Nested Named Entity Recognition with Span-level Graphs

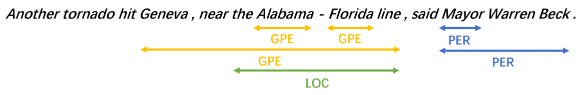
Juncheng Wan, Dongyu Ru, Weinan Zhang,[*] Yong Yu
Shanghai Jiao Tong University
{junchengwan,maxru,wnzhang,yyu}@apex.sjtu.edu.cn

Abstract

Span-based methods with the neural networks backbone have great potential for the nested named entity recognition (NER) problem. However, they face problems such as degenerating when positive instances and negative instances largely overlap. Besides, the generalization ability matters a lot in nested NER, as a large proportion of entities in the test set hardly appear in the training set. In this work, we try to improve the span representation by utilizing retrieval-based span-level graphs, connecting spans and entities in the training data based on n-gram features. Specifically, we build the entity-entity graph and span-entity graph globally based on n-gram similarity to integrate the information of similar neighbor entities into the span representation. To evaluate our method, we conduct experiments on three common nested NER datasets, ACE2004, ACE2005, and GENIA datasets. Experimental results show that our method achieves general improvements on all three benchmarks (+0.30 $\sim$ 0.85 micro-F1), and obtains special superiority on low frequency entities (+0.56 $\sim$ 2.08 recall).

1 Introduction

Named entity recognition is one of the major subtasks of information extraction for extracting categorized named entities from unstructured text. Recently, neural-based NER architectures have shown remarkable performance with minimal feature engineering, such as CNN-CRF (Collobert et al., 2011), BiLSTM+CRF (Lample et al., 2016), LSTM-CNN-CRF (Ma and Hovy, 2016) and Lattice LSTM (Zhang and Yang, 2018a). Despite their great success, nested NER raises new challenges due to the deeply overlapping or nested entities (Finkel and Manning, 2009). In nested NER, a token may be included in multiple entities (Wang and Lu, 2018) instead of a single one in the conventional setting, making the problem more difficult to solve.


Another tornado hit Geneva , near the Alabama - Florida line , said Mayor Warren Beck .
GPE
GPE
PER
GPE
PER
LOC
Figure 1: An example of nested NER in ACE2005 dataset.

Previous exploration on nested NER can be mainly divided into three categories, using various architectures or different formulation for adaptation to the nested scenario. *Hypergraph-based* methods use explicit hypergraphs to represent the possible nested structure or investigate the graph-shaped lexical/syntactic features (Lu and Roth, 2015; Katiyar and Cardie, 2018; Wang and Lu, 2018). *Layered-based* methods construct the nested structure through an action sequence (Wang et al., 2018; Fisher and Vlachos, 2019; Shibuya and Hovy, 2020) or layered-models (Ju et al., 2018; Wang et al., 2020). *Span-based* methods directly enumerate spans in a sentence, and perform categorical prediction on each span (Lin et al., 2019; Eberts and Ulges, 2020; Luan et al., 2019; Tan et al., 2020). *Span-based* methods adopt the most simple and straightforward formulation as span classification, thus widely used and applied in joint relation extraction recently.

Despite the simplicity of span-based models, they can hardly fully utilize the rich semantics in spans. Previous investigation has shown that span-based models are usually confused when the positive and negative instances are largely overlapped (Finkel and Manning, 2009; Tan et al., 2020), as shown in Fig. 1. The minor differences between long entities and their similar spans can easily fool the span-based models. Besides, most entities during inference statistically never appear in the training set in nested NER. For example, in ACE2004, ACE2005, and GENIA, there are

[*]Corresponding author

53.06%, 41.64% and 51.42% entity mentions from the validation set appear fewer than three times in the training set. Learning powerful span representations for the prediction of those "unfamiliar" entities is difficult for conventional span-based models.

In this work, we try to improve the span representation in span-based methods by utilizing retrieval-based span-level graphs. We seek helpful information in the training set beyond the current sentence. The intuitive assumption is that the entity spans similar to the candidate spans contain related information for discrimination on the candidate spans. We use n-gram similarity to measure the distance between spans. Specifically, we treat each entity in the training set and each raw span as nodes, connecting those with high n-gram similarity. The constructed span-level heterogeneous graph records the lexical correlations among entities and various raw spans. We enhance the span representation by including the retrieved local subgraph for feature extraction of a specific span. We perform message passing with GCNs (Kipf and Welling, 2017) on the retrieved subgraph for neighbor entities representation. The representation of neighbor entities provides rich correlations beyond the current sentence, thus improving the performance on confusing long spans and low-frequency spans. Our main contributions are listed as follows:

- We firstly introduce retrieval-based span-level graphs in nested NER to model the lexical correlations among candidate spans and entities beyond the current sentence.

- We perform message passing with GCNs and conduct multitask learning to effectively extract the rich information from the entity neighbors of candidate spans.

- We conduct experiments on three common nested NER datasets (ACE2004, ACE 2005, and GENIA). The empirical results and extensive analysis show that our method outperforms strong baselines on all three benchmarks and has special superiority on long and low-frequency spans.

2 Related Work

Our work is closely related to nested NER and graphs used in NER. We introduce them accordingly as below.

2.1 Nested NER

Nested NER, with the challenging nested entities included, has attracted many researchers recently. There are three categories of mainstream solutions: span-based, hypergraph and layered methods.

Span-based method exhausts all spans in the sequence and predicts their classes instead of performing sequence labeling. The classifier can be a maximum entropy tagger (Byrne, 2007) or a neural network (Sohrab and Miwa, 2018; Fu et al., 2021; Ouchi et al., 2020; Li et al., 2020). Some works propose to utilize relations (Luan et al., 2019; Eberts and Ulges, 2020) for span prediction or correlation intensities (Xu et al., 2021), while Tan et al. (2021) proposes to learn the patterns of the valuable spans. The span-based method is also improved by the two-stage method, including locating entities and predicting type (Lin et al., 2019; Tan et al., 2020; Zheng et al., 2019; Shen et al., 2021). Our method abuses complicated tagger or the boundary auxiliary but utilizes the n-gram features to improve the span representation.

Hypergraph method learns hypergraph of nested relationship (Lu and Roth, 2015), dependency tree (Yu et al., 2020) or others (Dozat and Manning, 2017; Muis and Lu, 2017; Wang and Lu, 2018; Katiyar and Cardie, 2018). This method with the designed proprietary structures can explicitly capture the nested entities. While a proper graph needs subtle work or external tools, such as the parser.

Layered method stacks flat NER layers and is naturally suitable for nested structures. While it suffers from layer disorientation or error propagation problem. Ju et al. (2018) recognizes inner entities and then entities of next layers. Others enhance this idea with a merge and label method (Fisher and Vlachos, 2019) or by applying a pyramid-shaped decoder (Wang et al., 2020). The second-best path decoding method is explored (Shibuya and Hovy, 2020) and improved (Wang et al., 2021) by excluding the influence of the best path.

2.2 Graphs Used in NER

Graphs, as a common formulation for structured information, are widely used in flat NER and nested NER. For Chinese NER, models with lexicon-based graph (Zhang and Yang, 2018b; Ding et al., 2019; Gui et al., 2019) are proposed to fully use gazetteers. Cetoli et al. (2017) investigates the use of the dependency tree. Yu et al. (2020) improves the idea from graph-based dependency parsing via

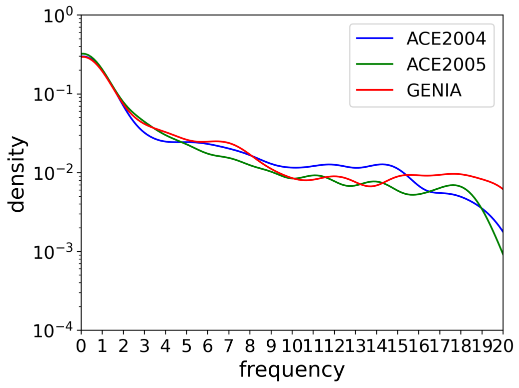


Figure 2: The density of entity frequency in the test set, where entities have frequency ≤ 20 in the training set.

a biaffine model. The biaffine model is also explored with the graph of original token sequence and the graph of tokens in recognized entities (Luo and Zhao, 2020). Muis and Lu (2017); Wang and Lu (2018); Katiyar and Cardie (2018) resolve spurious structures and ambiguous issues of hypergraph structure of nested NER. Relation information is used in the graph (Fu et al., 2019) with a relation extraction model. Instead of building graphs from a single sentence, our span-level graphs are built from the whole training set, which has better data utilization. Besides, our method does not use the parser for dependency tree or gazetteers as external knowledge.

3 Our Approach

In this part, we introduce the details of our approach. We first formulate the target problem, nested NER, as follows.

Nested NER as Span Classification Following Eberts and Ulges (2020), we formulate the nested NER problem as the span classification task. The span classification task treats multiple adjacent tokens as a span and predicts the corresponding label. Specifically, for a sentence $X = \{x_1, \ldots, x_n\}$ of n tokens, we extract all spans (with length ≤ 10) to a span set $S_X = \{s_{ij} | 1 \leq i \leq j \leq n\}$, where s_{ij} indicates the span from x_i to x_j. We predict the corresponding label of s_{ij} as one of the pre-defined entity types or NA (not an entity).

The formulation as span classification instead of sequence labeling is more suitable for NER in nested scenarios but brings two challenges. The *insensitivity to boundaries* make the long candidate spans hard to identify. The *low-frequency spans*, taking the majority in data (Fig. 2), also increase the difficulty on capturing intra-span representations. We use span-level graphs connecting entities and raw spans to tackle the problems.

The overview of our model architecture is shown in Fig. 3. We construct the span-level graph with n-gram similarity to determine the adjacency (Sec. 3.1). We initialize the representation of spans and entities with the encoder described in Sec. 3.2. The structured correlations among entities and spans are modeled with GCN (Sec. 3.3). We incorporate in training the entity categorical prediction to utilize the label of entity mentions on the graph (Sec. 3.4).

3.1 Span-level Graph

We propose to improve the span representation by constructing retrieval-based graphs according to n-gram features. Our method uses two span-level graphs, i.e. entity-entity graph and span-entity graph. If treating each entity mention or raw span as a span of multiple adjacent tokens, both of these two graphs model the relationship between spans.

Before describing the method, we denote $\mathcal{E}$ the set of entity mentions, $\mathcal{R}$ the set of raw spans, $\mathcal{S} = \mathcal{E} \bigcup \mathcal{R}$ the set of all spans, and $\mathcal{N}_G^k(v)$ the set of k hop neighborhood vertices of vertex v in graph G. In this work, we design the n-gram similarity function between spans $f_n : \mathcal{S} \times \mathcal{S} \longrightarrow \mathbb{R}$ at byte pair encodings (BPE) (Sennrich et al., 2016) level. More precisely, f_n is the cardinality of the intersection set of n-gram BPE sets, i.e. $f_n(s, s') = |n\text{-gram}(\texttt{BPE}(s)) \cap n\text{-gram}(\texttt{BPE}(s'))|$ for $s, s' \in \mathcal{S}$.

Entity-entity graph First, we introduce the entity-entity graph $G_{EE} = (V_{EE}, E_{EE})$. In G_{EE}, nodes are from the set of entity mentions $\mathcal{E}$. Entity mentions with the same tokens but different types are treated as different nodes. For $e_i, e_j \in V_{EE}$, the edge weight $w(e_i, e_j)$ is calculated by the weighted n-gram similarity from f_n as follows:

$$w(e_i, e_j) = \frac{1}{N} \sum_{n=1}^{N} \alpha_n f_n(e_i, e_j) \qquad (1)$$

where α_n indicates the importance of each n-gram feature, and N is the largest gram length. A high value of $w(e_i, e_j)$ indicates high frequency of the words co-occurrence between e_i and e_j.

Span-entity graph Span-entity graph $G_{SE} = (V_{SE}, E_{SE})$ models the relationship between raw

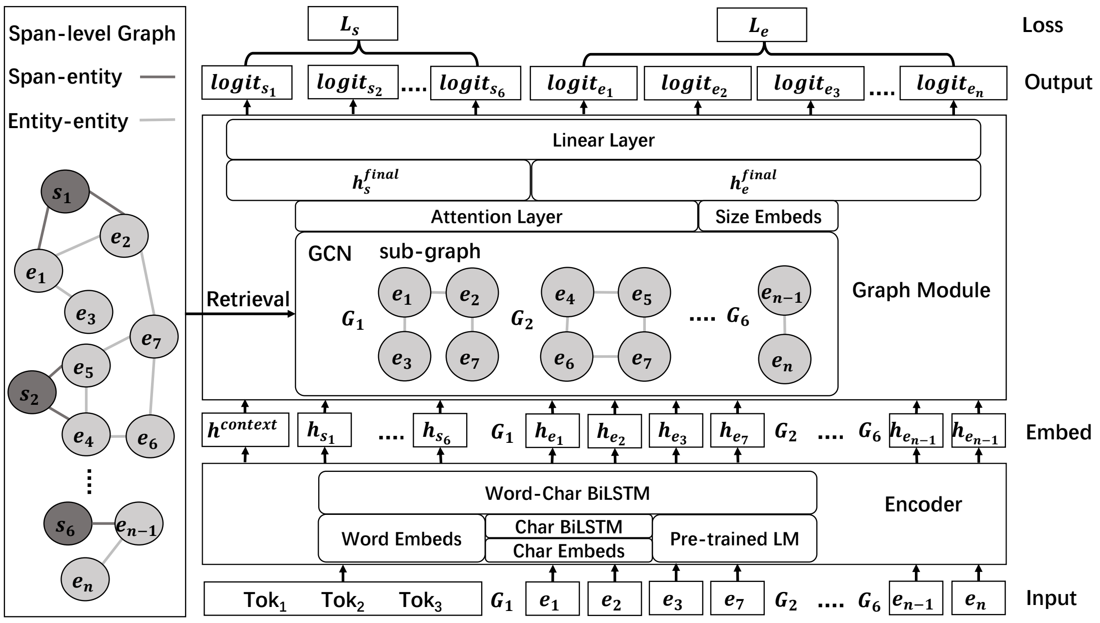

Span-level Graph
Span-entity
Entity-entity
s_1 e_2 e_1 e_3 e_7 e_5 s_2 e_4 e_6 s_6 e_{n-1} e_n
L_s
L_e
Loss
$logit_{s_1}$ $logit_{s_2}$ .... $logit_{s_6}$ $logit_{e_1}$ $logit_{e_2}$ $logit_{e_3}$ .... $logit_{e_n}$
Output
Linear Layer
h_s^{final}
h_e^{final}
Attention Layer
Size Embeds
GCN sub-graph
Retrieval
G_1 G_2 .... G_6
Graph Module
$h^{context}$ h_{s_1} .... h_{s_6} G_1 h_{e_1} h_{e_2} h_{e_3} h_{e_7} G_2 .... G_6 $h_{e_{n-1}}$ $h_{e_{n-1}}$
Embed
Word-Char BiLSTM
Word Embeds
Char BiLSTM
Char Embeds
Pre-trained LM
Encoder
Tok$_1$ Tok$_2$ Tok$_3$ G_1 e_1 e_2 e_3 e_7 G_2 .... G_6 e_{n-1} e_n
Input

Figure 3: An overview of our model. The sub-graph takes 2-hop neighbors of raw spans.

spans and entity mentions, which is more complicated. In G_{SE}, nodes include raw spans and entity mentions from $\mathcal{S}$ and each edge connects one raw span and one entity mention. By observation, the gram features of raw spans in natural sentences are different from that of entities in two aspects. (1) A raw span can be arbitrary long, as it can be a single token to the whole sentence. A raw span of the whole sentence unfairly has more gram overlap with entity mentions than other spans within it. (2) Raw spans have more irregular patterns than entities and always link to meaningless entity mentions as noise.

Thus, constructing G_{SE} as G_{EE} is not suitable. Here we propose two simple and effective methods for these problems. For problem (1), we penalize long raw spans s_{ij} for any edge weight $w(s_{ij}, e)$ by the length $l(s_{ij}) = j - i + 1$ as follows:

$$w(s_{ij}, e) = \frac{1}{N \cdot l(s_{ij})} \sum_{k=1}^{N} \alpha_n f_n(s_{ij}, e) \qquad (2)$$

For problem (2), we exclude those noise span-entity edges simply by setting the hard threshold $\tau \in \mathbb{R}^+$ and remove edges with weight below it.

Span-level (sub-)graph The span-level graph $G = (V, E)$ is the union of G_{EE} and G_{SE} excluding raw spans. We exclude raw spans for the training efficiency of a homogeneous graph. Thus, $V = V_{EE} \bigcup V_{SE} - \mathcal{R}$ and $E = E_{EE}$. For mini-batch training, we dynamically extract span-level sub-graphs from G_{EE} and G_{SE}. As the inference goal is to classify raw spans, we only extract the K-hop sub-graph of raw spans during training.

The extraction process is as follows. (1) For the raw span node v_s, we take its first-order neighbors $V_1 = \mathcal{N}^1_{G_{SE}}(v_s)$ from G_{SE}. (2) The union of i-hop $(1 \leq i \leq K-1)$ entity neighbors of $\mathcal{N}^1_{G_{SE}}(v_s)$, i.e. $V_2 = \bigcup_{v \in \mathcal{N}^1_{G_{SE}}(v_s)} \bigcup_{1 \leq i \leq K-1} \mathcal{N}^i_{G_{EE}}(v)$, are extracted from G_{EE}. (3) We exclude the raw span node v_s and preserve edges between the rest nodes. Thus, the sub-graph of v_s is an induced sub-graph $G[V_1 \bigcup V_2 - v_s]$ from G.

3.2 The Encoder

For the initialization of raw spans and entity mentions, we use char embeddings, word embeddings, and pre-trained LM. Both sentence and entity mentions are treated as a sequence of tokens and are encoded separately. First, the char embeddings are fed into bidirectional LSTM (Lample et al., 2016) (Char-BiLSTM) to capture the orthographic and morphological features of words. Then, the pre-trained LM, such as BERT (Devlin et al., 2019), is used for contextualized representation. The representations are averaged BPE embeddings in the last layer. Finally, the char hidden states, contextualized embeddings, and word embeddings are concatenated and then fed into another bidirectional LSTM (Word-Char BiLSTM) for the encoded representation of words. For the span-level representation, we use max-pooling for encoded representation of words within the span.

3.3 Graph Module

To model the span-level graph, we adopt graph convolutional networks (GCN) (Kipf and Welling, 2017). Let A be the normalized symmetric adjacency matrix of G. The number of GCN layers is also the hop number K of the sub-graph. The $(k+1)$-th layer of the feature matrix H^{k+1} is computed as:

$$H^{k+1} = \texttt{ReLU}(AH^kW_k) \tag{3}$$

where W_k is the learnable matrix, $0 \leq k \leq K$, H^0 is the output from the encoder.

To integrate the representations of neighborhood nodes for sub-graph embedding, we use the attention mechanism. Denote h_0^k $(0 \leq k \leq K)$ the hidden state of the raw span in the k-th layer of GCN and h_i^k $(i \geq 1)$ that of the i-th entity mention neighbor of the span in the k-order neighbors. The sub-graph embedding h^{graph} of the raw span is:

$$\gamma_i = \frac{\exp((h_i^K)^T W_a h_0^0)}{\sum_{j \geq 1} \exp((h_j^K)^T W_a h_0^0)} \tag{4}$$

$$h^{\text{graph}} = \sum\nolimits_{i \geq 1} \gamma_i h_i^K \tag{5}$$

where W_a is a learnable matrix.

Besides, context information of entities is important as entities are interpreted differently under different contexts. We use the last hidden state of "[CLS]" token in pre-trained LM as the context representation h^{context}. We use a learnable weight matrix for size embeddings h^{size}.

The raw span representations consists of the encoder output h_0^0, the raw span sub-graph embedding, context embedding, and size embedding. The final representation of a raw span h_0^{final} or a entity mention h_i^{final} $(i \geq 1)$ is as follows:

$$h_0^{\text{final}} = \texttt{concat}(h_0^0, h^{\text{graph}}, h^{\text{context}}, h^{\text{size}}) \tag{6}$$

$$h_i^{\text{final}} = \texttt{concat}(h_i^0, h_i^K, h^{\text{size}}) \tag{7}$$

3.4 Multitask Learning

To utilize the label of entity mentions, we force GCN to predict graph neighbors of the raw span simultaneously. We use feed-forward layers to get logits as follows:

$$logits_s = \texttt{Linear}_s(h_0^{\text{final}}) \tag{8}$$

$$logits_{e_i} = \texttt{Linear}_e(h_i^{\text{final}}) \tag{9}$$

Thus, our algorithm contains two losses to minimize, the cross entropy loss $\mathcal{L}_s$ ($\mathcal{L}_e$) of raw span (entity mention) prediction.

$$\mathcal{L}_s = \texttt{CE}(logits_s) \tag{10}$$

$$\mathcal{L}_e = \sum\nolimits_i \texttt{CE}(logits_{e_i}) \tag{11}$$

To balance the two losses, we adopt a multi-task learning framework with a hyperparameter β:

$$\mathcal{L} = \mathcal{L}_s + \beta \mathcal{L}_e \tag{12}$$

where $\mathcal{L}$ is the total loss. At the inference stage, we only infer the raw span.

4 Experiments

In this section, we evaluate our method on three common nested NER datasets, including ACE2004, ACE2005, and GENIA.

4.1 Dataset

We use three nested English NER datasets: ACE2004[1], ACE2005[2], and GENIA (Kim et al., 2003). For GENIA, we use GENIAcorpus3.02p3, and follow the train/validation/test split of previous works (Finkel and Manning, 2009; Lu and Roth, 2015) i.e.: (1) split first 81%, subsequent 9%, and last 10% as train, dev and test set, respectively; (2) collapse all DNA, RNA, and protein subtypes into DNA, RNA, and protein, keeping cell line and cell type, and (3) remove other entity types, resulting in 5 entity types. There are statistical results of these datasets in Table 1.

4.2 Baselines

As we use pre-trained LM, we compare our method with methods with similar settings. Besides, we also include the results of models using additional supervision, which are not directly comparable to ours. Our baselines are as follows.

Models without pre-trained LM: Hyper-Graph (Katiyar and Cardie, 2018) proposes a hypergraph-based model based on LSTMs. **Stack-LSTM** (Wang et al., 2018) uses a scalable transition-based method to model the nested structure of mentions. **Seg-Graph** (Wang and Lu, 2018) proposes a segmental hypergraph representation to model overlapping entitys. **ARN**

[1]`https://catalog.ldc.upenn.edu/LDC2005T09`

[2]`https://catalog.ldc.upenn.edu/LDC2006T06`

		ACE2004			ACE2005			GENIA		
		Train	Valid	Test	Train	Valid	Test	Train	Valid	Test
sentence	# total	6,198	742	809	7,285	968	1,058	15,022	1,669	1,855
	# nested (%)	2,718 (43.9%)	294 (39.6%)	388 (48.0%)	2,797 (38.4%)	352 (36.4%)	339 (32.0%)	3,222 (21.4%)	328 (19.7%)	448 (24.2%)
	avg (max) length (words)	21.4 (120)	22.1 (84)	22.0 (91)	18.8 (99)	18.8 (102)	16.9 (76)	26.5 (174)	25.7 (136)	27.1 (123)
entity	# total	22,195	2,514	3,034	24,700	3,218	3,029	47,006	4,461	5,596
	# nested (%)	10,157 (45.8%)	1,092 (43.4%)	1,417 (46.7%)	9,946 (40.3%)	1,191 (37.0%)	1,179 (38.9%)	8,382 (17.8%)	818 (18.3%)	1,212 (21.7%)
	avg (max) length (words)	2.5 (57)	2.6 (35)	2.5 (43)	2.3 (49)	2.1 (31)	2.3 (27)	2.0 (20)	2.2 (20)	2.2 (15)
	(%) low frequency (≤ 3)	-	53.06%	55.08%	-	41.64%	50.08%	-	51.42%	53.97%
word	# total	132,726	16,417	17,822	137,138	18,174	17,909	397,913	42,847	50,182
	avg (max) length (chars)	4.4 (67)	4.4 (18)	4.5 (58)	4.2 (58)	4.2 (19)	4.2 (19)	5.2 (99)	5.2 (36)	5.2 (55)

Table 1: Statistical results of nested NER datasets.

(Lin et al., 2019) leverages the head-driven phrase structures of entity mentions.

Models with pre-trained LM: Seq2seq (Straková et al., 2019) views the nested NER as a sequence-sequence problem. **Path-BERT** (Shibuya and Hovy, 2020) treats the tag sequence as the second-best path within the span of their parent entity. **ML** (Fisher and Vlachos, 2019) proposes a merge and label method. **Pyramid** (Wang et al., 2020) is a layered model, in which text region embeddings are recursively inputted into stacked flat NER layers. **SpERT** (Eberts and Ulges, 2020) is an attention model for span-based joint entity and relation extraction. We implement this method with BERT. **BENSC** (Tan et al., 2020) is a boundary enhanced span classification model.

Models with additional supervision: BERT-MRC (Li et al., 2020) formulates NER as a machine reading comprehension task. **NER-DP** (Yu et al., 2020) uses ideas from graph-based dependency parsing to model nested structure. **DYGIE** (Luan et al., 2019) shares span representations using dynamically constructed span graphs.

4.3 Training Details

For word embeddings, we use 100-dimensional GloVe embeddings trained on 6B tokens [3] for ACE2004/ACE2005, 200-dimensional embeddings trained on biomedical[4] for GENIA. We fix word embeddings during training. We use 30-dimensional char embeddings and Char BiLSTM of 60-dimensional hidden state. The hidden size of Word-Char BiLSTM is 300-dimensional. For size embeddings, we use 25-dimensional vectors.

For pretrained LM, we use BERT-base-cased model (Devlin et al., 2019) [5] for ACE2004 and ACE2005, BioBERT v1.1 (Lee et al., 2020)[6] for GENIA. We fine-tune BERT with optimizer Adam (Kingma and Ba, 2015) with learning rate in $\{1e-5, 2e-5, 3e-5\}$ and weight regularization $1e-8$. For other model parameters, we use learning rate in $\{1e-4, 5e-4, 1e-3\}$. Batch size is in $\{2, 4, 8\}$ and dropout in $\{0.1, 0.2, 0.3\}$. For stable convergence, we use linear learning rate scheduler, with maximal number of epoch 50 and warm-up ratio 0.01.

For the graph, we set the largest gram size $N = 3$ and use BERT-base-cased tokenizer for BPE encoding. The n-gram weight $\alpha_k = 0.5k$, where $k \in \{1, 2, 3\}$. We prune the span-level graph G with $\tau = 0.8$ and edges with weight below it are removed. The GCN layer size $K = 2$ and hidden size is 400. These hyperparameters are selected by ourselves and more detailed analyses are in Appendix A. During training, we sample 100 negative spans randomly. The multi-task coefficient $\beta = 0.1$. We use DGL[7] to implement GCN. The inference speed and GPU usage are discussed in Appendix B.

We pick the model by the performance in the validation set. We use span-level micro-averaged precision, recall, and F1 on the test set for evaluation. Results are averaged on 3 runs for reproducibility.

4.4 Main Results

In Table 2, our method has significant improvement compared with nested NER models without pre-trained LM and with pre-trained LM. Compared with models without pre-trained LM, our method has at least +6.01, +5.69, +1.50 F1 improvement for ACE2004, ACE2005, and GENIA, which is statistically significant. Compared with methods using pre-trained LM, our method also yields at least +0.85, +0.78 F1 improvement for ACE2004 and ACE2005. Besides, our method has

[3]https://nlp.stanford.edu/projects/

[4]https://github.com/cambridgeltl/BioNLP-2016

[5]https://github.com/huggingface/transformers

[6]https://github.com/naver/biobert-pretrained

[7]https://www.dgl.ai

	ACE2004			ACE2005			GENIA		
	P	R	F1	P	R	F1	P	R	F1
Hyper-Graph (Katiyar and Cardie, 2018)	73.60	71.80	72.70	70.60	70.40	70.50	77.70	71.80	74.60
Stack-LSTM (Wang et al., 2018)	-	-	73.30	-	-	73.00	-	-	73.90
Seg-Graph (Wang and Lu, 2018)	78.00	72.40	75.10	76.80	72.30	74.50	77.00	73.30	75.10
BENSC (Tan et al., 2020)	78.10	72.80	75.30	77.10	74.20	75.60	78.90	72.70	75.70
Pyramid (Wang et al., 2020)	81.10	79.40	80.30	80.00	78.90	79.40	78.60	77.00	77.80
ARN (Lin et al., 2019)	-	-	-	76.20	73.60	74.90	75.80	73.90	74.80
ML (Fisher and Vlachos, 2019)	-	-	-	75.10	74.10	74.60	-	-	-
Pyramid-Full (Wang et al., 2020)	81.14	79.42	80.27	80.01	78.85	79.42	78.60	77.02	77.78
with Pre-trained LM									
Seq2seq (BERT) (Straková et al., 2019)	-	-	84.40	-	-	84.33	-	-	78.31
ML (ELMo) (Fisher and Vlachos, 2019)	-	-	-	79.70	78.00	78.90	-	-	-
ML (BERT) (Fisher and Vlachos, 2019)	-	-	-	82.70	82.10	82.40	-	-	-
Path-BERT (Shibuya and Hovy, 2020)	83.73	81.91	82.81	82.98	82.42	82.70	78.07	76.45	77.25
BENSC (BERT-based) (Tan et al., 2020)	85.80	84.80	85.30	83.80	83.90	83.90	79.20	77.40	78.30
Pyramid (BERT-based) (Wang et al., 2020)†	85.41	85.50	85.46	83.39	85.04	84.21	-	-	-
Pyramid (BioBERT-based) (Wang et al., 2020)†	-	-	-	-	-	-	79.63	78.38	79.00
SpERT (BERT-based) (Eberts and Ulges, 2020)‡	85.04	84.33	84.68	82.25	85.31	83.75	-	-	-
SpERT (BioBERT-based) (Eberts and Ulges, 2020)‡	-	-	-	-	-	-	77.24	78.56	77.89
with Additional Supervision									
DYGIE (Luan et al., 2019)	-	-	84.70	-	-	82.90	-	-	76.20
BERT-MRC (Li et al., 2020)	85.05	86.32	85.98	87.16	86.59	86.88	85.18	81.12	83.75
NER-DP (Yu et al., 2020)	87.30	86.00	86.70	85.20	85.60	85.40	81.80	79.30	80.50
Our method (BERT-based)	86.70	85.93	**86.31**	84.37	85.87	**85.11**	-	-	-
Our method (BioBERT-based)	-	-	-	-	-	-	77.92	80.74	**79.30**

Table 2: Comparison of our method with other models on three nested NER datasets. Models with "†" are rerun with released code by ourselves. Models with "‡" are implemented and rerun by ourselves.

Model	P	R	F1
Char + Word + LM Embeds	81.98	85.61	83.75
+ Span-entity Graph	83.35	85.28	84.30(**+0.55**)
+ Entity-entity Graph	84.60	84.68	84.64(+0.34)
+ Multitask Training	84.37	85.87	85.11(+0.47)

Table 3: Abalation study on ACE2005 dataset. We add components to our method step by step.

		ACE2004		ACE2005		GENIA	
Model		base.	ours.	base.	ours.	base.	ours.
Graph usage		×	✓	×	✓	×	✓
Valid	Overall	84.37	85.33(+0.96)	82.85	84.46(+1.61)	79.29	82.16(+2.87)
	Nested	81.78	82.78(**+1.00**)	82.70	84.55(**+1.85**)	66.26	72.13(**+5.87**)
Test	Overall	84.33	85.93(**+1.60**)	85.31	85.87(+0.66)	78.56	80.74(+2.18)
	Nested	83.91	84.84(+0.93)	83.38	84.65(**+2.27**)	66.09	70.13(**+4.04**)

Table 4: Comparison of SpERT and our method on the recall of nested entities on the validation and test sets.

		SpERT. (w/o graph)			**Ours.** (with graph)		
len.	%	P	R	F1	P	R	F1
1	56.32	86.81	87.16	86.98	86.66	88.34	87.49(+0.51)
2	20.96	81.38	83.31	82.33	82.90	85.51	84.19(+0.82)
3	8.19	75.56	81.05	78.21	79.09	83.87	81.41(+0.36)
4	4.62	74.00	79.29	76.55	79.72	81.43	80.57(+4.02)
5	2.97	84.38	90.00	87.10	84.78	86.67	85.71(-1.39)
6	1.65	64.29	72.00	67.92	78.43	80.00	79.21(+11.29)
7	1.02	54.29	61.29	57.58	68.97	64.52	66.67(+9.09)
8	0.83	68.00	68.00	68.00	80.00	64.00	71.11(**+13.11**)
9	0.63	64.71	57.89	61.11	76.47	68.42	72.22(+11.11)
10	0.66	66.67	60.00	63.16	85.71	60.00	70.59(+7.43)

Table 5: Length-wise results on ACE2005 test set to detect the effect of graph module.

comparable results with the Pyramid model in GENIA. Although a little worse than BERT-MRC and NER-DP, our method does not introduce additional supervision, such as the syntax and dependency structures, or human prior knowledge.

4.5 Ablation Study

In Table 3, we conduct an ablation study on the ACE2005 dataset by adding components to our method step by step. It shows that the span-entity graph is the most effective component (+0.55 F1 score). As the first-order sub-graph in the span-level graph, it provides direct guidance for the prediction of the raw span. Besides, multitask training also improves our method by +0.47 F1 score. As multitask training utilizes the label of graph neighbors, which may contain the ground truth of the raw span.

4.6 Recognition of Different Entities

To analyze the benefits of including span-level graphs for extracting span representations, we take studies on the performance of nested entities, enti-

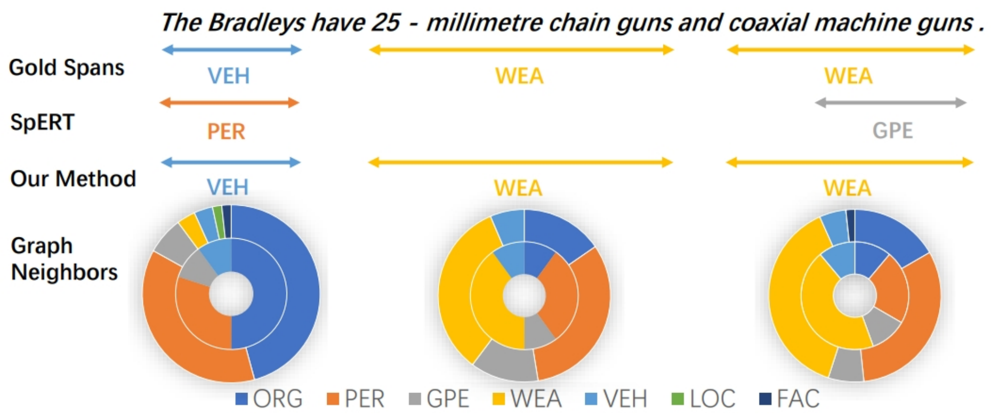


Figure 4: Case study in ACE2005 validation set. "Graph neighbors" presents the statistics of entity types for each raw span. The inner circle represents first-order neighbors and the outer circle represents the whole 2-hop sub-graph.

	ACE2004			**ACE2005**		
freq.	%	SpERT	ours.	%	SpERT	ours.
$= 0$	46.3	77.71	80.06(+2.35)	41.0	78.66	79.47(**+0.81**)
≤ 1	52.1	78.43	80.96(**+2.56**)	45.9	79.71	80.43(+0.72)
≤ 2	55.1	78.88	81.21(+2.33)	50.1	80.62	81.28(+0.66)
≤ 3	57.0	79.06	81.20(+2.14)	52.7	81.02	81.58(+0.56)
≤ 4	58.5	79.27	81.35(+2.08)	54.3	81.03	81.69(+0.66)
$\leq +\infty$	100.00	84.33	85.93(+1.60)	100.00	85.31	85.87(+0.56)

Table 6: Comparison of SpERT and our method on low-frequency entities on ACE2004 and ACE2005 test sets.

ties with different lengths, and entities with different frequencies in training data, respectively.

Nested entities In Table 4, we compare the recall of nested entities of our method and baseline SpERT on the validation and test sets of nested NER. Our method improves the recall of nested entities in both validation and test set by +1.00 $\sim$ 5.87. Generally, the improvement on nested entities is slightly larger than overall entities, which proves that our method tackles the nested problem better than the baseline.

Entities of different length Table 5 makes a detailed comparison with SpERT for entities of length 1 $\sim$ 10 on ACE2005 test set. Generally, our method has a higher F1 score. It can be seen that our method can effectively handle long entities with length ≥ 6. The largest improvement is +13.11 F1 for entities of length 8. We attribute it to that longer entities have richer n-gram features. The longer entities link more useful entity mentions in the span-level graph. The more informative neighbors of long entities significantly boost the performance by utilizing the label information of neighbors.

Low-frequency entities Table 6 compares with SpERT on the recall of entities with frequency ≤ 4 in the training set in ACE2004 and ACE2005 test sets. Our method yields +2.08 $\sim$ 2.56 and +0.56 $\sim$ 0.83 improvements in ACE2004 and ACE2005 respectively. For entities not in the training set, our method has +2.35 and $+0.81$ improvement in ACE2004 and ACE2005. Our span-level graphs provide information beyond the current entity and the sentence as lexically correlated similar entities. The information beyond the current sentence help improve the performance of "unseen" entities.

4.7 Case Study

Figure 4 is a case study in ACE2005 validation set to compare our method with SpERT. The original sentence has three entities to recognize. One entity is "The Bradleys", which is the Bradley fighting vehicle (VEH). However, the baseline SpERT classifies "The Bradleys" as a person (PER) with the misleading context. Our sub-graph links 2 VEH entities and makes the prediction correct. Besides, our method predicts "coaxial machine guns" correctly as a whole entity instead of "machine guns" partially. This attributes to the external guidance of 23 weapon (WEA) entity nodes in the span-level graph.

5 Conclusions

In this work, we enhance the span-based method for nested NER by including retrieval-based span-level graphs. Our method builds the entity-entity graph and the span-entity graph globally based on n-gram feature similarity. We use GCN to encode such structured correlations to obtain better span representation. We include multi-task learning to encode the label information of similar entities in the graph. The experimental results on three commonly used nested NER datasets, i.e. ACE2004, ACE2005 and GENIA, show that our method can improve the F1 score generally and improve recall for entities with low frequency in the training set.

6 Acknowledgements

This work is partly supported by Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102) and National Natural Science Foundation of China (62177033).

References

Kate Byrne. 2007. Nested named entity recognition in historical archive text. In *ICSC 2007*, pages 589–596.

Alberto Cetoli, Stefano Bragaglia, Andrew O'Harney, and Marc Sloan. 2017. Graph convolutional networks for named entity recognition. In *Proceedings of the 16th International Workshop on Treebanks and Linguistic Theories*, pages 37–45, Prague, Czech Republic.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

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

Ruixue Ding, Pengjun Xie, Xiaoyan Zhang, Wei Lu, Linlin Li, and Luo Si. 2019. A neural multi-digraph model for Chinese NER with gazetteers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1462–1467, Florence, Italy. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Markus Eberts and Adrian Ulges. 2020. Span-based joint entity and relation extraction with transformer pre-training. In *ECAI 2020*, volume 325, pages 2006–2013.

Jenny Rose Finkel and Christopher D. Manning. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 141–150, Singapore. Association for Computational Linguistics.

Joseph Fisher and Andreas Vlachos. 2019. Merge and label: A novel neural network architecture for nested NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5840–5850, Florence, Italy. Association for Computational Linguistics.

Jinlan Fu, Xuanjing Huang, and Pengfei Liu. 2021. SpanNER: Named entity re-/recognition as span prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7183–7195, Online. Association for Computational Linguistics.

Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun Ma. 2019. GraphRel: Modeling text as relational graphs for joint entity and relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1409–1418, Florence, Italy. Association for Computational Linguistics.

Tao Gui, Yicheng Zou, Qi Zhang, Minlong Peng, Jinlan Fu, Zhongyu Wei, and Xuanjing Huang. 2019. A lexicon-based graph neural network for Chinese NER. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1040–1050, Hong Kong, China. Association for Computational Linguistics.

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

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

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19:i180–i182.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations,* *ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinform.*, 36(4):1234–1240.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859, Online. Association for Computational Linguistics.

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

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

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 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: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3036–3046, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

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

Hiroki Ouchi, Jun Suzuki, Sosuke Kobayashi, Sho Yokoi, Tatsuki Kuribayashi, Ryuto Konno, and Kentaro Inui. 2020. Instance-based learning of span representations: A case study through named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6452–6459, Online. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Yongliang Shen, Xinyin Ma, Zeqi Tan, Shuai Zhang, Wen Wang, and Weiming Lu. 2021. Locate and label: A two-stage identifier for nested named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2782–2794, Online. Association for Computational Linguistics.

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

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

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

Chuanqi Tan, Wei Qiu, Mosha Chen, Rui Wang, and Fei Huang. 2020. Boundary enhanced neural span classification for nested named entity recognition. In *The Thirty-Fourth AAAI Conference on Artificial*

Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pages 9016–9023. AAAI Press.

Zeqi Tan, Yongliang Shen, Shuai Zhang, Weiming Lu, and Yueting Zhuang. 2021. A sequence-to-set network for nested named entity recognition. In *IJCAI 2021*, pages 3936–3942.

Bailin Wang and Wei Lu. 2018. Neural segmental hypergraphs for overlapping mention recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 204–214, Brussels, Belgium. Association for Computational Linguistics.

Bailin Wang, Wei Lu, Yu Wang, and Hongxia Jin. 2018. A neural transition-based model for nested mention recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1011–1017, Brussels, Belgium. Association for Computational Linguistics.

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

Yiran Wang, Hiroyuki Shindo, Yuji Matsumoto, and Taro Watanabe. 2021. Nested named entity recognition via explicitly excluding the influence of the best path. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3547–3557, Online. Association for Computational Linguistics.

Yongxiu Xu, Heyan Huang, Chong Feng, and Yue Hu. 2021. A supervised multi-head self-attention network for nested named entity recognition. In *AAAI 2021*, pages 14185–14193.

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

Yue Zhang and Jie Yang. 2018a. Chinese NER using lattice LSTM. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1554–1564, Melbourne, Australia. Association for Computational Linguistics.

Yue Zhang and Jie Yang. 2018b. Chinese NER using lattice LSTM. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1554–1564, Melbourne, Australia. Association for Computational Linguistics.

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

A Hyper-parameters Analysis

Loss coefficient Multi-task loss coefficient β is important for the balance of raw span classification and entity label usage. We set $\beta \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$ and test the sensitivity in Fig. 5. The results show that both large and small values of β degenerate the performance. The best results are attained when β is $0.1 \sim 0.3$ for ACE2004 and ACE2005, and 0.1 for GENIA. Small β won't train entity representation sufficiently, which the raw span takes attention from. While β disables models focusing on span classification.

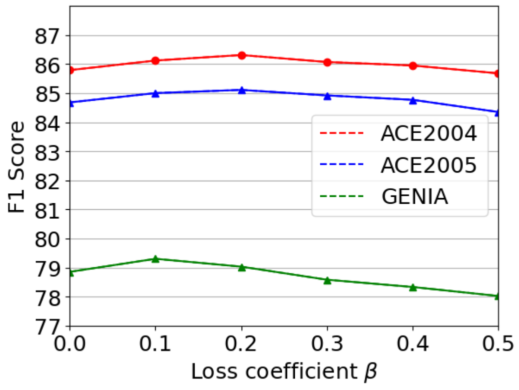
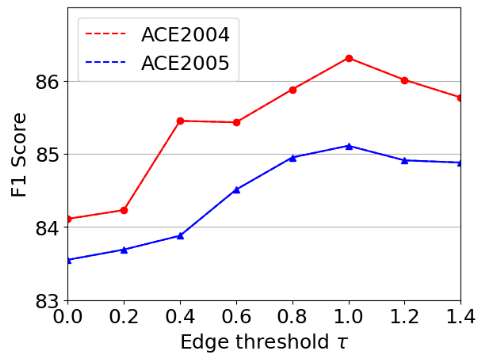


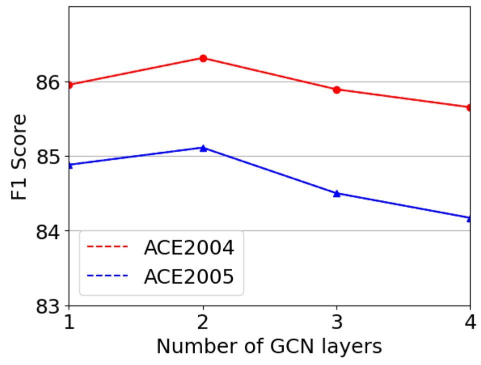
Figure 5: The effect of loss balance coefficient β. We set $\beta \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$ for each dataset.

Graph hyper-parameters We explore two graph hyper-parameters edge threshold τ and GCN layers K in ACE2004/ACE2005. In Fig. 6(a), we set $\tau \in \{0.0, , 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4\}$ and τ around 1.0 achieves the best performance. Large edge threshold influences the performance slightly while small τ degenerates the performance a lot by introducing much noise into the graph.

In Fig. 6(b), we set GCN layers $K \in \{1, 2, 3, 4\}$ and find that 2-layer GCN with 2-hop span-level graph performs better. When $K = 1$, our method degrades to using only a span-entity graph and the performance decreases a little. Stacking multiple layers of GCNs leads to high complexity in back-propagation, which is not conducive to the training.



(a) Edge Threshold



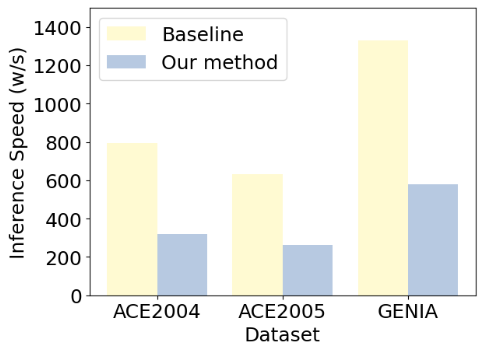
(b) GCN Layers

Figure 6: The effect of graph hyper-parameters in ACE2004 and ACE2005 datasets. We set edge threshold $\tau \in \{0.0, , 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4\}$ and GCN layers $M \in \{1, 2, 3, 4\}$.

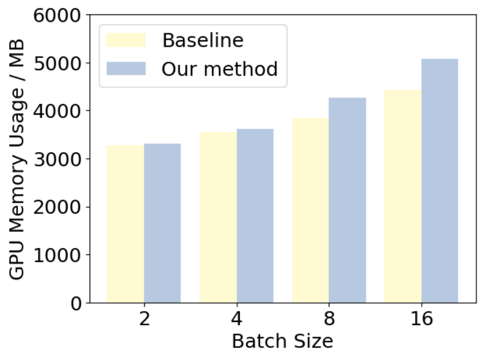
B Time and Space Analysis

Inference speed In Fig. 7(a), we compare our method with SpERT at the inference speed, where metrics is the number of words decoded per second. We evaluate at four nested NER test set on GTX 1080 Ti. The batch size is fixed to 4. Our inference speed is around half of SpERT. Although the introduction of the graph module increases the inference time, the cost is acceptable considering the significant recognition improvement.

GPU usage In Fig. 7(b), we compare the GPU memory usage (/MB) of our method and baseline SpERT on ACE2005 test set with batch size in $\{2, 4, 8, 16\}$. It shows that the graph module only increases the GPU usage for a small part, around $100 \sim 500$ MB. With the increase of batch size, the GPU usage increment also increases.



(a) Inference Speed



(b) GPU Usage

Figure 7: The inference speed of our method and baseline on the test sets. We evaluate word per second (w/s). The GPU memory usage (/MB) of our method and baseline with different batch size on ACE2005 training set. We use GeForce GTX 1080 Ti as the device.