Trigger-GNN: A Trigger-Based Graph Neural Network for Nested Named Entity Recognition

Anonymous Authors

***Abstract*—Nested named entity recognition (NER) aims to identify the entity boundaries and recognize categories of the named entities in a complex hierarchical sentence. Some works have been done using character-level, word-level, or lexicon-level based models. However, such researches ignore the role of the complementary annotations. In this paper, we propose a trigger-based graph neural network (Trigger-GNN) to leverage the nested NER. It obtains the complementary annotation embeddings through entity trigger encoding and semantic matching, and tackle nested entity utilizing an efficient graph message passing architecture, aggregation-update mode. We posit that using entity triggers as external annotations can add in complementary supervision signals on the whole sentences. It helps the model to learn and generalize more efficiently and cost-effectively. Experiments show that the Trigger-GNN consistently outperforms the baselines on four public NER datasets, and it can effectively alleviate the nested NER.**

***Index Terms*—Nested named entity recognition, recursive graph neural network, entity trigger.**

I. Introduction

Named entity recognition (nested-NER) aims to identify the entity boundaries and recognize the categories of named entities in a sentence [1], [2]. The categorizes belong to the predefined semantic types, such as person, location, organization [3]. With applications ranging from AI-based dialogue systems to combing the natural language and semantic web in learning environments, NER plays a standalone role.

However, nested NER is always a thorny challenge due to its complex hierarchical structure. Fig 1 illustrates two examples of the nested entity strings. The upper example shows that "Thomas Jefferson, third president of the United States" should be labeled jointly to constitute a complete entity statement and expressed as a person entity (PER). However, it also involves two distinct entities: "the United States" and "third president of the United States", which can be expressed as a geopolitical entity (GEO) or a PER separately. The nest problem hampers the judgment of the entities boundaries. Sometimes, it appears as an overlapping text. As shown as the bottom example of Fig 1, "Pennsylvania radio station" is overlapped. "Pennsylvania" can be treated as a PER or concatenate with the following two words as a organization institute (ORI), "Pennsylvania radio station".

Generally, one intuitive way to leverage the nested NER is to stack flat NER layers [4]–[7]. Ju *et al.* [5] proposed to identify nested entities by dynamically stacking flat NER layers. It merges the output of the LSTM from the current NER layer and subsequently feeds them into the next flat NER layer. It makes full use of the information encoded in the corresponding internal entities (entities exist in the internal layers) in an inside-out manner. However, this kind of layered model cannot reuse the outside information, which means it has a single direction for information transfer from the inner layer to the outside way. Hence, joint learning flat entities and their inner dependencies has attracted research attention. Luo *et al.* [8] proposed a bipartite flat graph network considering the bidirectional delivery of information from innermost layers to outer ones. It indeed tackle the lack of the dependencies of inner entities. However, it still lack clear rationales to execute the delivery process. The nested NER is still a nontrivial task due to its complex structure.

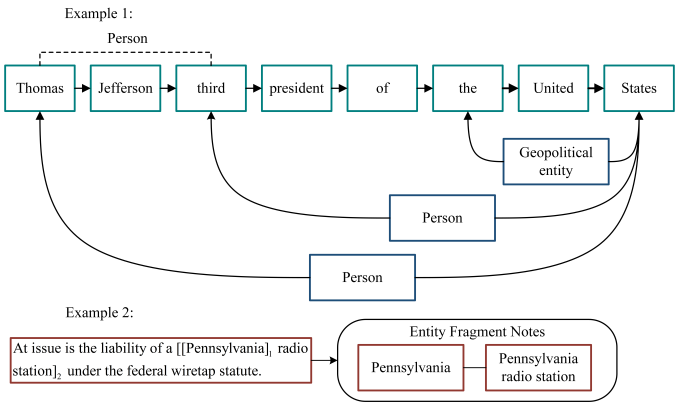

Fig. 1. Examples of the nested entity recognition.

Recent advances in nested-NER mainly focus on training an superior neural network model on different levels of the semantic hierarchy, *e.g.*, character-level [9], word-level [2], and lexicon-level [10], [11]. However, such researches ignore the role of the complementary annotation [1]. In recent years, some works have been done using this external supplementary explanation as compensation [12]. Lin *et al.* first proposed entity trigger as an complementary annotation for facilitating label-efficient learning of NER models. It defines the entity trigger as a group of words of the sentence, which helps to explain why humans would recognize an entity in the sentence. For example, given a sentence "We had a fantastic lunch at Rumble Fish yesterday, where the food is my favorite.", where "had $\cdots$ lunch at" and "where the food" are two

[1]Complementary annotation refers to the supplementary explanation representing why humans would recognize an entity in a sentence. For example, it can be inferred that 'Silicon' is likely to be a location entity in the sentence "Tom traveled a lot last year in Silicon." We recognize this entity because of the cue phrase "travel . . . in", which suggests there should be a location entity following the word 'in'. This is how the complementary annotation works.

distinct triggers associated with the restaurant entity "Rumble Fish". These distinct triggers [2] explicitly indicate the location information of the candidate entities and helps to better anchor them. In addition, the defined entity triggers can be reused. Statements with similar phrase structure can be linked to a equivalent class and reuse the same entity trigger as well. It means the most of our commonly used sentences can be formed into some entity triggers rather than labelling all these sentences manually. It helps the model training to be cost-effectively.

Our intuition is that these triggers can add in complementary supervision on the whole sentences and thus help the nested-NER models to learn and generalize more efficiently and cost-effectively. However, this trigger-based method utilizes the recurrent neural networks (RNN) to encode the sentences sequentially. It overlooks that the underlying structure of the sentence is not strictly sequential. In fact, RNN-based models process words in a strictly serial order, and a word have precedence when assigned to its left word. More seriously, these methods label the candidate entities only using the previous partial sequences without seeing the remain words. It lacks the capacity of capturing the long-range dependency and high-level features in the sentence.

To this end, we introduce a trigger-based graph neural network (Trigger-GNN). It casts the nested-NER into a node classification task, and breaks the serialization processing of RNNs using an recursive graph neural network. The unique challenge in nested NER is that we have to deal with the rules which are discontinuous token sequences and there may be multiple rules applied at the same time for an input instance. We address this problem by using a updation-aggregation training manner, *i.e.*, the node representation is computed by aggregating its adjacent edges and the graph-level node recursively. The multiple iterations of aggregation enables Trigger-GNN to continuously verify the nested words based on the global context information. In addition, we design a graph-level node to capture the long-range dependency and some high-level features of the entire sentence. The key contributions of this paper can be summarized as follows:

- We develop a trigger-based graph neural network for the nested NER task in a cost-effective manner, and cast the problem into a graph node classification task.
- We propose to capture the global context information and local compositions to tackle nested NER through a recursively aggregating mechanism.
- Experiments show that the Trigger-GNN is cost-effective and efficient on four public NER datasets.

II. Related Work

A. Graph Neural Networks on Texts

Graph neural networks have been successfully applied to several text classification tasks [13]–[17]. Liu *et al.* [16] proposed a tensor graph convolutional networks called Tensor-GCN. The Tensor-GCN uses the text graph tensor to describe semantic, syntactic, and sequential contextual information. The model uses the combination of the intra-graph propagation and inter-graph propagation to aggregate the information from the neighborhood nodes in a single graph, and to harmonize heterogeneous information between graphs. Huang *et al.* [17] proposed to build graph for each input text with global sharing parameters instead of training only on a single graph for the whole corpus. It indeed removes the burden of dependence between an individual text and the entire corpus. However, it still preserves the global information. Yao *et al.* [14] developed a single text graph convolutional network (Text-GCN) based on the word co-occurrence and document-word relations. It jointly learns the embeddings of both words and documents, and it is supervised by the documents' class labels.

B. Nested Named Entity Recognition

There has been a growing amount of efforts towards NER, and have been explored in several directions, including rule-based, statistics-based and deep neural network-based [1], [6], [9], [18]–[20]. However, nested named entity recognition is always a thorny issue because of its complex hierarchical structure.

Towards alleviating the nested NER, recent works have mainly focus on stacking flat NER layers [4]–[7]. Ju *et al.* [5] proposed to identify the nested entities by dynamically stacking flat NER layers. The model merges the output of the LSTM in the current flat NER layer to build a revolutionary representation for detected entities, and subsequently feeds them into the next flat NER layer. It makes full use of the encoded information in the corresponding internal entities in an inside-out manner. However, the layered model cannot reuse the outside information, which means it has a single direction for information transfer from the inner layer to the outside way.

Another line of research aims to combine the bipartite graph with the flat NER layers. Luo *et al.* [8] proposed a bipartite flat GNN to learn the flat entities and their inner dependencies jointly. It constructs a graph module to deliver the bidirectional information from innermost graph layer to the outermost one. The leaned information carries the dependencies of inner entities and can be exploited to improve outermost entity predictions. Due to the graph module, the transfer of the information is bidirectional. The layered model can reuse the outside information as well. However, this method lacks clear rationales to conduct the notation process, unlike the trigger annotations.

Inspired by the recent advances in aforementioned sections (e.g., graph neural networks on texts [10] and learning with entity triggers [12]), we propose to use a graph neural network integrating the complementary annotation. The Trigger-GNN enables two significant issues: (1) multiple graph-based interactions among the words, entity triggers, and the whole sentence semantics; and (2) using trigger-based instead of lexicon-based [10] can add in complementary supervision

[2]To be noted that an entity trigger should follows some rules: (1) an entity trigger should involves necessary and sufficient cue for entity recognition process; (2) some modifier words should be removed, *e.g.*, "fantastic".

signals and thus help the model to learn and generalize more efficiently.

III. Trigger-Based Graph Neural Network

In this section, we detail the Trigger-GNN model. The idea of the whole model is briefly illustrated in Fig 2. Trigger-GNN obtains the complementary annotation embeddings through entity trigger encoding and semantic matching. And it uses an efficient graph message passing architecture [21], aggregation-updation mode, to better extract the interactions among the words, sentences, and complementary annotations.

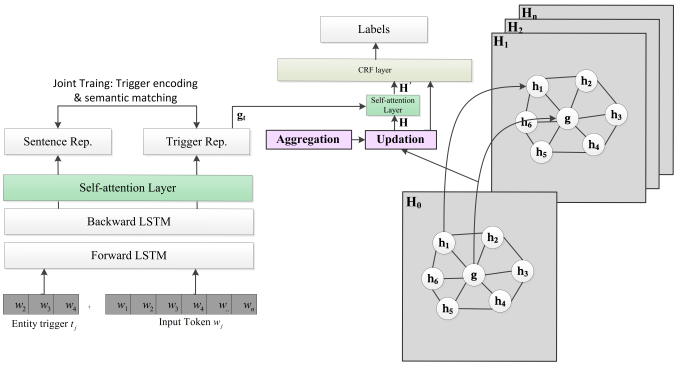

Fig. 2. Trigger-based graph neural networks model. It obtains the sentence representation and trigger representation from the input entity trigger t_j and input token w_j through a bidirectional LSTM and self-attention based soft matching. We jointly train the model using both trigger encoding and semantic matching. In addition, the learned embedding will be executed through an aggregation-updation based graph neural network to better extract the interactions among the words, sentences, and the complementary annotations.

A. Trigger Encoding and Semantic Matching

In this section, we propose to train an trigger encoder, and to match the triggers and the sentences using an attention-based representations. Our intuition is that the desired trigger vectors should capture the semantics in a shared embedding space with the hidden states of the token. Specifically, for a sentence $S = w_1, w_2, \ldots, w_n$ with multiple entities $e_1, e_2, \ldots, e_k$, we assume that there is a set of triggers $T_i = t_1, t_2, \ldots, t_n$. To enable efficient batch training, we reformat to link each entity with one of its triggers denoted as (x, e_i, t_j), where x is the tokenization list of the sentence S. For each reformed training batch, we first apply a bidirectional LSTM on the sequence of word vectors of x using Glove word embedding [22]. It obtains the hidden states h_i for each token x_i. $\mathbf{H}$ denotes the matrix containing the hidden vectors of all the tokens. And $\mathbf{Z}$ denotes the matrix containing the hidden vectors of all the trigger T. We utilize the self-attention method introduced by [12] to obtain the representation of both triggers and sentences:

$$\begin{aligned} \mathbf{g_s} &= \text{SoftMax}\left(W_2 \tanh\left(W_1 \mathbf{H}^T\right)\right)\mathbf{H} \\ \mathbf{g_t} &= \text{SoftMax}\left(W_2 \tanh\left(W_1 \mathbf{Z}^T\right)\right)\mathbf{Z} \end{aligned} \tag{1}$$

where W_1 and W_2 are two trainable parameters in the model. $\mathbf{g_s}$ is denoted as the final sentence's vector, representing the weighted sum of all the token vectors in the entire sentence. Similarly, $\mathbf{g_t}$ denotes the final trigger's vector, representing the weighted sum of the all the trigger vectors.

We use the type of the associated entity of the trigger as an supervision for training. Before that, the trigger's vector $\mathbf{g_t}$ is fed into a multi-class classifier to predicate the type of the associated entity. The obtained associated entity is denoted as Ω. The loss function of the trigger classification is as follows:

$$L_1 = -\sum log(\Omega|\mathbf{g_t}; \theta_1) \tag{2}$$

where θ_1 is a trainable parameter in the model.

We match the triggers and sentences by $\mathbf{g_t}$ and $\mathbf{g_s}$ according to Eq.4. For the training process, we sample some negative examples by randomly mixing the triggers and sentences. For the negative examples, we expect a margin m between their embeddings. The loss function of the matching module is defined as follows:

$$L_2 = \alpha\frac{1}{2}(||\mathbf{g_s} - \mathbf{g_t}||_2)^2 + (1-\alpha)\frac{1}{2}\{\max(0, m - ||\mathbf{g_s} - \mathbf{g_t}||_2)\}^2 \tag{3}$$

where α is defined to confirm whether the trigger is originally in the sentence of not. When α is set to 1, the trigger is originally in the sentence; and when it is set to 0, the trigger is not originally in the sentence. The joint loss function of the trigger encoding and semantic matching is $L = L1 + \lambda L2$, where λ is a hyper-parameter for fine-tuning. In our experiment, we define $\lambda = 1.3$.

B. Text Graph Construction

In this section, we detail how to convert the whole sentences into a directed graph. Each word in the sentences is represented by a node, and the edges between each node are according to lexicon. Also, we design a graph-level node, which is used as a virtual hub to gather the information from all the nodes and edges. The graph-level node can help the node representation by removing the ambiguity.

Formally, let $S = w_1, w_2, \ldots, w_n$ denote a sentence, where w_i denotes the i-th word. The potential lexicon $T = t_1, t_2, \ldots, t_n$ matching a word sub-sequence can be formulated as $t_{b,e} = w_b, w_{b+1}, \ldots, w_{e-1}, w_e$, where the index of the first word and the last word are b and e. In this work, we propose to denote directed and labeled multi-graphs as $G = (V, E, U)$ with nodes (words: $v_i \in V$), labeled edges (relations: $(t_{b,e}, r) \in E$, where $r \in R$ is a relation type according to the entity trigger), and graph-level node (global attributes: $u_i \in U$). Once a word sub-sequence matches a candidate lexicon such as $t_{b,e}$, we add one edge $e_{b,e} \in E$, pointing from the beginning word w_b to the ending word w_e. To capture the global information, we add a graph-level node to connect each word node and its corresponding lexicon edge. For a graph with n word nodes and m edges, there are $m + n$ global connections linking each node and edge to the shared graph-level node. In addition, we construct the transpose of the constructed graph according to [10]. It is another directed graph on the same set of nodes with all edges reversed compared to G. It denotes as G^{T}. Similar to the bidirectional LSTM, we compose G and G^{T} as a bidirectional

graph and concatenate the node states of G and G^{T} as final outputs.

C. Recursive Graph Neural Networks

In this part, we detail the structure of the recursive-based graph neural network from three subsections: updation module, aggregation module, and trigger-enhanced decoding and tagging.

1) Updation Module: The hidden vectors of all of the tokens $\mathcal{H}$ is used as the representation of each node in the graph which denoted as $\mathcal{H} = x_1, x_2, \ldots, x_n$. And the hidden vectors of the trigger token $\mathcal{Z}$ is used as the representation of each edge which denoted as $\mathcal{Z} = y_1, y_2, \ldots, y_m$. Formally, at the ℓ-th layer, the hidden representation of the text graph is denoted as:

$$H^{\ell} = \left\langle h_1^{\ell}, h_2^{\ell}, \ldots, h_m^{\ell}, g^{\ell} \right\rangle \tag{4}$$

where h_i is the hidden status of each node. g represents the graph-level node.

For the initial state H^0, the hidden states of the i-th nodes is set to its embedding, $h_i^0 = x_i$. The transition from $h_i^{\ell-1}$ to h_i^{ℓ} is calculated as follows:

$$\begin{aligned}
i_i^l &= \sigma\left(W_i\left[h_i^{l-1}; x_i; g^{l-1}; \mathcal{N}_i^{l-1}\right] + b_i\right) \\
f_i^l &= \sigma\left(W_f\left[h_i^{l-1}; x_i; g^{l-1}; \mathcal{N}_i^{l-1}\right] + b_f\right) \\
o_i^l &= \sigma\left(W_o\left[h_i^{l-1}; x_i; g^{l-1}; \mathcal{N}_i^{l-1}\right] + b_o\right) \\
u &= \tanh\left(W_u\left[h_i^{l-1}; x_i; g^{l-1}; \mathcal{N}_i^{l-1}\right] + b_u\right) \\
c_i^l &= f_i^l \odot c_i^{l-1} + i_i^{l-1} \odot u \\
h_i^l &= o_i^l \odot \tanh\left(c_i^l\right)
\end{aligned} \tag{5}$$

where x_i in each layer is used to introduce the original meaning of the token; $\mathcal{N}_i$ denotes the aggregated hidden representation of the node w_i's neighbors; $[;;]$ represents the concatenation of the vectors; i, f, o represent the input, forget and output gates respectively.

2) Aggregation Module: It assumes that w_i has C_i neighbors in the text graph, the aggregated hidden representation $\mathcal{N}_i$ of the neighbors C_i is calculated as follows:

$$\begin{aligned}
\mathcal{P}_i &= h_i + p_i \\
\alpha_j &= u\left(W_n\left[\mathcal{P}_i; x_i; g; \mathcal{P}_j\right] + b_n\right) \\
\mathcal{S}_j &= \frac{\exp\left(\alpha_j\right)}{\sum_k^{C_i} \exp\left(\alpha_k\right)} \\
\mathcal{N}_i &= \sum_k^{C_i} \mathcal{S}_k h_{i_k}
\end{aligned} \tag{6}$$

where p_i refers to the position embedding of the node w_i; h_{i_k} refers to the hidden state of the k-th neighbor of node w_i. With positional encoding added, it makes easier to aware of the The positional encoding is added to make our model more general, *i.e.*, aware of the position information of word i and j more seriously. The combination of the gated-graph neural network and attention-based aggregation makes our model capable of gathering the information from the long-dependency as the layer number increases. It determines which part of the information should be passed to higher layers.

The value of g^{ℓ} is computed based on the hidden states from the last layer $H^{\ell-1}$. And $\widetilde{h}^{\ell-1}$ is calculated using attention mechanism as: $\left\langle h_1^{\ell}, h_2^{\ell}, \ldots, h_m^{\ell} \right\rangle$.

$$\begin{aligned}
\alpha_i &= u\left(W_a h_i\right) \\
\mathcal{S}_i &= \frac{\exp\left(\alpha_i\right)}{\sum_j \exp\left(\alpha_j\right)} \\
\widetilde{h} &= \sum_j \mathcal{S}_j h_j
\end{aligned} \tag{7}$$

Then, for each $h_i^{\ell-1}$, a gate f_i^{ℓ} is calculated to decide which information should be considered by the graph-level vector g^{ℓ}. The candidate state c_g^{ℓ} is calculated based on the candidate graph state at the last layer $c_g^{\ell-1}$, and the candidate states of the nodes at the last layer is denoted as $c_i^{\ell-1}$:

$$\begin{aligned}
\hat{f}_g^l &= \sigma\left(W_g\left[g^{l-1}; \widetilde{h}^{l-1}\right] + b_g\right) \\
\hat{f}_i^l &= \sigma\left(W_f\left[g^{l-1}; h_i^{l-1}\right] + b_f\right) \\
o^l &= \sigma\left(W_o\left[g^{l-1}; \bar{h}^{l-1}\right] + b_o\right) \\
f_0^l, \cdots, f_m^l, f_g^l &= \mathrm{Softmax}\left(\hat{f}_0^l, \cdots, \hat{f}_m^l, \hat{f}_g^l\right) \\
c_g^l &= f_g^l \odot c_g^{l-1} + \sum_i f_i^l \odot c_i^{l-1} \\
g^l &= o^l \odot \tanh\left(c_g^l\right)
\end{aligned} \tag{8}$$

3) Trigger-Enhanced Decoding and Tagging: Given the the vector of the whole graph g^L and the vectors of each node $\mathbf{H} = \left\langle h_1^L, h_2^L, \cdots, h_m^L \right\rangle$, we use the previously trained module in III-A to compute the mean value of all the vectors of the trigger $\hat{\mathbf{g_t}}$ associated with its sentence. We incorporate the attention-based token representation $\mathbf{H}^{'}$ as follows:

$$\mathbf{H}^{'} = \mathrm{SoftMax}\left(W^T \tanh\left(U_1 \mathbf{H}^T + U_2 \hat{\mathbf{g_t}}^T\right)\right)^T \mathbf{H} \tag{9}$$

where U_1, U_2, and W are trainable parameters. We concatenate the original token representation $\mathbf{H}$ with the trigger-enhanced one $\mathbf{H}^{'}$ as the input ($[\mathbf{H}; \mathbf{H}^{'}]$) to the final CRF tagger. Note that in this stage, our learning objective is the same as the conventional NER according to [12], which is to correctly predict the tag for each token.

When predicting tags on an unlabeled sentences, we do not know the sentence's triggers. Instead, we use the trigger encoding and matching module to compute the similarities between the sentence's representations and the trigger's representations according to L2-norm distances. The most suitable triggers will be used as additional inputs to the last tagging process.

IV. Experiment Analysis

A. Datasets

We first introduce the NER datasets used in the experiments. All the datasets are public and popular in NER task.

- **CoNLL-2002 & CoNLL-2003**: We use the English corpora from the CoNLL 2002 and 2003 shared task: language-independent named entity recognition [23], [24] with standard split.

- **JNLPBA**: We use a molecular biology dataset for identifying the technical terms [25]. It contains various type of nested entities. Examples of such entities include the names of proteins, genes and their location of activity such as cells or organism. We use the same standard split according to [26].
- **BC5CDR**: Another bio-medical domain dataset which is well-studied and popular in evaluating the performance of nested-NER [27]. We use the standard split.

Both JNLPBA and BC5CDR datasets involves a large number of nested entities in the bio-medical domain.

B. Lexicon

We use the lexicon generated on a corpus of article pairs from Gigaword [3], consisting around 4 millon articles. Use tensorflow as implementation [4]. The embeddings of the lexicon words were pre-trained by Glove word embedding [22] and fine-tuned during training.

C. Baselines

Several state-of-the-art NER algorithms (sorted by the timeline) for effectiveness evaluation are listed below:

- **Neural layered model**: It merged the output of the stacked LSTM layers to build new representation for detected entities. [5].
- **Lattice LSTM**: It raised a lattice-structure based LSTM model, which encodes a sequence of input characters as well as all the potential words matching a lexicon [11].
- **Boundary-aware neural model**: It proposed a boundary-aware neural model for nested NER using sequence labelling [28].
- **Biaffine-NER**: It proposed a graph-based dependency parsing to provide a global view on the sentence [29].
- **BiFlaG**: It firstly used the entities recognized by the flat NER module to construct an entity graph [8].
- **LGN**: It developed a lexicon-based graph network with global semantics and proposed a schema to connect the character using external lexicon words [10].
- **Trigger-NER**: It first introduced the concept of "entity trigger" and proposed a RNN-based trigger matching network for NER [12].
- **ACE + document-context**: It proposed an automated concatenation method of embeddings (ACE) to automate the process of finding better concatenations of embeddings for structured prediction tasks, NER is one of them [30].

D. Experiment Settings

1) Annotating Entity Triggers as Explanatory Supervision: We follow [12] to crowd-source the entity triggers from human annotators, and use the LEAN-LIFE developed by [31] for annotating. Annotators are asked to annotate a group of words which would be helpful in typing and detecting the occurrence of the particular entity in the sentence. We mask the entity tokens with their types so that human annotators are more focused on the non-entity words in the sentence when considering the triggers. The multiple triggers are consolidated for each entity taking the intersection of all annotators' results. We reuse the 14K triggers from [12] and release another 8K triggers on the bio-domain for future domain research in trigger-enhanced NER, which is also one of the contributions of our work.

2) Hyper-parameter Settings: We develop the Trigger-GNN based on Pytorch, with the same learning rate setting selected from $\{2e-5, 2e-4\}$ as [10]. To further reduce the overfitting issue, we employed the Dropout [32] with a rate of 0.4 for embeddings and a rate of 0.3 for aggregation module outputs. The embedding size and the hidden states were both set to 150. The initial word vectors is based on the Glove word embedding [22]. Steps of message passing T are selected from $\{1, 2, 3, 4, 5, 6\}$ which is detailed analyzed in IV-E1. We use the standard metrics for evaluation: Precision (P), Recall (R), and F1 score (F1).

E. Evaluation and Discussion

We demonstrate the main results of Trigger-GNN for nested NER across general and bio domains. Results on the CoNLL-2002 & 2003, JNLPBA and BC5CDR datasets are shown in Table I, II, and III, respectively. Compared with the recent methods, our Trigger-GNN obtains the best results by a large margin. In particular, Trigger-GNN obtains 1.93% and 3.52% boosting than its baseline model LGN on the general domain dataset CoNLL-2002 & 2003. And as shown in Table II, and III, Trigger-GNN obtains 3.41% improvement on JNLPBA than LGN, and 5.32% on BC5CDR. The observation from the main results on these four dataset demonstrates that our proposed model can adapt to the both general domain and bio domain, and perform better than the recent methods.

TABLE I
MAIN RESULTS ON CONLL-2002 & 2003

Models	CoNLL-2002	CoNLL-2003
Neural layered model [5]	85.23%	85.13%
Lattic LSTM [11]	90.34%	91.28%
Boundary-aware model [28]	87.58%	91.58%
Biaffine-NER [29]	91.38%	91.63%
BiFlaG [8]	92.54%	92.67%
ACE+document-context [30]	93.95%	94.63%
Trigger-NER [12]	86.83%	86.95%
LGN [10]	92.19%	91.86%
Trigger-GNN	**94.92%**	**95.38%**

1) Steps of Message Passing: To investigate the influence of step number T during the update process, we analyze the performance of Trigger-GNN under different step numbers as shown in Fig. 3. The results show that the number of update steps has an important impact on the performance of Trigger-GNN. The F1 score decreases 4.1% on average against the best results when T is less than 3. Specifically, the F1 score on both JNLPBA and BC5CDR datasets even suffered a reduction

[3]https://catalog.ldc.upenn.edu/LDC2003T05
[4]https://www.tensorflow.org/datasets/catalog/gigaword

TABLE II
MAIN RESULTS ON JNLPBA.

Models	P	R	F1
Neural layered model [5]	81.75%	81.24%	81.49%
Lattic LSTM [11]	83.25%	80.25%	81.72%
Boundary-aware model [28]	80.12%	75.12%	77.53%
ACE+document-context [30]	85.75%	83.62%	84.67%
Trigger-NER [12]	79.12%	76.34%	77.70%
LGN [10]	84.12%	82.14%	83.12%
Trigger-GNN	**87.75%**	**85.34%**	**86.53%**

TABLE III
MAIN RESULTS ON BC5CDR.

Models	P	R	F1
Neural layered model [5]	87.12%	86.14%	86.62%
Lattic LSTM [11]	89.34%	87.89%	88.61%
Boundary-aware model [28]	86.12%	85.12%	85.62%
ACE+document-context [30]	93.65%	92.32%	92.98%
Trigger-NER [12]	85.75%	83.62%	84.67%
LGN [10]	90.12%	86.21%	88.13%
Trigger-GNN	**94.19%**	**92.73%**	**93.45%**

around 3.63% and 3.53%. After several rounds of updates, the model gives steady and competitive results and reveals that the Trigger-GNN benefits from the update process. Empirically, at each update step, graph nodes aggregate information from their neighbors and incrementally gain more information from the global-based graph node as the process iterates.

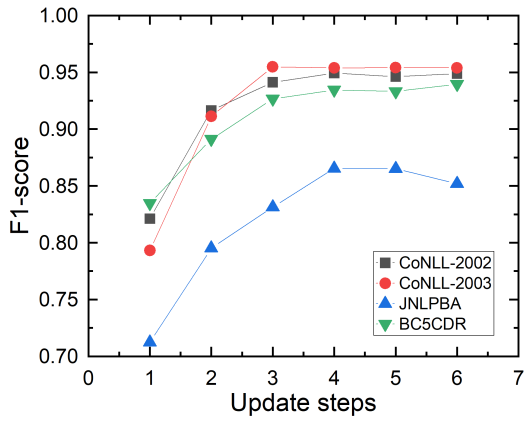

Fig. 3. F1 variation under different update steps on the four datasets.

2) Ablation Studies: To study the contribution of each component in Trigger-GNN, we conduct the ablation studies on four public datasets and the results are as illustrated in Table IV. The results show that the model's performance is degraded when the graph-level node is removed. It indicats that the global connections are useful in the graph structure. We observe that the entity triggers play an vital role in the nested NER. Specifically, the performance of the CoNLL-2003, BC5CDR and JNLPBA are seriously hurt by average 3.0% without entity triggers. Moreover, missing the edge/lexicon will cause a further performance loss about 1.5% on average.

TABLE IV
AN ABLATION STUDY OF TRIGGER-GNN. F1 SCORES WERE EVALUATED.

Models	CoNLL-2003	JNLPBA	BC5CDR
Trigger-GNN	95.38%	86.53%	93.45%
-graph-level node	94.31%	85.87%	92.34%
-trigger	**91.86%**	**83.12%**	**88.13%**
-edge/lexicon	90.42%	81.74%	87.83%
-bidirectional	88.12%	76.51%	84.38%
-crf	86.87%	74.93%	82.80%
LGN	91.86%	83.12%	88.13%
-graph-level node	89.73%	81.56%	87.32%

To better illustrate the advantage of our model, we compare our trigger-based GNN with LGN using lexicon instead. The results show that the Trigger-GNN achieves a higher F1 score by 2.68% on average than the LGN. In addition, the two models have a performance gap when remove the global node. This is because the trigger-based GNN can add in complementary supervision on the words, entity triggers and the whole sentence. The F1 score of LGN decreases by 1.83% on average on the four datasets without graph-level node. In contrast, our Trigger-GNN decreases by 0.89% which manifests the Trigger-GNN has stronger ability to model sentences.

3) Performance Against Labeled Data: Fig. 4 shows the performance of Trigger-GNN and several baseline methods on the CoNLL2003 dataset using different number of the labeled data. We can see that by using only 20-30% of the trigger-annotated data for training, Trigger-GNN model delivers comparable performance as the baseline model LGN using 50-70% traditional training data like lexicon. It indicates the cost-effectiveness of using triggers as an additional source of supervision.

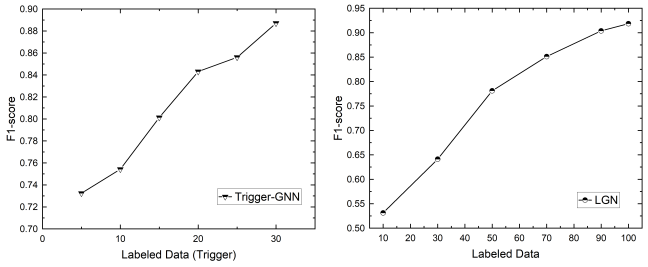

Fig. 4. F1 score against labeled data on CoNLL-2003.

4) Case Studies: To further validate the Trigger-GNN can alleviate the nested NER, we perform an case study on BC5CDR as illustrated in Table V. It demonstrates that the Trigger-GNN performs well on the nested NER case. In particular, Trigger-GNN can not only identify the "selegiline" as a chemical, but also can integrate the "supine systolic and diastolic blood pressures" as a integral disease. However, models like LGN and Trigger-NER can only capture part of the entities or incorrectly split the diseases, which illustrates the superiority of our proposed model.

TABLE V
CASE STUDIES ON BC5CDR.

Models	Cases
LGN	Stopping selegiline also significantly reduced the supine systolic and diastolic blood pressures (Disease) consistent with a previously supine pressure action.
Trigger-NER	Stopping selegiline (Chemical) also significantly reduced the supine systolic (Disease) and diastolic blood pressures consistent with a previously supine pressure action.
Trigger-GNN	Stopping selegiline (Chemical) also significantly reduced the supine systolic and diastolic blood pressures (Disease) consistent with a previously supine pressure action.

V. CONCLUSION

In this work, we investigate a trigger-based graph neural network approach to alleviate the nested NER. Entity triggers are used to provide more explicit supervision. The Trigger-GNN enables two significant issues: (1) multiple graph-based interactions among the words, entity triggers, and the whole sentence semantics; and (2) using trigger-based instead of lexicon-based [10] can add in complementary supervision signals and thus help the model to learn and generalize more efficiently. As a result, the experiments indicate the significant performance of our proposed model on four real-world datasets in both general and bio domains. The explanatory experiments also illustrate the efficiency and the cost-effectiveness of our proposed model.

REFERENCES

[1] V. Yadav and S. Bethard, "A survey on recent advances in named entity recognition from deep learning models," *CoRR*, vol. abs/1910.11470, 2019. [Online]. Available: http://arxiv.org/abs/1910.11470

[2] J. P. C. Chiu and E. Nichols, "Named entity recognition with bidirectional lstm-cnns," *Trans. Assoc. Comput. Linguistics*, vol. 4, pp. 357–370, 2016. [Online]. Available: https://transacl.org/ojs/index.php/tacl/article/view/792

[3] Q. Tran, A. MacKinlay, and A. Jimeno-Yepes, "Named entity recognition with stack residual LSTM and trainable bias decoding," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers*, G. Kondrak and T. Watanabe, Eds. Asian Federation of Natural Language Processing, 2017, pp. 566–575. [Online]. Available: https://aclanthology.org/I17-1057/

[4] M. Ju, M. Miwa, and S. Ananiadou, "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, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, M. A. Walker, H. Ji, and A. Stent, Eds. Association for Computational Linguistics, 2018, pp. 1446–1459. [Online]. Available: https://doi.org/10.18653/v1/n18-1131

[5] ——, "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, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, M. A. Walker, H. Ji, and A. Stent, Eds. Association for Computational Linguistics, 2018, pp. 1446–1459. [Online]. Available: https://doi.org/10.18653/v1/n18-1131

[6] Q. Wang and M. Iwaihara, "Deep neural architectures for joint named entity recognition and disambiguation," in *IEEE International Conference on Big Data and Smart Computing, BigComp 2019, Kyoto, Japan, February 27 - March 2, 2019*. IEEE, 2019, pp. 1–4. [Online]. Available: https://doi.org/10.1109/BIGCOMP.2019.8679233

[7] T. H. Nguyen, A. Sil, G. Dinu, and R. Florian, "Toward mention detection robustness with recurrent neural networks," *CoRR*, vol. abs/1602.07749, 2016. [Online]. Available: http://arxiv.org/abs/1602.07749

[8] Y. Luo and H. Zhao, "Bipartite flat-graph network for nested named entity recognition," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds. Association for Computational Linguistics, 2020, pp. 6408–6418. [Online]. Available: https://doi.org/10.18653/v1/2020.acl-main.571

[9] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," *CoRR*, vol. abs/1508.01991, 2015. [Online]. Available: http://arxiv.org/abs/1508.01991

[10] T. Gui, Y. Zou, Q. Zhang, M. Peng, J. Fu, Z. Wei, and X. Huang, "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 2019, Hong Kong, China, November 3-7, 2019*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Association for Computational Linguistics, 2019, pp. 1040–1050. [Online]. Available: https://doi.org/10.18653/v1/D19-1096

[11] Y. Zhang and J. Yang, "Chinese NER using lattice LSTM," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, I. Gurevych and Y. Miyao, Eds. Association for Computational Linguistics, 2018, pp. 1554–1564. [Online]. Available: https://aclanthology.org/P18-1144/

[12] B. Y. Lin, D.-H. Lee, M. Shen, R. Moreno, X. Huang, P. Shiralkar, and X. Ren, "TriggerNER: Learning with entity triggers as explanations for named entity recognition," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 8503–8511. [Online]. Available: https://aclanthology.org/2020.acl-main.752

[13] W. Li, S. Li, S. Ma, Y. He, D. Chen, and X. Sun, "Recursive graphical neural networks for text classification," *ArXiv*, vol. abs/1909.08166, 2019.

[14] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," *ArXiv*, vol. abs/1809.05679, 2019.

[15] F. Lei, X. Liu, Z. Li, Q. Dai, and S. Wang, "Multihop neighbor information fusion graph convolutional network for text classification," *Mathematical Problems in Engineering*, vol. 2021, pp. 1–9, 2021.

[16] X. Liu, X. You, X. Zhang, J. Wu, and P. Lv, "Tensor graph convolutional networks for text classification," *ArXiv*, vol. abs/2001.05313, 2020.

[17] L. Huang, D. Ma, S. Li, X. Zhang, and H. WANG, "Text level graph neural network for text classification," 2019.

[18] Y. Luo and H. Zhao, "Bipartite flat-graph network for nested named entity recognition," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds. Association for Computational Linguistics, 2020, pp. 6408–6418. [Online]. Available: https://doi.org/10.18653/v1/2020.acl-main.571

[19] S. Zheng, F. Wang, H. Bao, Y. Hao, P. Zhou, and B. Xu, "Joint extraction of entities and relations based on a novel tagging scheme," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, R. Barzilay and M. Kan, Eds. Association for Computational Linguistics, 2017, pp. 1227–1236. [Online]. Available: https://doi.org/10.18653/v1/P17-1113

[20] L. Yao, C. Mao, and Y. Luo, "KG-BERT: BERT for knowledge graph completion," *CoRR*, vol. abs/1909.03193, 2019. [Online]. Available: http://arxiv.org/abs/1909.03193

[21] W. Li, S. Li, S. Ma, Y. He, D. Chen, and X. Sun, "Recursive graphical neural networks for text classification," *CoRR*, vol. abs/1909.08166, 2019. [Online]. Available: http://arxiv.org/abs/1909.08166

[22] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, A. Moschitti, B. Pang, and

W. Daelemans, Eds. ACL, 2014, pp. 1532–1543. [Online]. Available: https://doi.org/10.3115/v1/d14-1162

[23] E. F. Tjong Kim Sang and S. Buchholz, "Introduction to the CoNLL-2000 shared task chunking," in *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*, 2000. [Online]. Available: https://aclanthology.org/W00-0726

[24] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, 2003, pp. 142–147. [Online]. Available: https://aclanthology.org/W03-0419

[25] N. Collier and J. Kim, "Introduction to the bio-entity recognition task at JNLPBA," in *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications, NLPBA/BioNLP 2004, Geneva, Switzerland, August 28-29, 2004*, N. Collier, P. Ruch, and A. Nazarenko, Eds., 2004. [Online]. Available: https://aclanthology.org/W04-1213/

[26] M. Habibi, L. Weber, M. L. Neves, D. L. Wiegandt, and U. Leser, "Deep learning with word embeddings improves biomedical named entity recognition," *Bioinform.*, vol. 33, no. 14, pp. i37–i48, 2017. [Online]. Available: https://doi.org/10.1093/bioinformatics/btx228

[27] J. Li, Y. Sun, R. J. Johnson, D. Sciaky, C. Wei, R. Leaman, A. P. Davis, C. J. Mattingly, T. C. Wiegers, and Z. Lu, "Biocreative V CDR task corpus: a resource for chemical disease relation extraction," *Database J. Biol. Databases Curation*, vol. 2016, 2016. [Online]. Available: https://doi.org/10.1093/database/baw068

[28] C. Zheng, Y. Cai, J. Xu, H. Leung, and G. Xu, "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 2019, Hong Kong, China, November 3-7, 2019*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Association for Computational Linguistics, 2019, pp. 357–366. [Online]. Available: https://doi.org/10.18653/v1/D19-1034

[29] J. Yu, B. Bohnet, and M. Poesio, "Named entity recognition as dependency parsing," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds. Association for Computational Linguistics, 2020, pp. 6470–6476. [Online]. Available: https://doi.org/10.18653/v1/2020.acl-main.577

[30] X. Wang, Y. Jiang, N. Bach, T. Wang, Z. Huang, F. Huang, and K. Tu, "Automated concatenation of embeddings for structured prediction," 2021.

[31] D.-H. Lee, R. Khanna, B. Y. Lin, J. Chen, S. Lee, Q. Ye, E. Boschee, L. Neves, and X. Ren, "Lean-life: A label-efficient annotation framework towards learning from explanation," 2020.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html