Nested Named Entity Recognition as Building Local Hypergraphs

Yukun Yan,[1,2] **Bingling Cai,**[1,2] **Sen Song**[1,2*]

[1]Biomedical Department, Tsinghua University
[2]Laboratory of Brain and Intelligence, Tsinghua University
{yanyk13, caibl13}@mails.tsinghua.edu.cn, songsen@tsinghua.edu.cn

Abstract

Named entity recognition is a fundamental task in natural language processing. Based on the sequence labeling paradigm for flat named entity recognition, multiple methods have been developed to handle the nested structures. However, they either require fixed recognition order or introduce complex hypergraphs. To tackle this problem, we propose a novel model named **Local Hypergraph Builder Network (LHBN)** that builds multiple simpler local hypergraphs to capture named entities instead of a single complex full-size hypergraph. The proposed model has three main properties: (1) The named entities that share boundaries are captured in the same local hypergraph. (2) The boundary information is enhanced by building local hypergraphs. (3) The hypergraphs can be built bidirectionally to take advantage of the identification direction preference of different named entities. Experiments illustrate that our model outperforms previous state-of-the-art methods on four widely used nested named entity recognition datasets: ACE04, ACE05, GENIA, and KBP17. The code is available at https://github.com/yanyk13/local-hypergraph-building-network.git.

Introduction

Named Entity Recognition (NER) plays an important role in natural language processing. It provides information for many downstream applications like coreference resolution, entity linking, and event extraction. Previous tagging methods have achieved significant successes in flat named entity recognition by formulating it as assigning a tag to each token. However, such a paradigm fails to handle overlapping named entities, as shown in Figure 1, which are very common in many fields.

In recent years, researchers have proposed multiple new methods to tackle the nested structure. The span-based approaches (Sohrab and Miwa 2018; Tan et al. 2020) follow a two-stage strategy. They first propose a certain number of spans, where nesting is allowed, and then predict the category for each candidate. Latest sequence labeling methods (Alex, Haddow, and Grover 2007; Ju, Miwa, and Ananiadou 2018; Wang et al. 2020) leverage multiple decoding layers to separately predict entities with different lengths following a certain order, like inner to outer. Other approaches (Lu and Roth 2015; Muis and Lu 2017; Katiyar and Cardie 2018; Iwakura, Takamura, and Okumura 2011) attempt to capture named entities by building special structures.

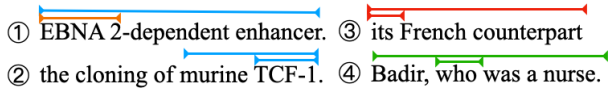

Figure 1: Nested named entities extracted from ACE04 and GENIA. The nested structures are caused by: ① A named entity as the qualifier of another one. ② Different length of qualifiers. ③ ④ Relatives or pronouns.

Dataset	Nested Entity Pairs (%)		
	Left	Right	Total
ACE04	45.67	19.77	65.44
ACE05	30.05	28.77	58.82
GENIA	75.03	14.62	89.65
KBP17	32.61	30.53	63.14

Table 1: The proportion of nested entity pairs that share a boundary. The second and third columns separately indicate the pairs that share left and right boundaries.

To address the nested structure, we take a closer look at the overlapping entities and propose a novel paradigm that use the sequence labeling method to form a local hypergraph structure. Normally, a named entity consists of a headword and its qualifiers. There are two kinds of nested structures: (1) An entity is a qualifier of another one, like *'EBNA2'* and *'EBNA2-dependent enhancer'*, and (2) The headword forms multiple named entities with qualifiers of different lengths, like *'human T cell'* and *'T cell'*. They are likely to share one side of the boundary, as shown in Table 1. In some datasets like ACE04 and ACE05, due to the requirement of downstream applications, such as event extraction and coreference resolution, some pronouns and the relatives in attributive clauses are also annotated as named entities. This leads to the third kind of nested structure, like *'those who worked for her'* and *'who'*. As for the first two kinds of structures, it requires a sequence labeling model to not only recognize a named entity but also estimate the possibility of extending its boundary to form a new one. On the other hand, the identification of the inner entities of the last kind can be regarded

*Corresponding Author.

as a token-wise classification task.

We propose a novel model, named **Local Hypergraph Building Network (LHBN)**, as illustrated in Figure 3 to formulate the identification of named entity as a two-stage process: (1) Proposing boundary token candidates of entities with a token-wise classifier, (2) Given a left/right boundary, sequentially casting the subsequence into a local hypergraph. By the word 'local', we emphasize that each hypergraph only captures the entities that share a specific boundary. Compared to previous hypergraph structures that are required to encapsulate all the entities, ours has much lower complexity and thus is easier to build. Although we also use a sequence-to-sequence paradigm, different from previous sequence labeling methods, our method does not require a subjectively defined recognition order, like bottom-up, and the decoding layer has no preference for entity length.

Our main contributions are as follows:

- We introduce a novel paradigm based on a local hypergraph for nested named entity recognition. The construction process is much simpler than mapping the full text into a single hypergraph as used in previous studies.
- Instead of identifying named entities with a subjectively defined order or a preference for entity length, our method extracts entities with the same boundary in a local hypergraph, which makes it easier to synthesize the information in nested structures.
- We evaluate our methods on four widely used nested named entity recognition datasets: ACE04, ACE05, GENIA, and KBP17. Experiments show that the proposed method achieves state-of-the-art performance on all four datasets.
- By extending our methods to bidirectionally build local hypergraphs, we achieve superior performances on all four datasets. In addition, we illustrate that there is a preference in identifying order for different nested structures.

Related Work

There are various paradigms for nested named entity recognition (NER). We can roughly divide them into span-based methods, hypergraph-based methods, sequence labeling methods, and other approaches.

Span-based Methods. The span-based approaches are the most mainstream way for nested NER. Generally, they first propose a certain number of span candidates and then classify them into different categories. These studies focus on span sampling strategies and span representation methods. As an early attempt, Exhaustive Model (Sohrab and Miwa 2018) samples all possible spans. Tan et al. (2020) leverages sub-modules to estimate boundaries before sampling spans. Shen et al. (2021) introduces a new module to adjust boundaries of span candidates to further use boundary information. Fu et al. (2021) uses a TreeCRF to enhance the interactions between nested spans. The most recent method Yuan et al. (2022) proposes tri-affine mechanism to integrate all useful information of different formats including tokens, labels, boundaries, and related spans to enhance the span representation. However, there is a trade-off between sampling span candidates and cross-span attention: (1) A small number of span candidates brings a risk of omitting entities. (2) With a large sampling number, the cross-span attention tends to be noneffective for involving too many lower qualified spans, and it requires a large computation resource.

Sequence Labeling Methods. Previous sequence labeling methods for Nested NER(Alex, Haddow, and Grover 2007; Luo and Zhao 2020; Wang et al. 2020; Shibuya and Hovy 2020) use multiple decoding layers to handle nested structures. The identification of entities in these methods follows a certain order, such as from inner to outer or from bottom-up, which is difficult to learn because the labels of entities are unordered.

Hypergraph-based Methods. Hypergraph-based models did not achieve competitive performance in recent years. Most previous hypergraph-based methods, like that Lu and Roth (2015) proposed, are rule-based, and they attempt to map a text into carefully designed hypergraphs to capture all possible nested structures. Although other works (Muis and Lu 2017; Katiyar and Cardie 2018) leverage a hypergraph to transform Nested NER to a modified sequence labeling task as we do in this work, they use a single structure to represent all the named entities in the input sentence. Due to the complexity of the hypergraph, these models are difficult to train. This is described in detail in the Method Section.

Other Methods. Tan et al. (2021) provides a fixed set of learnable vectors to learn the patterns of the valuable spans. Both Li et al. (2020) and Shen et al. (2022) use a machine reading comprehension framework to identify entities.

To our best knowledge, we are the first to formulate nested named entities recognition as building multiple local hypergraphs via a sequence labeling method.

Method

In this section, we first introduce the proposed local hypergraph structure and then our task formulation. Figure 3 illustrates an overview of the proposed method.

Encoding Scheme

Using the standard BIEO tag scheme, the desired tag sequences of three overlapping named entities (*'peripheral blood mononuclear cell'*, *'peripheral blood mononuclear cell glucocorticoid receptors'*, and *'glucocorticoid receptors'*) are shown in Figure 2(a), where a nested structure requires different types of labels at the same position.

To tackle this problem, several previous works (Lu and Roth 2015; Muis and Lu 2017; Katiyar and Cardie 2018) introduce directed hypergraphs that encode the token-level tags for all entities in the input sentence. A directed hypergraph is very similar to a standard directed graph except that its edges (hyperarcs) can connect more than one head node and tail node. Specifically, in a hypergraph representing named entities, there are two types of hyperarcs: normal edges that connect a single head node and tail node, and hyperedges that connect a set of head nodes and tail nodes. As illustrated in Figure 2(b), the hyperedges are introduced to encode nested structures.

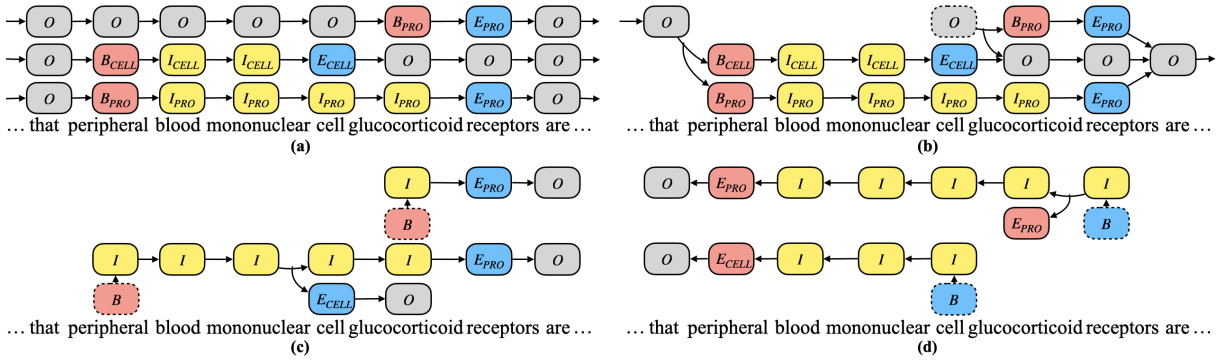

Figure 2: Tag scheme and hypergraphs for three overlapping entities. (a): Entities in an unfolded hypergraph. Each row corresponds to a tag sequence using standard BIEO tagging scheme. (b): Hypergraph structure introduced in (Katiyar and Cardie 2018). (c): Forward local hypergraphs. (d): Backward local hypergraphs. Curved edges represent hyperedges and straight edges are normal edges in (b)(c)(d).

Previous hypergraph-based methods use a single hypergraph to encode all the entities in the input sentence. To build such a complex structure, they have to either extend label types (Muis and Lu 2017) or generate hyperedges as a multi-label problem (Katiyar and Cardie 2018), which increases the difficulty of modeling. To alleviate this problem and leverage the aforementioned syntactic features of named entities, we propose a novel local hypergraph structure. Instead of representing all the name entities in the input sentence, the proposed local hypergraph only encodes the entities that share the same boundary, as illustrated in Figure 2(c)(d). Each local hypergraph has a generation direction (forward/backward) indicating whether built from a left or right boundary. Our tagging scheme is similar to the standard BIEO scheme, except that the $\boldsymbol{B}$ and $\boldsymbol{I}$ are not type-specific. The following four types of nodes are used:

- $\boldsymbol{B}$ denotes the left/right boundary token of one or more named entities in a forward/backward local hypergraph.
- $\boldsymbol{I}$ denotes a token inside any named entity.
- $\boldsymbol{E_{cls_X}}$ denotes the right/left boundary token of a named entity, whose type is cls_X in a forward/backward local hypergraph.
- $\boldsymbol{O}$ denotes a token outside of any named entities.

Nodes and hyperarcs are used together to encode named entities: A path from a $\boldsymbol{B}$ node to a $\boldsymbol{E_{cls_X}}$ node through any numbers of $\boldsymbol{I}$ nodes represent a named entity (type cls_X) in a local hypergraph. We perform depth-first searches for such paths on all the local hypergraphs to decode named entities. It is important to note that both forward and backward local hypergraph sets can encode all named entities in the input sentence independently.

Task Formulation

As shown in Figure 3, we formulate the recognition of named entities as a two-stage process: boundary proposing and hypergraph building, as described below.

Boundary Proposing. Given an input sentence $X = \{x_1, x_2, ..., x_N\}$ consisting of N tokens, the model is required to propose a set of left boundary candidates $X^L = \{x_{l_i}\}$ and a set of right boundary candidates $X^R = \{x_{r_j}\}$ as initial $\boldsymbol{B}$ nodes of local hypergraphs, where l_i is the index of the i-th left boundary, and r_j is the index of the j-th right boundary. We regard this process as token-wise binary classification.

Hypergraph Building. Given an input sentence $X = \{x_1, x_2, ..., x_N\}$ and a left boundary candidate x_{l_i}, a forward local hypergraph is build by sequentially assigning tags to each token of the subsequence $\{x_{l_i}, x_{l_i+1}, ..., x_N\}$. Similarly, a backward local hypergraph corresponding to a right boundary x_{r_j} is built by tagging the reversed subsequence $\{x_{r_j}, x_{r_j-1}, ..., x_1\}$.

Model Structure

In general, there are two modules in our model, referred to as **Boundary Detector** and **Hypergraph Builder**. The boundary detector first proposes potential boundaries of named entities in the input sentence. Then, the hypergraph builder generates a local hypergraph for each boundary and its subsequence. In this section, we illustrate the detailed structure of the proposed model, and the specific algorithm to build local hypergraphs.

Boundary Proposer

The boundary proposer consists of a text encoder and a feedforward layer. As a token-wise binary classifier, it estimates the probability that each token in the input sentence is a boundary of any named entities. We use the same method as (Shen et al. 2021) proposed to generate token representation. Given a sentence $X = \{x_1, x_2, ..., x_N\}$ with N tokens, the representation of the i-th token is built by first concatenating four components: (1) the encoded result t_i^{lm} of a language model; (2) word embedding t_i^w; (3) part-of-speech (POS) embedding t_i^{pos}; and (4) character-level embedding t_i^{char} which is generated by a char-level BiLSTM. Then, we feed them into a token-level BiLSTM layer and regard its hidden states as the token representation t_i. Finally, the feedforward layer takes t_i as input and output a score $p_i^{l/r}$ to

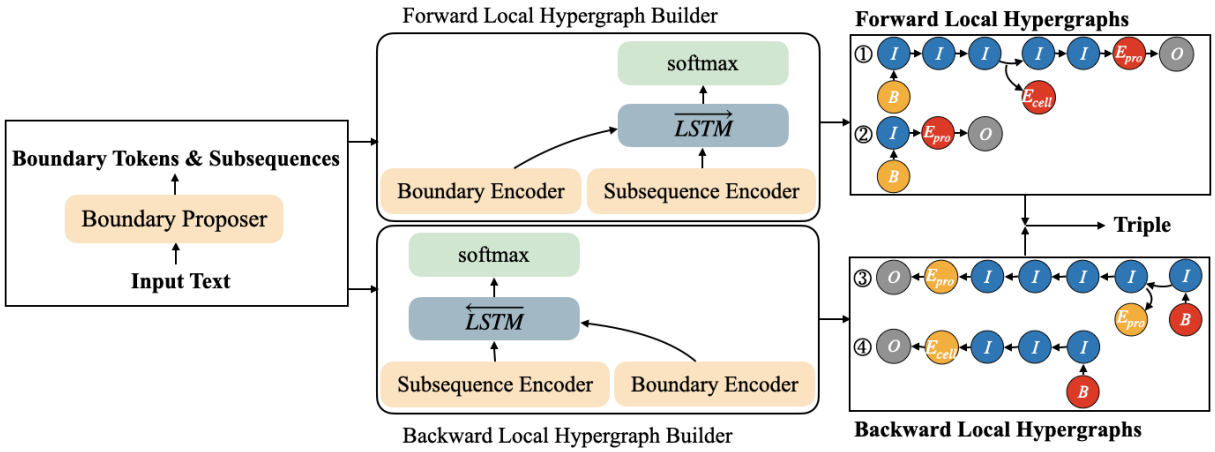

Figure 3: Overview of the proposed method and model structure. Our model first proposes boundary candidates and their corresponding subsequences. Then, they are feed into a sequence labeling module to generate a set of local hypergraphs in both directions, which are decoded to extract named entities. The results in two directions are merged to gain the final predictions. A unidirectional prediction is indicated as a quadruple (left boundary, right boundary, entity type, score). A final prediction is indicated as a triple (left boundary, right boundary, entity type).

evaluate the likelihood of i-th token being a left/right boundary.

$$z_i = t_i^{lm} \oplus t_i^{w} \oplus t_i^{pos} \oplus t_i^{char} \quad (1)$$

$$\overrightarrow{h_i} = \overrightarrow{LSTM}([z_0, z_1, ..., z_N]) \quad (2)$$

$$\overleftarrow{h_i} = \overleftarrow{LSTM}([z_0, z_1, ..., z_N]) \quad (3)$$

$$t_i = \overrightarrow{h_i} \oplus \overleftarrow{h_i} \quad (4)$$

$$p_i^{l/r} = sigmoid(FFN(t_i)) \quad (5)$$

Similar to the proposing strategy of span-based methods, we prefer to obtain a high-recall sampling result in proposing boundary candidates. Thus, we use a focal loss(Lin et al. 2017) $L^{l/r}$ as our target function to minimize.

$$L^{l/r} = \sum_{i=1}^{N} -y_i^{l/r} \cdot (1 - p_i^{l/r})^{\gamma} log(1 - p_i^{l/r}), \quad (6)$$

where $y_i^{l/r}$ is the ground label of the i-th token. The hypergraph builder should be able to drop low-qualified boundaries. Thus, during training, we scale the number of token candidates by a factor of λ to include some low-qualified boundary candidates.

Hypergraph Builder

As aforementioned, we can represent all the named entities in the input sentence as a collection of local hypergraphs in a single direction. In practice, we separately generate local hypergraphs in both directions and then merge the results to get the final predictions. To simplify the description of the model structure, we illustrate the proposed method by taking the process of building a local hypergraph forward as an example in the rest of this section.

The hypergraph builder has two text encoders which has the same structure of the one in the boundary proposer that separately generate the representation of a left boundary x_{l_i} and corresponding subsequence $\{x_{l_i}, x_{l_i+1}, ...x_N\}$, indicated as $t_{l_i}^b$ and $\{t_{l_i}^s, t_{l_i+1}^s, ...t_N^s\}$.

We use a standard unidirectional LSTM layer to generate a local hypergraph structure. Its cell memory $C_{i,k}$ and hidden state $h_{i,k}$ are initialized with $t_{l_i}^b$ instead of zero vectors to enhance the boundary information. Taking the subsequence as input, it produces a tag at each time step as a multi-class problem, and then extends the local hypergraph using the following rules R: (1) The local hypergraph is initialized as a single $\boldsymbol{B}$ node. (2) At each time step, only the last $\boldsymbol{B}$/$\boldsymbol{I}$ node is regarded as the head node. (3) If a cls_X tag is predicted, an $\boldsymbol{I}$ node is connected to the head node by a hyperedge together with a $\boldsymbol{E_{cls_X}}$ node. (4) When an $\boldsymbol{I}$/$\boldsymbol{O}$ tag is predicted, an $\boldsymbol{I}$/$\boldsymbol{O}$ node will be connected to the head node with a normal edge. (5) Once an $\boldsymbol{O}$ node is predicted, the local hypergraph stops growing.

$$C_{i,0} = h_{i,0} = t_{l_i}^b \quad (7)$$

$$h_{i,k+1} = LSTM(t_{l_i+k}^s, h_{i,k}) \quad (8)$$

$$p_{i,k} = \text{softmax}(FFN(h_{i,k})) \quad (9)$$

$$\hat{y}_{i,k} = \arg\max_c(p_{i,k}) \quad (10)$$

$$\hat{Y}_i = \{\hat{y}_{i,0}, \hat{y}_{i,1}, ..., \hat{y}_{i,N-l_i}\} \quad (11)$$

$$G_i^f = R(\hat{Y}_i), \quad (12)$$

where $k \in \{0, 1, ..., N - l_i\}$. $p_{i,k}$ is the predicted probability of tags for the k-th token in the subsequence corresponding to the i-th left boundary x_{l_i}, and $\hat{y}_{i,k}$ is its predicted tag. G_i^f is the i-th forward local hypergraph generated by implementing R on the tag sequence $\hat{Y}_i$.

We then decode the named entities from the local hypergraph by implementing a depth-first search for the paths that start with a $\boldsymbol{B}$ node and end at a $\boldsymbol{E_{cls_X}}$ node. We use a set of quadruples $E^{f/b} = \{(idx_l, idx_r, cls, p^{f/b})\}$ to represent

forward/backward predicted named entities, where idx_l and idx_r are the boundary indexes, cls is the predicted entity type, and p is the predicted probability of the last node being $\boldsymbol{E_{cls}}$.

We use cross-entropy L^g as the target function to minimize while training the hypergraph builder. We limit the length of a tag sequence to only contain at most one O tag to save computation resources since the generation stops growing. Suppose the i-th tag sequence has a length of V_i:

$$L^g = -\frac{1}{M}\sum_{i=1}^{M}\frac{1}{V_i}\sum_{k=1}^{V_i}\sum_{c\in\xi}(\mathbb{1}[y_{i,k}=c]\log(p^c_{i,k}) + \mathbb{1}[y_{i,k}\neq c]\log(1-p^c_{i,k})), \xi = \{I, O, cls_1, cls_2, ...\} \tag{13}$$

where ξ is the set of node types. $y_{i,k}$ is the ground truth type of the k-th token in the subsequence starts with the i-th left boundary. $p^c_{i,k}$ is the predicted probability of the k-th token in corresponding subsequence being tagged as type c.

We observe that the order of the combination of qualifiers and headwords in named entities is very diverse, leading to differences in their ease of recognition from different directions. Therefore, we first separately identify named entities in both directions. The sets of named entities E^f and E^b are then merged to form the final predictions. Specifically, we compute an average score p^m for each entity candidate (idx_l, idx_r, cls) in the sets as below

$$p^m = \frac{1}{2}(\mathbb{1}[(idx_l, idx_r, cls, p^f) \in E^f]p^f + \mathbb{1}[(idx_l, idx_r, cls, p^b) \in E^b]p^b) \tag{14}$$

A proposed entity candidate is dropped if $p^m < \theta$, where θ is a hyper-parameter as a threshold.

Experiments

To evaluate the proposed method, we conduct experiments on four widely used datasets for Nested NER: ACE04, ACE05, KBP17 and GENIA.

ACE04 and ACE05(Doddington et al. 2004; Stephanie Strassel and Maeda 2006) are nested datasets with 7 entity categories, we use the same setup as previous works(Katiyar and Cardie 2018; Shen et al. 2021) and split them into train, dev, and test sets by 8:1:1.

GENIA(Ohta et al. 2002) is a nested dataset consisting of biology texts. There are 5 entity types: DNA, RNA, protein, cell line and cell categories. Following (Shen et al. 2021), we use a 90%/10% train/test split.

KBP17(Ji et al. 2017) has 5 entity categories. We split all the samples into 866/20/167 documents for train/dev/test set following the same setup as previous works(Shen et al. 2021).

Evaluation Metrics

We employ precision, recall, and F1-score to evaluate the performance. Here we use strict evaluation metrics that an entity is considered correctly labeled only if its boundary and category are correct simultaneously.

Parameter Setting

In the experiments on **ACE04**, **ACE05** and **KBP17**, we leverage BERT-large(Devlin et al. 2019) and GloVe(Pennington, Socher, and Manning 2014) to initialize our encoders. The dimensions for t_i^{lm}, t_i^w, t_i^{pos}, t_i^{char}, and t_i are 1024, 300, 512, 1024, and 1024, respectively. We replace BERT and GloVe with BioBERT-large(Lee et al. 2020) and BioWordvec(Chiu et al. 2016) for **GENIA**. Corresponding dimensions of t_i^{lm}, t_i^w are 1024, 200. Based on the performance on the dev sets of ACE04, ACE05, and KBP17, γ used in equation (6) is set to 0.9, the scale hyper-parameter λ for sampling boundary candidates is set to 5, and the merging threshold θ is set to 0.5. For all the experiments, we train our model for 100 epochs with an AdamW optimizer and a linear warmup-decay learning rate. The initial learning rate for BERT modules and other parameters are set to 1e-5, and 1e-3 respectively.

Baselines

We compare our method with several state-of-the-art approaches, including span-based, hypergraph-based, sequence labeling, and other methods, on ACE04, ACE05, GENIA, and KBP17 datasets:

- Katiyar and Cardie (2018) make use of the BILOU tagging scheme to learn the hypergraph representation.
- Luo and Zhao (2020) proposes a bipartite flat-graph network with two interacting subgraph modules.
- Wang et al. (2020) designs the normal and inverse pyramidal structures to identify entities through bidirectional interactions.
- Shen et al. (2021) proposes a two-stage entity identifier to maintain high-quality span candidates.
- Tan et al. (2021) provides a fixed set of learnable vectors to learn the patterns of the valuable spans.
- Yuan et al. (2022) proposes a novel tri-affine mechanism including tri-affine attention and scoring.
- Shen et al. (2022) proposes Parallel Instance Query Network (PIQN), which sets up global and learnable instance queries to extract entities from a sentence in a parallel manner.

Result and Discussion

Main Result

The performance of the proposed method and baselines is shown in Table 2 on all four datasets. To make it fair, we do not compare the proposed approach to those based on extra-large pre-trained models. Our method outperforms all the state-of-the-art models on all nested named entity recognition datasets, achieving F1-scores of 88.46%, 87.82%, 82.09%, and 86.55% on ACE04, ACE05, GENIA, and KBP17 with +0.32%, +0.41%, +0.32% and +1.05% improvements, respectively.

We believe that our method outperforms previous approaches for the following reasons: (1) The boundary information plays an important role in named entity recognition, especially those contained in the same nested structure. LHBN enhances this information by initializing a lo-

	Model	Pr.	Rec.	F1
ACE04	Katiyar and Cardie (2018)	73.60	71.80	72.70
	Wang et al. (2020)	86.08	86.48	86.28
	Tan et al. (2021)	88.46	86.10	87.26
	Yuan et al. (2022)	87.13	87.68	87.40
	Shen et al. (2021)	88.24	86.82	87.52
	Shen et al. (2022)	88.48	87.81	88.14
	LHBN	**88.78**	**88.13**	**88.46**
ACE05	Katiyar and Cardie (2018)	72.70	70.60	70.50
	Luo and Zhao (2020)	75.00	75.20	75.10
	Wang et al. (2020)	83.95	85.39	84.66
	Yuan et al. (2022)	86.70	86.94	85.50
	Shen et al. (2021)	86.09	87.27	86.67
	Tan et al. (2021)	**87.48**	86.63	87.05
	Shen et al. (2022)	86.27	88.60	87.42
	LHBN	86.76	**88.93**	**87.83**
GENIA	Katiyar and Cardie (2018)	79.80	68.20	73.60
	Luo and Zhao (2020)	77.40	74.60	76.00
	Wang et al. (2020)	80.33	78.31	79.31
	Tan et al. (2021)	82.31	78.66	80.44
	Shen et al. (2021)	80.19	80.89	80.54
	Yuan et al. (2022)	80.42	82.06	81.23
	Shen et al. (2022)	**83.24**	80.35	81.77
	LHBN	81.17	**83.03**	**82.09**
KBP17	Luo and Zhao (2020)	77.10	74.30	75.60
	Tan et al. (2021)	84.91	83.04	83.96
	Shen et al. (2021)	85.46	82.67	84.05
	Yuan et al. (2022)	86.50	83.65	85.50
	Shen et al. (2022)	85.67	83.37	84.50
	LHBN	**86.84**	**86.25**	**86.55**

Table 2: Main result.

cal sequence labeling task for each boundary and sequentially generating a local hypergraph, whereas the usage of boundaries is limited to the assisted scoring method for span candidates. (2) The tags of named entities are essentially an unordered set. However, previous sequence labeling methods set restrictions on the recognition order, like inner-outer, which is not required in our method. (3) The local hypergraphs can be built forward and backward to handle different types of nested structures.

Ablation Study

#	Model	ACE04		
		Pr.	Rec.	F1
1	default	**88.78**	**88.13**	**88.46**
2	Forward Only	88.33	87.84	88.09
3	Backward Only	88.50	86.52	87.50
4	w/o Individual Encoder	88.42	87.05	87.73
5	w/o Boundary Initialization	88.37	87.67	88.02

Table 3: Ablation result. w/o Forward/Backward only denotes a unidirectional model. w/o Individual Encoder refers to a reduced model that has no separate encoder for boundary tokens. w/o Boundary Initialization indicates that the LSTM layer of the hypergraph builder is initialized with zero vectors.

We conduct several ablation experiments to elucidate important designs of the proposed method on **ACE04**, and the results are shown in Table 3.

Effectiveness of Bidirectional Prediction. There are various syntactic structures of named entities, which brings differences in the difficulty of identifying them from forward and backward. For example, if an entity starts with a headword, and ends with a long attributive clause, like "The lawmaker who served as chief of staff to late President Roh Moohyun". It is simple to process forward, first identifying the head word and then sequentially tagging the qualifiers. In contrast, it is hard to recognize 'Moohyun' as a right boundary. In the proposed method, the final result is generated by merging predictions from both directions. Compared to unidirectional results, as shown in line 2 (forward) and line 3 (backward), it brings consistent improvements.

Effectiveness of Individual Encoder for Boundary Tokens. In this work, we leverage separate encoders to generate the representation of boundary tokens and subsequences. Comparing line 4 and line 1 in Table 4, one could observe that the separate encoders bring +0.73% improvement in F1-scores on ACE04. One explanation could be that the semantic information of a token is different when it is regarded at the boundary of or inside a named entity.

Effectiveness of Boundary Initialization. In our method, the LSTM layer is initialized with each boundary token, which is similar to some question-answering formulations. The building of each local hypergraph is regarded as a query-based process: given a boundary token as a query, the model is required to predict the named entities in the subsequence as the answer. Compared to the reduced version that uses zero vectors to initialize the cell memory of the LSTM unit, it achieves +0.44% improvement of F1-scores, as shown in line 1 and line 5. We believe that the boundary-specific initialization enhances the different context information for different local hypergraphs.

Analysis

Directional Preference. To further study how bidirectional merging improves the performance, we count the contribution of forwarding and backward prediction to the overall true positive predictions on GENIA. The result shows that 3.44% and 5.93% true positive named entities comes from forward and backward predictions. In other words, there is a directional preference for the recognition of different named entities. Thus, merging bidirectional local hypergraphs benefits the overall performance.

Comparison to Span-based Models. For a detailed comparison between span-based methods and LHBN, we implement one of the most recent state-of-the-art models that (Shen et al. 2021) proposed. We illustrate the recognition F1-score on entities of different length ranges in Table 5. The results show that our model has significant advantages in the vast majority of length ranges, especially for the recognition of named entities with lengths longer than 15. We believe there are two main reasons: (1) Due to the limited computation resource, lots of long-span candidates are abnegated in

#	Samples	Proposed Entities	LHBN			(Shen et al. 2021)
			BI	FWD	BWD	
1	Meanwhile 70 of the 178 people , who were originally admitted to hospital , are still receiving treatment in various hospitals in the region , with 22 of them reported to still be in critical condition after the train was derailed while traveling at 190 kilometers an hour on a corner with a speed limit o 80km / h .	① $[PER_1, PER_{13}]$	✓	✓	✓	$\boldsymbol{\times}_{FN}$
		② $[PER_3, PER_{13}]$	✓	$\boldsymbol{\times}_{FN}$	✓	$\boldsymbol{\times}_{FN}$
		③ $[FAC_{12}, FAC_{13}]$	✓	✓	✓	✓
		④ $[FAC_{19}, FAC_{24}]$	✓	✓	✓	✓
		⑤ $[LOC_{22}, LOC_{24}]$	✓	✓	✓	✓
		⑥ $[PER_{26}, PER_{29}]$	✓	✓	✓	✓
		⑦ $[PER_{28}, PER_{29}]$	✓	$\boldsymbol{\times}_{FN}$	✓	✓
		⑧ $[PER_1, PER_6]$				$\boldsymbol{\times}_{FP}$
		⑨ $[PER_3, PER_6]$				$\boldsymbol{\times}_{FP}$
		⑩ $[PER_7, PER_8]$				$\boldsymbol{\times}_{FP}$
		⑪ $[LOC_{49}, LOC_{51}]$				$\boldsymbol{\times}_{FP}$
2	A fast track court has been set up at Saket court by the country ' s chief justice and the case will be delivered by this weekend to the fast track court .	① $[ORG_0, ORG_4]$	✓	✓	✓	✓
		② $[FAC_9, FAC_{11}]$	✓	✓	$\boldsymbol{\times}_{FN}$	$\boldsymbol{\times}_{FN}$
		③ $[GPE_{12}, GPE_{14}]$	✓	✓	✓	✓
		④ $[PER_{12}, PER_{18}]$	✓	✓	✓	✓
		⑤ $[ORG_{28}, ORG_{32}]$	✓	✓	✓	✓
		⑥ $[GPE_9, GPE_{10}]$		$\boldsymbol{\times}_{FP}$		$\boldsymbol{\times}_{FP}$
		⑦ $[ORG_9, ORG_{11}]$			$\boldsymbol{\times}_{FP}$	

Table 4: Case Study. $[cls_{index1}, cls_{index2}]$ indicates a named entity that starts at 'index1' and ends at 'index2', which belongs to type 'cls'. The third column, fourth column, and last column separately show the bidirectionally merged (BI), forward (FWD), and backward (BWD) prediction. The last column shows the predictions of the model of Shen et al. (2021). ✓, $\boldsymbol{\times}_{FN}$, and $\boldsymbol{\times}_{FP}$ separately indicates true positive, false negative, and false positive prediction.

Datasets	Entity length	F1		support
		Shen et al. (2021)	**LHBN**	
ACE04	1-4	89.10	**89.75**	2612
	5-9	82.35	**84.52**	309
	10-14	**73.44**	71.56	60
	≥15	51.69	**67.31**	53
ACE05	1-4	87.71	**88.64**	2611
	5-9	83.51	**86.28**	274
	10-14	69.63	**76.26**	66
	≥15	58.46	**64.00**	40
GENIA	1-4	81.29	**82.84**	4979
	5-9	77.10	**78.20**	727
	10-14	**74.51**	72.22	44
	≥15	72.73	**85.71**	7
KBP17	1-4	85.58	**87.40**	11410
	5-9	71.64	**77.41**	899
	10-15	65.94	**73.67**	186
	≥15	54.43	**68.02**	101

Table 5: Comparison between a state-of-the-art span-based Model (Shen et al. 2021) and LHBN.

the proposing stage of a span-based method. (2) Although multiple span-based methods introduce cross-span information interaction by implementing an attention mechanism (Luo and Zhao 2020; Yuan et al. 2022), it is ineffective for that due to the pursuit of high recall in the span proposing stage, and low-quality span candidates introduce noise into the computation. In contrast, sequential generation of local hypergraphs makes it easier to process information inside nested structures. Our method also has the advantage of time complexity over span-based models. They need to classify almost all possible spans, which leads to the high computational cost with $O(N^2)$ time complexity, which will further increases if we take cross-span attention into account. For the proposed method, the time complexity of proposing boundaries is $O(N)$. Assuming that the number of entities in the sentence is q, the time complexity of building local hypergraphs is $O(qN)$. Thus, the total time complexity is $O(N + qN)$ where $q << N$.

Case Study. We provide two cases in Table 4 to show the identification results of our model. As illustrated in the first column, LHBN is capable of identifying long entities with complex nested structures. From both cases, we can see the directional preference that a named entity is easier to identify from the direction the headword starts. In addition, bidirectional prediction results can complement each other well. From the predictions of the two models, we can see that LHBN outperforms the model (Shen et al. 2021) proposed on various complex samples.

Conclusion

We propose a novel method that treats nested named entity recognition as building local hypergraphs. First, we propose a certain number of boundary candidates, and then we generate a local hypergraph for each candidate with a sequence labeling method. The local hypergraph structure avoids an overly complex construction process in previous works (Muis and Lu 2017; Katiyar and Cardie 2018). In addition, leveraging local hypergraph makes our sequence labeling module free from identifying entities in a certain order that the previous sequence labeling methods suffer from. Compared to span-based methods, our method has a lower time complexity. Our method achieves a new state-of-the-art F1-score on four widely used datasets: ACE04, ACE05, GENIA, and KBP17.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China (61836004), in part by the grant from Institute Guo Qiang, Tsinghua University, in part by Beijing Brain Science Special Project (No.Z181100001518006), in part by Tsinghua University Initiative Scientific Research Program (20197010009), in part by the IDG/McGovern Institute for Brain Research at Tsinghua University, and in part by the National Key Research and Development Program of China (Grant No.2021ZD0200300).

References

Alex, B.; Haddow, B.; and Grover, C. 2007. Recognising nested named entities in biomedical text. In *Biological, translational, and clinical language processing*, 65–72.

Chiu, B.; Crichton, G.; Korhonen, A.; and Pyysalo, S. 2016. How to Train good Word Embeddings for Biomedical NLP. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, 166–174. Berlin, Germany: Association for Computational Linguistics.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Doddington, G. R.; Mitchell, A.; Przybocki, M. A.; Ramshaw, L. A.; Strassel, S. M.; and Weischedel, R. M. 2004. The automatic content extraction (ACE) program-tasks, data, and evaluation. In *Lrec*, volume 2, 837–840. Lisbon.

Fu, Y.; Tan, C.; Chen, M.; Huang, S.; and Huang, F. 2021. Nested named entity recognition with partially-observed treecrfs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 12839–12847.

Iwakura, T.; Takamura, H.; and Okumura, M. 2011. A Named Entity Recognition Method based on Decomposition and Concatenation of Word Chunks. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, 828–836. Chiang Mai, Thailand: Asian Federation of Natural Language Processing.

Ji, H.; Pan, X.; Zhang, B.; Nothman, J.; Mayfield, J.; McNamee, P.; Costello, C.; and Hub, S. I. 2017. Overview of TAC-KBP2017 13 Languages Entity Discovery and Linking. In *TAC*.

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

Katiyar, A.; and Cardie, C. 2018. Nested Named Entity Recognition Revisited. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 861–871. New Orleans, Louisiana: Association for Computational Linguistics.

Lee, J.; Yoon, W.; Kim, S.; Kim, D.; Kim, S.; So, C. H.; and Kang, J. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4): 1234–1240.

Li, X.; Feng, J.; Meng, Y.; Han, Q.; Wu, F.; and Li, J. 2020. A Unified MRC Framework for Named Entity Recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 5849–5859. Online: Association for Computational Linguistics.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2980–2988.

Lu, W.; and Roth, D. 2015. Joint Mention Extraction and Classification with Mention Hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 857–867. Lisbon, Portugal: Association for Computational Linguistics.

Luo, Y.; and Zhao, H. 2020. Bipartite Flat-Graph Network for Nested Named Entity Recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 6408–6418. Online: Association for Computational Linguistics.

Muis, A. O.; and Lu, W. 2017. Labeling Gaps Between Words: Recognizing Overlapping Mentions with Mention Separators. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2608–2618. Copenhagen, Denmark: Association for Computational Linguistics.

Ohta, T.; Tateisi, Y.; Kim, J.-D.; Mima, H.; and Tsujii, J. 2002. The GENIA corpus: An annotated research abstract corpus in molecular biology domain. In *Proceedings of the human language technology conference*, 73–77.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Shen, Y.; Ma, X.; Tan, Z.; Zhang, S.; Wang, W.; and Lu, W. 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)*, 2782–2794. Online: Association for Computational Linguistics.

Shen, Y.; Wang, X.; Tan, Z.; Xu, G.; Xie, P.; Huang, F.; Lu, W.; and Zhuang, Y. 2022. Parallel Instance Query Network for Named Entity Recognition. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 947–961. Dublin, Ireland: Association for Computational Linguistics.

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

Sohrab, M. G.; and Miwa, M. 2018. Deep Exhaustive Model for Nested Named Entity Recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2843–2849. Brussels, Belgium: Association for Computational Linguistics.

Stephanie Strassel, C. W.; and Maeda, K. 2006. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Linguistic Data Consortium, Philadelphia*, 57.

Tan, C.; Qiu, W.; Chen, M.; Wang, R.; and Huang, F. 2020. Boundary enhanced neural span classification for nested named entity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 9016–9023.

Tan, Z.; Shen, Y.; Zhang, S.; Lu, W.; and Zhuang, Y. 2021. A Sequence-to-Set Network for Nested Named Entity Recognition. In Zhou, Z.-H., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 3936–3942. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Wang, J.; Shou, L.; Chen, K.; and Chen, G. 2020. Pyramid: A layered model for nested named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 5918–5928.

Yuan, Z.; Tan, C.; Huang, S.; and Huang, F. 2022. Fusing Heterogeneous Factors with Triaffine Mechanism for Nested Named Entity Recognition. In *Findings of the Association for Computational Linguistics: ACL 2022*, 3174–3186.