep recurrent neural networks (RNNs) with conditional random fields (CRFs) (Lafferty, McCallum, and Pereira 2001). However, Finkel and Manning (2009) point out that named entities are often nested. For example, 43.27% and 37.35% entities are nested in the ACE2004 and ACE2005 datasets, respectively. Figure 1 and Figure 2 show two examples in the ACE2005 and GENIA datasets, respectively. In the first example, "Britain" is an entity with the type of "GPE". It is nested in "Britain's ITN" with the type of "ORG". They are further nested in "Reporter Carl Dinnon of Britain 's ITN" with the type of "PER". The above nested structure cannot be handled by the predominant sequence labeling models.

Figure 1: An example in the ACE dataset.

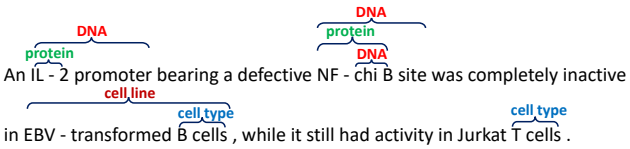

Figure 2: An example in the GENIA dataset.

Various approaches for nested NER have been proposed in recent years. One representative direction is based on hypergraph-based methods (Lu and Roth 2015; Wang and Lu 2018) that recognize nested entities by designing expressive tagging schemas. However, the hypergraph-based method needs a lot of human efforts to carefully design the unambiguous hypergraph. Another direction is based on span-based methods that recognize nested entities by classifying subsequences of the sentence (Xu, Jiang, and Watcharawittayakul 2017; Sohrab and Miwa 2018; Xia et al. 2019). The span-based method has its own advantages that we can easily find all candidate entities with different subsequences, which is straightforward and does not need human efforts. We therefore solve the nested NER task with the span-based method in this work. However, the span-based method still has two main issues. First, classifying all subsequences in the sentence is computationally expensive. Second, the span-based methods mainly focus on learning span representations but lack of explicit boundary supervision. Compared with methods under the sequence labeling framework and hypergraph methods, we observe that span-based methods usually perform worse in detecting bound-

aries of entities. Span-based methods are usually confused by spans with minor difference. For example, as shown in Figure 2, "B cells" and "EBV - transformed B cells" are entities, while "transformed B cells" is not, which brings great difficulty in learning span representations and usually leads to false-positive errors at inference time.

To alleviate the above-mentioned issues in the span-based method, we propose a **B**oundary **E**nhanced **N**eural **S**pan **C**lassification (**BENSC**) model. In addition to classifying spans into corresponding semantic tags, we propose incorporating an additional boundary detection task to enhance the boundary supervision in learning span representations. Specifically, given a sentence, we first encode the word with the token-level representation, and then jointly train the boundary detection model and the span classification model under a multitask framework. The boundary detection model consists of two token-level classifiers predicting whether each word is the first or last word of an entity respectively. The span classification model is to aggregate the inside information of the span to predict its semantic tag. During inference, we can obtain the boundary confidence scores P_s and P_e via the boundary detection model and the tag C of the span with the confidence score P_{sp} via the span classification model. The three scores will jointly determine whether a span is an entity with the tag C.

We conduct experiments on three standard benchmark datasets. Experimental results show that our approach achieves 75.3, 75.6, and 75.7 scores in terms of F_1 on the ACE2004, ACE2005, and GENIA datasets, respectively. With the pre-trained language model BERT, our approach further improves the result to 85.3, 83.9, and 78.3 on three datasets, which outperforms all existing methods and our span classification baselines. Ablation tests show that jointly learning boundary detection and span classification tasks benefits the model with better representation and it improves both two tasks. In addition, the boundary detection model has the ability to generate high-quality candidate spans, which greatly reduces the number of spans feeding into the span classification model and therefore reduces the time complexity of the whole model. We also show that incorporating the boundary probability can help avoid mistakes by the span classification model through case studies.

Related Work

It has been a long history of research involving named entity recognition (McCallum and Li 2003). Zhou and Su (2002) present a system for recognizing named entities using an HMM-based approach. McDonald and Pereira (2005) apply conditional random fields to recognize the protein and gene entities in biomedical texts. Alex, Haddow, and Grover (2007) propose building models on top of linear-chain conditional random fields for recognizing nested entities in biomedical texts. With the development of deep learning methods, LSTM-CRF achieves very promising results in recognizing named entities (Huang, Xu, and Yu 2015; Lample et al. 2016). However, traditional sequential labeling models cannot handle the nested structure because they can only assign one label to each token.

Finkel and Manning (2009) point out that named entities are often nested. The earliest research efforts on nested NER are rule-based (Zhang et al. 2004). The authors first detect the inner-most mentions and then identify overlapping mentions based on the rule-based post-processing methods. Lu and Roth; Katiyar and Cardie; Wang and Lu (2015; 2018; 2018) propose the hypergraph-based method to solve this problem. They design a hypergraph to represent all possible nested structures, which guarantees that nested entities can be recovered from the hypergraph tags. However, the hypergraph needs to be carefully designed to avoid spurious structures and structural ambiguities, and inevitably leads to higher time complexity during both training and inference. In addition, Muis and Lu (2017) develop a gap-based tagging schema to capture nested structures. Wang et al. (2018) propose a transition-based method to construct nested mentions via a sequence of specially designed actions. Fisher and Vlachos (2019) propose forming nested structures by merging tokens and/or entities into entities for entity representation. Lin et al. (2019) propose a sequence-to-nuggets architecture that first identifies anchor words with corresponding semantic types of all entities, and then recognizes the boundaries of the entity for each anchor word.

Another strategy for the nested NER problem is the span-based methods. In the span-based method, nested entities can be easily detected because they belong to different subsequences. Recently, Xu, Jiang, and Watcharawittayakul (2017) try to directly classify all subsequences of a sentence by encoding each subsequence into a fixed-size representation. Sohrab and Miwa (2018) also enumerate all possible regions or spans as potential entity mentions and classify them with deep neural networks. Xia et al. (2019) propose MGNER that consists of a Detector that examines all possible spans and a Classifier that categorizes spans into corresponding semantic tags. Luan et al. (2019) propose a general framework that leverages the coreference and relation type confidences for better span representations. These approaches are straightforward for nested mention detection, but have two main drawbacks. First, classifying all subsequences in the sentence need high computational cost. Second, compared with methods based on the sequence labeling, span-based methods usually show worse performance in determining the boundary of an entity because of less supervision in boundary detection. Our work is also under the span classification framework. To alleviate the above-mentioned issues, we propose incorporating boundary detection into span classification, which can help model learn better representation with boundary supervision and reduce the time complexity by generating high-quality candidates[1].

Approach

Following the overview in Figure 3, our approach consists of two parts as boundary detection and span classification. The boundary detection part aims to predict whether a word is the first or last word of an entity. The span classification part aims to classify spans to corresponding semantic tags.

[1]We observe a contemporaneous work that leverages entity boundaries to predict entity categorical labels (Zheng et al. 2019).

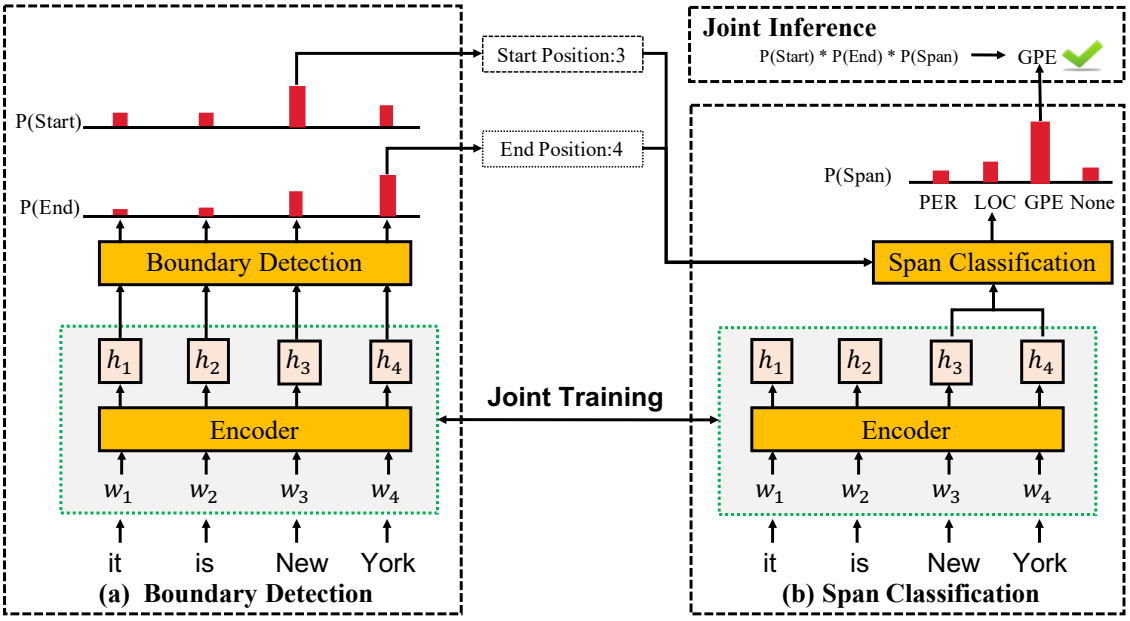

Figure 3: An overview of our proposed approach. The whole model consists of (a) boundary detection to predict the span's boundary (b) span classification to predict the semantic tag of the span. The two parts are jointly trained under the multitask learning framework, and jointly determine the final result.

The two parts are jointly trained under a multitask learning framework. Specifically, we first apply an encoder to the sequence for the contextual word representation. This representation will be shared in the downstream boundary detection and span classification tasks. The boundary detection model consists of two token-level classifiers that predict the probabilities of a word being the start or end words of an entity respectively. The span classification model is to aggregate the span information for the multi-class classification. During inference, we will make the decision by jointly considering the boundary probability and the tag probability.

Encoder

Consider a sentence S with words $\{w_i\}_{i=1}^{N}$, we first convert the words to their respective word-level embeddings and contextual embeddings. In this work, we implement two kinds of encoders as LSTM (Hochreiter and Schmidhuber 1997) and BERT (Devlin et al. 2019) respectively. For LSTM encoder, we first convert the words to their respective word-level embeddings, character-level embeddings, and part-of-speech embeddings. The character-level embeddings are generated by taking the final hidden states of a bi-directional LSTM applied to embeddings of characters in the token. We then use a bi-directional LSTM to produce new representation $h_1, \ldots, h_N$ of all words.

$$x_i = [w_i; char_i; pos_i] \quad (1)$$

$$\overrightarrow{h_i} = \text{LSTM}(x_i, \overrightarrow{h_{i-1}}) \quad (2)$$

$$\overleftarrow{h_i} = \text{LSTM}(x_i, \overleftarrow{h_{i+1}}) \quad (3)$$

$$h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}] \quad (4)$$

For BERT encoder, we first tokenize the sentence with the wordpiece vocabulary, and then generate the input sequence $\tilde{\mathbf{w}}_\mathbf{i}$ by concatenating a `[CLS]` token, the tokenized sentence, and a `[SEP]` token. Next, we use a series of L stacked Transformer blocks (Vaswani et al. 2017) to project the input embeddings into a sequence of contextual vectors.

$$h_i = \text{Transformer Block}_L(\tilde{\mathbf{w}}_\mathbf{i}) \quad (5)$$

We do not combine the character embedding and part-of-speech embedding because we assume that they have already been encoded in the BERT representation.

Boundary Detection

Boundary detection aims to identify whether a word is the first or last word of an entity. Instead of detecting the boundary via sequence labeling methods, we predict the start and end positions with two token-wise classifiers.

Specifically, we feed the contextual representation h_i into a multi-layer perceptron (MLP) classifier, and apply a softmax layer to obtain the probability P_s^i of the word w_i being the first word of an entity.

$$P_s^i = \text{softmax}(\text{MLP}_{\text{start}}(h_i)) \quad (6)$$

Similarity, we can apply a MLP classifier to obtain the probability P_e^i of the word w_i being the last word of an entity.

$$P_e^i = \text{softmax}(\text{MLP}_{\text{end}}(h_i)) \quad (7)$$

During training, since each sentence may contain multiple entities, we label the span boundaries of all entities as the ground-truth. Then, we define the training objective function

as the sum of two following cross-entropy losses in detecting the start and end boundaries, respectively,

$$\mathcal{L}^s_{bdr} = -\sum_{i=1}^{N}[y^i_s \log P^i_s + (1 - y^i_s)\log(1 - P^i_s)] \quad (8)$$

$$\mathcal{L}^e_{bdr} = -\sum_{i=1}^{N}[y^i_e \log P^i_e + (1 - y^i_e)\log(1 - P^i_e)] \quad (9)$$

$$\mathcal{L}_{bdr} = \mathcal{L}^s_{bdr} + \mathcal{L}^e_{bdr} \quad (10)$$

where y^i_s and y^i_e denote the label that whether the word i is the first and last word of an entity, respectively.

Span Classification

Span classification is a span-wise classifier, which aims to classify spans to corresponding semantic tags. If a span is not an entity, it should be mapped to an additionalNone.

We propose to summarize the word representation from contextual word vectors according to its span boundary. For the LSTM encoder, we calculate a summarized vector v_{sp} using the attention mechanism (Bahdanau, Cho, and Bengio 2014) over tokens in its corresponding boundary (i, j),

$$\alpha = \mathrm{softmax}(\mathbf{W}h_{i:j}) \quad (11)$$

$$v_{sp} = \sum_{t=i}^{j} \alpha_t h_t \quad (12)$$

where $\mathbf{W}$ is the parameter to be learned.

For BERT encoder, we obtain the span representation by the mechanism of self attentions. For the span $\{w_i, \ldots, w_j\}$, we use Transformer Blocks (Vaswani et al. 2017) to further encode words inside the span based on their word representations $(h_i, \ldots, h_j)$,

$$h^*_{i:j} = \text{Transformer Blocks}(h_{i:j}) \quad (13)$$

We then use $v_{sp} = [h^*_i, h^*_j]$ to represent the span.

Next, we feed the span representation v_{sp} into a multilayer perceptron (MLP) classifier, and apply a softmax layer to obtain the probability P_{sp} to predict its semantic tag.

$$P_{sp} = \mathrm{softmax}(\mathrm{MLP_{sp}}(v_{sp})) \quad (14)$$

Finally, we minimize the following cross-entropy loss function,

$$\mathcal{L}_{sp} = -\sum_{t=1}^{k}(y^t_{sp} \log P^t_{sp} + (1 - y^t_{sp})\log(1 - P^t_{sp})) \quad (15)$$

where k is the number of semantic tags, and y^t_{sp} denotes a label that whether the span $(w_i, \ldots, w_j)$ is in tag t.

Joint Training and Inference

For training, we jointly minimize the following loss,

$$\mathcal{L} = w\mathcal{L}_{bdr} + (1 - w)\mathcal{L}_{sp} \quad (16)$$

where w is the hyper-parameter to balance two sub-tasks.

During inference, given the instance $(w_i, \ldots, w_j)$, we first obtain the boundary probabilities P^i_s and P^j_e predicted by the boundary detection model. We then classify all legal spans where j must be larger than i if $P^i_s * P^j_e$ is larger than the threshold pre-selected on the development set. We further feed the span into the span classification model for its semantic tag C with probability P_{sp}. If the score $P^i_s * P^j_e * P_{sp}$ is still larger than the threshold, we recognize the span as an entity with tag C.

Implementation Details

For the LSTM encoder, we use 300-dimensional uncased pre-trained *GloVe* embeddings (Pennington, Socher, and Manning 2014) without update during training. We use zero vectors to represent all out-of-vocabulary words. The size of character embedding and part-of-speech embedding are set to 50. The hidden vector length is set to 150. The model is optimized using Adam (Kingma and Ba 2014) with the learning rate of 0.002.

For BERT encoder, we use the $\mathrm{BERT_{BASE}}$ model (Devlin et al. 2019) to obtain the word representation, and the parameter in BERT is also trainable. The hidden vector length is 768 and the number of heads is 12. Detailed model size is referred to Devlin et al. (2019). We use Adam optimizer with the learning rate of 3e-5.

In addition, we also apply 0.2 dropout (Srivastava et al. 2014) between layers. w is set to 0.5 for both the LSTM and BERT encoder. To speed up the process of training, we only sample part of the negative spans from all subsequence. The rule is that the length of the span is less than 6 and negative spans should overlap with positive spans. We observe that training with this negative subset has no performance degradation compared with using full negative spans.

Experiments

We conduct experiments on three standard benchmark datasets as ACE2004, ACE2005, and GENIA respectively. Results show that our proposed approach achieves state-of-the-art performance on all three datasets. The ablation tests show that our multitask learning framework benefits both boundary detection and span classification tasks. Then, the analysis of the time complexity shows that the boundary detection model can generate high-quality candidates, which greatly reduces the time complexity to linear time. Finally, we analyze how the boundary detection model benefits the final result with two cases.

Datasets

We evaluate our model on the ACE2004, ACE2005 (Doddington et al. 2004), and GENIA (Kim et al. 2003) datasets. Specifically, there are seven different types of entities as 'FAC', 'LOC', 'ORG', 'PER', 'WEA', 'GPE', 'VEH' in the ACE datasets and five types of entities as 'G#DNA', 'G#RNA', 'G#protein', 'G#cell_line', 'G#cell_type' in the GENIA dataset. The statistics of these datasets are shown in Table 1. We observe that the statistics are not strictly consistent with previous works in the ACE datasets due to word tokenization and sentence segmentation, but the difference is less than 0.35%.

	ACE2004			ACE2005			GENIA		
	Train	Dev	Test	Train	Dev	Test	Train	Dev	Test
# sentences	7,078	859	922	7,194	969	1,047	14,836	1,855	1,855
with nested entities	2,691	290	377	2,691	338	330	3,199	362	448
# entities	22,172	2,510	3,024	24,441	3,200	2,993	46,473	5,014	5,600
# nested entities	10,080	1,086	1,410	9,389	1,112	1,118	8,337	903	1,217
avg length	20.38	20.69	20.96	19.21	18.93	17.19	30.13	29.17	30.48

Table 1: Statistics of ACE2004, ACE2005, and GENIA datasets.

	ACE2004			ACE2005			GENIA		
Model	P	R	F_1	P	R	F_1	P	R	F_1
LSTM-CRF (Lample et al. 2016)	71.3	50.5	58.3	70.3	55.7	62.2	75.2	64.6	69.5
Multi-CRF	-	-	-	69.7	61.3	65.2	73.1	64.9	68.8
FOFE(c=6) (Xu et al. 2017)	68.2	54.3	60.5	76.5	66.3	71.0	75.4	67.8	71.4
FOFE(c=n) (Xu et al. 2017)	57.3	46.8	51.5	76.9	62.0	68.7	74.0	65.5	69.5
Transition (Wang et al. 2018)	74.9	71.8	73.3	74.5	71.5	73.0	78.0	70.2	73.9
Cascaded-CRF (Ju et al. 2018)	-	-	-	74.2	70.3	72.2	78.5	71.3	74.7
MH (Lu and Roth 2015)	70.0	56.9	62.8	66.3	59.2	62.5	-	-	-
LH (Katiyar and Cardie 2018)	73.6	71.8	72.7	70.6	70.4	70.5	79.8	68.2	73.6
SH(c=6) (Wang and Lu 2018)	79.1	67.3	72.7	75.9	70.0	72.8	76.8	71.8	74.2
SH(c=n) (Wang and Lu 2018)	77.7	72.1	74.5	76.8	72.3	74.5	77.0	73.3	75.1
ARNs (c=6) (Lin et al. 2019)	-	-	-	75.2	72.5	73.9	75.2	73.3	74.2
ARNs (c=n) (Lin et al. 2019)	-	-	-	76.2	73.6	74.9	75.8	73.9	74.8
Merge and Label (Fisher and Vlachos 2019)	-	-	-	75.1	74.1	74.6	-	-	-
BENSC (LSTM)	78.1	72.8	**75.3**	77.1	74.2	**75.6**	78.9	72.7	**75.7**
with Pretrained LM									
MGNER (ELMo) (Xia et al. 2019)	81.7	77.4	79.5	79.0	77.3	78.2	-	-	-
Merge and Label (ELMo)	-	-	-	79.7	78.0	78.9	-	-	-
Merge and Label (BERT)	-	-	-	82.7	82.1	82.4	-	-	-
BENSC (BERT)	85.8	84.8	**85.3**	83.8	83.9	**83.9**	79.2	77.4	**78.3**

Table 2: Overall results on ACE2004, ACE2005, and GENIA datasets.

Baselines

We compare our model with the following baseline models:

LSTM-CRF is a classical baseline for NER, which cannot solve the problem of nested entities (Lample et al. 2016).

Multi-CRF is similar to LSTM-CRF but learns one model for each entity type.

FOFE is a span-based approach that classifies over all subsequences of a sentence by encoding each span with a fixed-size ordinarily forgetting encoding (Xu, Jiang, and Watcharawittayakul 2017).

Transition is a shift-reduce based system that learns to construct the nested structure in a bottom-up manner through an action sequence (Wang et al. 2018).

Cascaded-CRF applies several stacked CRF layers to recognize nested entities at different levels in an inside-out manner (Ju, Miwa, and Ananiadou 2018).

MH makes use of hypergraphs for recognizing overlapping entities (Lu and Roth 2015).

LH uses an LSTM model to learn features and then decodes them into a hypergraph (Katiyar and Cardie 2018).

SH improves LH by considering the transition between labels to alleviate labeling ambiguity (Wang and Lu 2018).

ARNs first identifies anchor words and then recognizes the mention boundaries for each anchor word. They propose a bag-loss to jointly train the two parts (Lin et al. 2019).

MGNER first applies the Detector to generate possible spans as candidates and then applies a Classifier for the entity type (Xia et al. 2019).

Merge and Label first merges tokens and/or entities into entities forming nested structures, and then labels entities to corresponding types (Fisher and Vlachos 2019).

Main Results

Table 2 shows the overall results on ACE2004, ACE2005, and GENIA datasets. Our BENSC model achieves state-of-the-art results in both the LSTM and BERT settings. When using the LSTM encoder, our BENSC model achieves 75.3, 75.6, and 75.7 scores in terms of F_1 on the ACE2004, ACE2005, and GENIA datasets, respectively. Compared with the span-based method FOFE, our BENSC model achieves 14.8, 4.6, and 4.3 absolute gains on the ACE2004, ACE2005, and GENIA datasets, respectively. Our model

	ACE2004			ACE2005			GENIA		
	P	R	F_1	P	R	F_1	P	R	F_1
Boundary Detection (Start)									
Only Boundary Detection	92.0	92.8	92.4	89.7	93.8	91.7	85.9	84.9	85.4
BENSC	92.8	92.5	**92.6**	90.0	94.1	**92.0**	86.5	85.9	**86.2**
Boundary Detection (End)									
Only Boundary Detection	92.3	92.5	92.4	88.8	93.3	91.0	88.8	87.8	88.3
BENSC	92.1	93.8	**92.9**	89.2	94.1	**91.6**	89.0	89.4	**89.2**
Span Classification									
Span Classification Only	75.1	87.4	80.8	73.2	85.7	79.0	72.1	81.4	76.5
BENSC	85.8	84.8	**85.3**	83.8	83.9	**83.9**	79.2	77.4	**78.3**

Table 3: Ablation tests on ACE2004, ACE2005, and GENIA datasets using the BERT encoder.

also outperforms the hypergraph-based methods LH and SH, and the other state-of-the-art methods such as the ARNs model and the Merge and Label model. With the pre-trained language model, our BENSC model with the BERT encoder achieves 85.3, 83.9, and 78.3 scores in terms of F_1 on the ACE2004, ACE2005, and GENIA datasets, respectively, which outperforms all existing baselines such as the MGNER model and the Merge and Label model.

Ablation Test

To analyze the effectiveness of our joint model, we show the result of ablation tests based on the BERT encoder in Table 3. We observe that jointly training the boundary detection model and the span classification model can improve the result of both two tasks. Firstly, we observe that the result of predicting the start and end boundaries achieves a little improvement compared with the isolate boundary detection model. Then, compared with the original span classification method, incorporating the boundary detection model obtains 4.5%, 4.9%, and 1.8% absolute gains on the ACE2004, ACE2005, and GENIA datasets, respectively. In addition, our BENSC model shows better precision than the original span classification method. As we mentioned before, the span-based model is usually confused when the positive and negative instances have many overlapping words, which may lead to some false-positive errors. However, our BENSC model takes the boundary information as additional supervision, which benefits the model to distinguish the confusing cases.

Time Complexity

Theoretically, given a sentence consisting of N words, there are altogether $\frac{n(n+1)}{2}$ possible candidates. Previous span-based methods need to classify almost all sentence subsequences into corresponding semantic tags, which leads to the high computational cost with $O(mn^2)$ time complexity where m is the number of semantic tags. However, in our work, the boundary detection model can help us generate high-quality candidates, which can significantly reduce the number of candidates and lead to much lower time complexity. The time complexity of our approach consists of two parts. The boundary detection model is a token-wise classification model with $O(n)$ time complexity. The span classification model needs to classify the span to corresponding semantic tags. Its time complexity is determined by the number of candidates. In our experiment, we prune spans whose boundary probability $P_s^i * P_e^j$ are lower than the pre-selected threshold after the boundary detection part since they cannot be triggered whatever the result of the span classification model is. Although in the worst case, every position is marked with both the start and end label, which leads to $\frac{n(n+1)}{2}$ possible candidates, we observe that the number of candidate spans is much closer to the number of entities c in practice. Ideally, for c entities, the model will detect c start positions and c end positions, which may form c^2 candidates. However, as shown in Figure 1, nested entities may share the same start or end positions. The actual number of candidates is therefore much less than c^2 and closer to c. As shown in Table 4, the average number of candidates in our experiments is 4.80, 4.19 and 3.56 in the ACE2004, ACE2005, and GENIA dataset, respectively, which is closed to the number of entities and much less than the average length of sequences. For example, on the ACE2005 dataset, we reduce candidates from over 100 thousand subsequences of the sentence to about 4.43 thousand spans that is only 1.5 times than the number of entities. Therefore, the total time complexity of our approach is approximated to $O(n + cm)$ where $c << n$. This analysis demonstrates that adding the boundary detection model can help us generate high-quality candidates to reduce the time complexity to almost linear time in practice.

Dataset	# Entities	# Candidates	# Words
ACE2004	3.28	4.80	20.96
ACE2005	2.86	4.19	17.19
GENIA	3.02	3.56	30.48

Table 4: The statistics of the average number of entities, candidates, and words in the sentence. We can observe that the number of our candidates is far less than the length of sentence.

Case Study

To demonstrate how each module of our model takes effect when predicting the final answer, we conduct a case study

Sentence 1: First , from different cell lines three or all four of the **nuclear proteins** were specifically cross-linked by UV irradiation to the **radioactively labeled TRE-DNA fragment** ..						
Candidate Spans:	$\mathbf{Pred_{sp}}$	**Gold**	$\mathbf{P_s^i}$	$\mathbf{P_e^j}$	$\mathbf{P_{sp}}$	**Output**
nuclear proteins	G#protein	G#protein	1.0	1.0	1.0	G#protein
TRE - DNA fragment	G#DNA	G#DNA	1.0	1.0	1.0	G#DNA
radioactively labeled TRE - DNA fragment	G#DNA	None	0.11	1.0	0.77	None
Sentence 2: reporter : now **willie williams the girl 's father** is qharthd attempted murder						
Candidate Spans:	$\mathbf{Pred_{sp}}$	**Gold**	$\mathbf{P_s^i}$	$\mathbf{P_e^j}$	$\mathbf{P_{sp}}$	**Output**
willie williams	PER	PER	1.0	0.99	0.99	PER
the girl	PER	PER	1.0	1.0	1.0	PER
the girl 's father	PER	PER	1.0	1.0	1.0	PER
willie williams the	PER	None	1.0	0.0	0.58	None
willie williams the girl	PER	None	1.0	1.0	0.76	PER
williams the	PER	None	0.0	0.0	0.54	None
williams the girl	PER	None	0.0	1.0	0.72	None
willie williams the girl 's father	PER	None	1.0	1.0	1.0	PER

Table 5: Case study on GENIA and ACE datasets. In practice, if $P_s^i * P_e^j$ is lower than the pre-selected threshold, we will not feed the span into the span classification model for its semantic tag. However, to analyze the effect of each module, we show its semantic tag $Pred_{sp}$ with corresponding probability P_{sp} in this case study.

in Table 5 with two cases in the GENIA and ACE datasets, respectively. In the first example, we can observe that the correct entity span "nuclear proteins" and "TRE - DNA fragment" obtain high probabilities in all three modules. If only considering the span classification model, the span "radioactively labeled TRE - DNA fragment" may be misidentified as "G#DNA", however, since the boundary detection model gives a lower P_s^i score, we can prune this span because its first word does not look like the start of an entity. In the second example, correct spans are also correctly recognized, and parts of wrong spans such as "willie williams the" and "williams the girl" can be correctly pruned. However, the span "willie williams the girl" and "willie williams the girl 's father" are false-positive errors. Actually, "willie" is the first word of "willie williams" and "girl" as well as "father" are last words of "the girl" as well "the girl 's father" respectively. Since our boundary detection model independently predicts the probability at the word level, it cannot distinguish whether the first word and the last word come from the same entity, and therefore make a mistake to recognize this two spans as entities. An alternative solution is to take the span into consideration when determining the start and end probabilities, but it will lead to higher time complexity.

Conclusion

In this paper, we tackle the problem of nested NER in which entities may be nested with others. We consider that the span-based approach has its advantages as nested entities correspond to different subsequences. However, the span-based method has two main drawbacks as the high time complexity and the weak supervision of the boundary. To overcome the two above issues, we propose a boundary enhanced neural span classification model (**BENSC**), which incorporates the boundary detection task into the span classification task under a multitask learning framework. We first apply an encoder for the token-level representation. On top of it, we implement a boundary detection model with two token-level classifiers to predict whether a word is the first or last word of an entity, and a span classification model to aggregate the span information for the semantic type. During inference, a span treated as an entity should have high probabilities at both the span level and the boundary level. Experiments show that our BENSC model achieves the state-of-the-art results on three standard benchmark datasets and outperforms the span classification baselines. Ablation tests demonstrate that the boundary detection benefits our BENSC model with better representation of the span. In addition, the boundary detection model can generate high-quality candidates, which greatly reduces the time complexity to almost linear time during inference.

Acknowledgments

We greatly thank all anonymous reviewers for their helpful comments. We also thank Kenny Q. Zhu, Xiaozhong Liu, Dingkun Long, Ruixue Ding, Yijia Liu, and Xiang Huang for helpful discussions.

References

Alex, B.; Haddow, B.; and Grover, C. 2007. Recognising nested named entities in biomedical text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, 65–72.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, 4171–4186.

Doddington, G. R.; Mitchell, A.; Przybocki, M. A.; Ramshaw, L. A.; Strassel, S. M.; and Weischedel, R. M. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Lrec*, volume 2, 1.

Finkel, J. R., and Manning, C. D. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, 141–150.

Fisher, J., and Vlachos, A. 2019. Merge and label: A novel neural network architecture for nested ner. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, 5840–5850.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Ju, M.; Miwa, M.; and Ananiadou, S. 2018. A neural layered model for nested named entity recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, 1446–1459.

Katiyar, A., and Cardie, C. 2018. Nested named entity recognition revisited. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, 861–871.

Kim, J.-D.; Ohta, T.; Tateisi, Y.; and Tsujii, J. 2003. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics* 19(suppl_1):i180–i182.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lafferty, J.; McCallum, A.; and Pereira, F. C. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

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*, 260–270.

Lin, H.; Lu, Y.; Han, X.; and Sun, L. 2019. Sequence-to-nuggets: Nested entity mention detection via anchor-region networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 5182–5192.

Lu, W., and Roth, D. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 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 *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, 3036–3046.

McCallum, A., and Li, W. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, 188–191.

McDonald, R., and Pereira, F. 2005. Identifying gene and protein mentions in text using conditional random fields. *BMC bioinformatics* 6(1):S6.

Muis, A. O., and Lu, W. 2017. Labeling gaps between words: Recognizing overlapping mentions with mention separators. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2608–2618.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *EMNLP 2014, A meeting of SIGDAT, a Special Interest Group of the ACL*, 1532–1543.

Sohrab, M. G., and Miwa, M. 2018. Deep exhaustive model for nested named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2843–2849.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, 5998–6008.

Wang, B., and Lu, W. 2018. Neural segmental hypergraphs for overlapping mention recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 204–214.

Wang, B.; Lu, W.; Wang, Y.; and Jin, H. 2018. A neural transition-based model for nested mention recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 1011–1017.

Xia, C.; Zhang, C.; Yang, T.; Li, Y.; Du, N.; Wu, X.; Fan, W.; Ma, F.; and Yu, P. 2019. Multi-grained named entity recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 1430–1440.

Xu, M.; Jiang, H.; and Watcharawittayakul, S. 2017. A local detection approach for named entity recognition and mention detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, 1237–1247.

Zhang, J.; Shen, D.; Zhou, G.; Su, J.; and Tan, C.-L. 2004. Enhancing hmm-based biomedical named entity recognition by studying special phenomena. *Journal of biomedical informatics* 37(6):411–422.

Zheng, C.; Cai, Y.; Xu, J.; Leung, H.-f.; and Xu, G. 2019. A boundary-aware neural model for nested named entity recognition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 357–366.

Zhou, G., and Su, J. 2002. Named entity recognition using an hmm-based chunk tagger. In *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 473–480.