Article

HTLinker: A Head-to-Tail Linker for Nested Named Entity Recognition

Xiang Li, Junan Yang *, Hui Liu and Pengjiang Hu

College of Electronic Engineering, National University of Defense Technology, Hefei 230009, China; lix20@nudt.edu.cn (X.L.); liuhui17c@nudt.edu.cn (H.L.); pjhu12@nudt.edu.cn (P.H.)

* Correspondence: yangjunan@ustc.edu.cn

Abstract: Named entity recognition (NER) aims to extract entities from unstructured text, and a nested structure often exists between entities. However, most previous studies paid more attention to flat named entity recognition while ignoring nested entities. The importance of words in the text should vary for different entity categories. In this paper, we propose a head-to-tail linker for nested NER. The proposed model exploits the extracted entity head as conditional information to locate the corresponding entity tails under different entity categories. This strategy takes part of the symmetric boundary information of the entity as a condition and effectively leverages the information from the text to improve the entity boundary recognition effectiveness. The proposed model considers the variability in the semantic correlation between tokens for different entity heads under different entity categories. To verify the effectiveness of the model, numerous experiments were implemented on three datasets: ACE2004, ACE2005, and GENIA, with F1-scores of 80.5%, 79.3%, and 76.4%, respectively. The experimental results show that our model is the most effective of all the methods used for comparison.

Citation: Li, X.; Yang, J.; Liu, H.; Hu, P. HTLinker: A Head-to-Tail Linker for Nested Named Entity Recognition. *Symmetry* **2021**, *13*, 1596. <https://doi.org/10.3390/sym13091596>

Academic Editor: Aviv Gibali

Received: 30 June 2021

Accepted: 25 August 2021

Published: 31 August 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: named entity recognition; information extraction; sequence labeling; natural language processing

1. Introduction

Named entity recognition (NER) is a fundamental task in natural language processing, aiming to extract entities with pre-defined categories from unstructured texts. Constructing effective NER models is essential for some downstream tasks, such as entity linking [1–3], relation extraction [4–7], event extraction [8,9], and question answering [10].

Traditional NER models [11–15] are usually based on a single-layer sequence labeling approach, which often assigns one label to each token. However, not all entities in the text exist independently, and nested structures may exist between different entities. As shown in Figure 1, the entity “European” is nested inside entity “European Judicial Systems”, and “European” is part of two entities. The nested structure in entities is both realistic and will further improve the accuracy of the model in extracting entities.

In recent years, numerous models have been proposed for nested NER. For instance, hypergraph-based approaches [16–18] were proposed to deal with nested structures among entities. However, the construction of hypergraphs relies on extensive hand-crafted design. For a complex task, a common idea is to decompose it into simple modules. MGNER [19] divides NER into two steps: a detector and a classifier, which first locates the entity spans and then determines the entity category based on the span representation. However, this model ignores the boundary information of entities, resulting in inaccurate localized entity boundaries. Zheng et al. [20] exploited a boundary-aware method to precisely locate entity boundaries and then determine the entity category. However, this model ignores the connection between entity boundary identification and entity category classification.

Figure 1. Examples of nested entities from GENIA and ACE2005.

Inspired by the hierarchical boundary tagger method [6,21] and linking to another method of relation extraction [7], we constructed a head-to-tail NER model that divides nested NER into two highly correlated steps. First, all entity heads in the text are identified, the candidate entity head information is integrated and passed to the text, and then the corresponding entity tails are identified for different entity categories. For a text with N entities, $1 + N$ sequence tagging models are needed to extract entities. One operation is required to identify N entity heads, and N operations are needed to identify the corresponding entity tails of the different entity categories for each entity head. This approach has two advantages: first, identifying entities from head to tail can handle nested entities efficiently; second, a strong correlation exists between the two steps. If an error occurs in entity head recognition, the text obtains the wrong condition and will not make further mistakes in recognizing the wrong entity. In addition, the head-to-tail approach is based on the boundary identification method. The entity category is determined with the integration of entity head-to-tail information and relative position information. The head and tail of an entity exist symmetrically for the corresponding entity category. Compared with direct recognition of head-to-tail pairs, it is a more effective method of extracting information from entity boundary recognition to entity category determination. The main contributions are as follows:

- We introduce a novel head-to-tail NER model to extract entities from head to tail, which can handle the nested structure existing between different entities.
- To more accurately extract entities, the proposed model sequentially extracts entity head h , entity tail t , and entity category e . Nested NER is divided into two steps: identifying the entity heads, and then identifying the corresponding entity tails of each candidate entity head for different entity categories.
- We conducted extensive experiments on three datasets: ACE2004, ACE2005, and GENIA. The F1-scores of HTLinker are 80.5%, 79.3%, and 76.4% on the three datasets, respectively. Compared to nine other models, the proposed method obtains the best results for extracting entities from unstructured texts.

2. Problem Formulation

Given a text sequence S , the purpose of nested named entity recognition is to extract entities of pre-defined entity categories. The extracted entities (h, t, e) have three main parts: entity head h , entity tail t , and entity category e . These entity categories are all derived from a pre-defined set E . The extracted entities are valid only if the boundaries of entities (h, t) and entity categories (e) are accurately located.

Note that there may be nested entities, e.g., the ORG “the First National Bank of America” contains the GPE “America”. Previous methods locate entities in two steps: they first locate the entity location ($f(S) \rightarrow (h, t)$) and then determine the entity category ($f(S, h, t) \rightarrow (e)$). HTLinker locates the entity head first ($f(S) \rightarrow h$), and then integrates the information to locate the corresponding entity tail and entity category ($f(S, h) \rightarrow (t, e)$).

3. Method

Figure 2 illustrates the framework of HTLinker. Specifically, HTLinker consists of an encoder and a decoder. The purpose of the encoder is to convert the given text sequence into a vectorized representation. The decoder consists of three main parts: the entity head tagger, the head feature transmitter, and the entity tail tagger. First, all possible entity heads are located by the entity head tagger. For the extracted candidate entity head h , the head feature transmitter passes its features to the sequence embedding, and then the corresponding entity tail and entity category are obtained by the entity tail tagger. An example is presented in Figure 2.

Figure 2. The framework of HTLinker with an example of nested named entity recognition. In the example, the purpose of the framework is to extract all the entities from the given text "... the First National Bank of America ...". Note that there is a nested structure of "the First National Bank of America" and "America" in the given text. First, the entity head tagger locates the two candidate entity heads "the" and "America" in the text. Then, for the candidate entity head "the", its features are fused into the text sequence embedding. Finally, combining the text context information and entity head information, the entity tail tagger is able to locate the corresponding entity tail "America" to the candidate entity head "the" for entity category ORG. In addition, for the candidate entity head "America", the corresponding ("America", GPE) can be identified.

3.1. Encoder

A text sequence S of length n needs to be encoded as a vectorized representation. The encoder uses a pre-trained model BERT, which mainly consists of an N -layer transformer. The pre-trained BERT has a strong ability to characterize the semantics of the text. Based on BERT, we transform the text sequence S into sentence embedding H_S with the following procedure:

$$H_S = BERT(S), \quad (1)$$

where $H_S \in \mathbb{R}^{m \times d_{emb}}$ and d_{emb} represent the encoded token embedding dimension.

3.2. Decoder

3.2.1. Entity Head Tagger

For the sequence embedding H_S from the encoder, the possible entity heads in it are located by the sequence tagging method. The entity head tagger ignores entity category information when positioning the head of all entities in the text. Specifically, a binary head classifier is used to assign a probability value to each token, which indicates the likelihood that the token is an entity head. The sigmoid function with a value range from 0 to 1 can be used to calculate the probability. The process of the tagger is shown below:

$$p_i^{head} = \text{sigmoid}(\mathbf{h}_i^S \mathbf{W}_{head} + \mathbf{b}_{head}), \quad (2)$$

where $\mathbf{W}_{head} \in \mathbb{R}^{d_{emb} \times 1}$ and $\mathbf{b}_{head} \in \mathbb{R}^{n \times 1}$ are trainable matrices, $p_i^{head} \in (0, 1)$ denotes the probability that the i th token of the given sequence is the head of an entity, $\mathbf{h}_i^S$ is the i th token embedding in $\mathbf{H}_S$, and p_i^{head} is the i th element in $\mathbf{P}_{head}$. Then, we can determine the positions of the heads of all possible candidate entities:

$$\mathbf{I}_{head} = \text{argwhere}(\mathbf{P}_{head} \geq t_{head}), \quad (3)$$

where $\mathbf{I}_{head} \in \mathbb{R}^{1 \times m}$ and m denote the number of candidate entity heads, and t_{head} is the judgment threshold for the entity head, which can be adjusted according to the training process.

To effectively locate the entity heads, the binary cross-entropy loss is gradually minimized during the training process:

$$\mathcal{L}_{head} = - \sum_{i=1}^n \left[t_i^{head} \log p_i^{head} + (1 - t_i^{head}) \log (1 - p_i^{head}) \right], \quad (4)$$

where t_i^{head} is 1 only if the i th token is the head of an entity; otherwise, it is 0.

3.2.2. Head Feature Transmitter

For the k th candidate entity head i_k^{head} of $\mathbf{I}_{head}$, to locate its corresponding entity tails with entity categories, the information of the entity head is fused into the sequence embedding $\mathbf{H}_S$. The information of the entity head includes the position information and the semantic information of the entity head.

First, we consider a combination of sequence embedding and position embedding of a candidate entity head. The relative positions of all tokens of the sequence and the candidate entity head can be encoded as the relative positional embedding, which can be used to learn the span features of the entities:

$$\mathbf{I}_{rp} = \mathbf{I}_S - i_k^{head}, \quad (5)$$

$$\mathbf{H}_{RP} = \text{Embedding}(\mathbf{I}_{rp}), \quad (6)$$

where $\mathbf{I}_S \in \mathbb{R}^{1 \times m}$ denotes the absolute position of all tokens in the sequence, and $\mathbf{I}_{rp} \in \mathbb{R}^{1 \times m}$ denotes the position of all tokens in the sequence relative to the candidate entity head. $\mathbf{H}_{RP} \in \mathbb{R}^{m \times d_{rp}}$ is a trainable parameter, which is randomly initialized at the beginning of training.

Then, conditional layer normalization is employed to fuse the information of the candidate entity head with the sequence embedding $\mathbf{H}_S$:

$$\mu_i = \frac{1}{d_{emb}} \sum_{j=1}^{d_{emb}} h_{ij}, \quad \sigma_i^2 = \frac{1}{d_{bert}} \sum_{j=1}^{d_{bert}} (h_{ij} - \mu_i)^2, \quad (7)$$

$$\gamma = \mathbf{h}_k^{head} \mathbf{W}_\gamma, \quad \beta = \mathbf{h}_k^{head} \mathbf{W}_\beta, \quad (8)$$

$$\mathbf{h}_i^{(out)} = \gamma \times \frac{\mathbf{h}_i^{(in)} - \mu_i}{\sigma_i + \epsilon} + \beta, \quad (9)$$

$$\mathbf{H}_C = [\dots, \mathbf{h}_i, \dots] \quad (10)$$

where $\mathbf{h}_i^{(in)}$ is the i th token embedding in $\mathbf{H}_S$ and $\mathbf{h}_i^{(out)}$ is the i th token embedding in $\mathbf{H}_C$. γ and β are trainable parameters that denote the mean and standard deviation of the conditional inputs, respectively; μ_i and σ_i denote the mean and standard deviation of the token embedding $\mathbf{h}_i^{(in)}$, respectively.

Finally, by combining $\mathbf{H}_{RP}$ and $\mathbf{H}_C$, we obtain $\mathbf{H}$ containing entity header information and contextual information:

$$\mathbf{H} = \text{concat}(\mathbf{H}_{RP}, \mathbf{H}_C), \quad (11)$$

where $\mathbf{H} \in \mathbb{R}^{n \times (d_{emb} + d_{rp})}$.

3.2.3. Entity Tail Tagger

For the k th candidate entity head, we can find the corresponding entity tails and entity categories using the entity tail tagger. Specifically, a binary tail classifier is used to assign a probability value to each token, which indicates the likelihood that the token is an entity tail of different entity categories for the candidate entity head:

$$\mathbf{p}_i^{tail} = \text{sigmoid}(\mathbf{h}_i \mathbf{W}_{tail}^e + \mathbf{b}_{tail}^e), \quad (12)$$

where $\mathbf{W}_{tail}^e \in \mathbb{R}^{(d_{emb}+d_{rp}) \times l}$ and $\mathbf{b}_{tail}^e \in \mathbb{R}^{1 \times l}$ are trainable parameters, l is the number of entity categories, and $\mathbf{h}_i$ is the i th token embedding of $\mathbf{H}$. Note that $\mathbf{p}_i^{tail}$ denotes not only the entity tail but also the entity category.

To effectively identify the entity tails and entity categories corresponding to the candidate entity head, the binary cross-entropy loss is gradually minimized during the training process:

$$\mathcal{L}_{tail} = - \sum_{i=1}^n \sum_{j=1}^l \left[t_{ij}^{tail} \log p_{ij}^{tail} + (1 - t_{ij}^{tail}) \log (1 - p_{ij}^{tail}) \right], \quad (13)$$

where t_{ij}^{tail} is 1 only if the i th token in the text sequence is the head of an entity of the j th entity category in E ; otherwise, it is 0.

3.3. Joint Learning

To learn the head and tail features of the entities in an integrated manner, we back-propagate the loss entropy of locating the head and tail of the entities together:

$$\mathcal{L} = \mathcal{L}_{head} + \mathcal{L}_{tail} \quad (14)$$

To better update the gradients, the Adam [22] optimizer is employed for the model update.

4. Experiments

4.1. Datasets

To demonstrate the effectiveness of the proposed method, extensive experiments were implemented on three datasets: ACE2004, ACE2005 [23], and GENIA [24]. Table 1 presents the statistics of the three datasets. Next, we provide a brief presentation of the three datasets.

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

Statistics	ACE2004	ACE2005	GENIA
Split ratio	8:1:1	8:1:1	8.1:0.9:1
No. sentences	8488	9311	18,546
No. entities	27,747	30,944	56,870
No. avg. entity per sentence	3.3	3.3	3.1
Nested entities	45.7%	39.8%	21.6%
Avg. entity length	2.6	2.2	2.9

Two ACE datasets (ACE2004 (<https://catalog.ldc.upenn.edu/LDC2005T09> accessed on 5 June 2021) and ACE2005 (<https://catalog.ldc.upenn.edu/LDC2006T06> accessed on 5 June 2021) have been used for several natural language processing tasks, including named entity recognition [14,15], relation extraction [4–6], event extraction [8,9], etc. The data contained in ACE datasets are derived from the news domain and contain seven entity categories: *PER*, *LOC*, *ORG*, *GPE*, *VEH*, *WEA*, and *FAC*. To effectively compare the proposed method with other named entity recognition models, the dataset was divided following the methods in previous work [16]. The split of training instances, development instances, and test instances in the datasets was 8:1:1. In addition, the percentages of nested entities in ACE2004 and ACE2005 are 45.7% and 39.8%, respectively.

The GENIA (<http://www.geniaproject.org/genia-corpus/pos-annotation> accessed on 5 June 2021) dataset is generally used for tasks such as named entity recognition [14,15] and event extraction [8,9]. The data contained in the dataset are derived from the biomedical domain, containing five entity categories: *DNA*, *RNA*, *Protein*, *Cell_Line*, and *Cell_Type*. To effectively compare the proposed method with other named entity recognition models, the dataset was divided following the steps in a previous work [17]. The split of training instances, validation instances, and test instances in the dataset is 8.1:0.9:1. In addition, about 21.6% of the entities in GENIA have a nested structure.

4.2. Baselines

We compared the proposed method with the following nine methods:

- Men-Graph [17] has a novel mention separator that can capture the nested structure in the text.
- Layered-BiLSTM-CRF [25] uses a dynamic network to handle the nested entity problem by dynamically extracting the outer entities in a layered pattern.
- Stack-LSTM-Trans [26] applies stack-LSTM based on the transition method to capture the dependencies between the nested entities.
- Hyper-Graph [27] is a novel LSTM-based network that handles nested entities by constructing a task-specific hypergraph.
- Seg-Graph [18] is a novel segmental hypergraph method that is able to handle the structural ambiguity issue during inference in hypergraph methods.
- Boundary-aware [20] applies a boundary-aware approach for extracting nested entities, which mitigates the error propagation in layered NER models.
- Anchor-region networks (ARNs) [28] leverage the head-driven phrase structures for extracting nested entities.
- MGNER [19] applies a novel entity position detector to locate entities in a certain range around each token, and is able to extract nested or non-overlapping entities from unstructured texts.
- BiFlaG [29] identifies the inner entities by GCN based on identifying the outer entities to handle the nested entity issue.

4.3. Settings and Evaluation Metrics

4.3.1. Settings

Table 2 shows the hyperparameter settings for the experiments. The optimizer in the experiment was Adam. The learning rate was 1.0×10^{-5} . The maximum number of learning epochs was 80. In addition, the threshold for the entity head position judgment was set to 0.5 and the threshold for the tail position judgment was set to 0.5. The thresholds could be adjusted to achieve a balance between precision and recall. All experiments were based on Tensorflow. We implemented the experiments on an NVIDIA Tesla V100 GPU and an Intel Xeon E5-2698 CPU. To prevent over-fitting, the training is terminated when the F1-score on the development set is not improved for ten consecutive epochs. The batch size was set to 8. The training time for each epoch was 253, 288, and 469 s on ACE2004, ACE2005, and GENIA, respectively.

Table 2. The hyperparameters of HTLinker.

Hyperparameters	Value
Epoch	80
Batch Size	8
Token Embedding	768
Relative Position Embedding	20
Learning Rate	1.0×10^{-5}
Optimizer	Adam
Threshold of the Entity Head Tagger	0.5
Threshold of the Entity Tail Tagger	0.5

4.3.2. Evaluation Metrics

For a predicted entity extracted from the text, the extraction is valid only if its boundary location and entity category are the same as the gold entity. Specifically, entity head h , entity tail t , and entity category e all need to be considered in the prediction. To fairly compare the proposed method with other methods, three metrics (precision, recall, and F1-score) were used to evaluate the effectiveness of the proposed method.

4.4. Results

4.4.1. Main Results

Table 3 presents the results of the three metrics of the nine NER models: precision, recall, and F1-score. First, HTLinker achieves better results in extracting nested named entities from given texts compared with the nine baselines. Specifically, the F1-scores of HTLinker are 80.5%, 79.3%, and 76.4% on ACE2004, ACE2005, and GENIA, respectively, which are 1.0%, 4.2%, and 0.4% better compared to the baselines, respectively. Second, although the precision of HTLinker is not the highest, HTLinker better balances precision and recall, which results in HTLinker performing better on the main metric, F1-score. Third, the F1-scores are higher on dataset B compared to another boundary-based NER model, Boundary-aware [20]. Third, compared to another boundary-based NER model, Boundary-aware [20], HTLinker extracts entities better on GENIA: F1-score, precision, and recall are higher by 1.7%, 0.1%, and 3.2%, respectively. Despite considering entity boundaries as a whole, a crucial issue is that Boundary-aware cannot effectively match entity heads and tails from text sequences. Compared with identifying entity category labels using entity boundary information as a condition, HTLinker is more effective in identifying entity tails under different entity categories by inputting entity heads as conditional information.

Table 3. Precision, recall, and F1-score of different methods on the ACE2004, ACE2005, and GENIA datasets.

Method	ACE2004			ACE2005			GENIA		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
Men-Graph [17]	79.5	51.1	62.2	75.5	51.7	61.3	75.4	66.8	70.8
Hyper-Graph [27]	73.6	71.8	72.7	70.6	70.4	70.5	76.7	71.1	73.8
Seg-Graph [18]	78.0	72.4	75.1	76.8	72.3	74.5	77.0	73.3	75.1
Stack-LSTM-Trans [26]	-	-	73.3	-	-	73.0	-	-	73.9
Layered-BiLSTM-CRF [25]	-	-	-	74.2	70.3	72.2	78.5	71.3	74.7
Boundary-aware [20]	-	-	-	-	-	-	75.9	73.6	74.7
ARNs [28]	-	-	-	76.2	73.6	74.9	75.8	73.9	74.8
MGNER [19])	81.7	77.4	79.5	79.0	77.3	78.2	-	-	-
BiFlaG [29]	-	-	-	75.0	75.2	75.1	77.4	74.6	76.0
HTLinker	79.5	81.6	80.5	79.5	79.2	79.3	76.0	76.8	76.4

4.4.2. Detailed Results

Table 4 demonstrates the capability of HTLinker to extract different elements of an entity (h, t, e). First, HTLinker is able to accurately locate the head or tail of the entities from a given text. Second, HTLinker is able to locate the boundaries (h, t) of the entities well. Specifically, the F1-scores of HTLinker are 87.0%, 85.9%, and 80.2% in locating entity boundaries on the ACE2004, ACE2005, and GENIA, respectively. Finally, by observing the effectiveness of HTLinker in identifying the different elements of the entities, HTLinker achieves promising results in identifying entity boundaries.

Table 4. Precision, recall, and F1-score of HTLinker for extracting different elements of entities on the ACE2004, ACE2005, and GENIA datasets.

Elements	ACE2004			ACE2005			GENIA		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
<i>h</i>	94.0	91.6	92.8	92.7	90.2	91.4	86.6	85.4	86.0
<i>t</i>	92.7	91.0	91.9	91.2	89.4	90.3	87.3	87.1	87.2
<i>(h, t)</i>	86.2	87.7	87.0	86.4	85.5	85.9	80.1	80.4	80.2
<i>(h, t, e)</i>	79.5	81.6	80.5	79.5	79.2	79.3	76.0	76.8	76.4

Tables 5 and 6 describe the performance of HTLinker on extracting different categories of entities. The proposed model performs similarly in identifying the same category of entities on the ACE2004 and ACE2005 datasets. Specifically, the difference in the F1-scores of HTLinker on extracting the four categories of entities, *PER*, *LOC*, *ORG*, and *GPE*, is around 2%. In addition, HTLinker achieves better results on both ACE datasets when extracting the entities of *PER*. This is due to the boundary of *PER* having a clear trigger word. On GENIA, HTLinker achieves the best result in extracting the entities of *RNA* with an F1-score of 84.4% due to the existence of distinct trigger words in the entities of *RNA*. However, the model is less effective in extracting the entities of *DNA*, and the boundary of *DNA* can be accurately located while being misclassified as *RNA*. This is due to the model being able to accurately locate entity boundaries while often being disturbed by similar boundary information for both *DNA* and *RNA*.

Table 5. Precision, recall, and F1-score of HTLinker for extracting different categories of entities from the ACE2004 and ACE2005 datasets.

Category	ACE2004			ACE2005		
	Prec.	Rec.	F1	Prec.	Rec.	F1
<i>PER</i>	80.5	82.2	81.4	80.7	79.1	79.9
<i>LOC</i>	62.2	66.8	64.4	65.5	64.2	64.9
<i>ORG</i>	71.5	69.9	70.7	71.0	66.3	68.6
<i>GPE</i>	78.3	79.2	78.7	73.5	81.6	77.3
<i>VEH</i>	92.3	92.3	92.3	70.2	64.5	67.2
<i>WEA</i>	79.5	81.6	57.4	65.0	83.6	73.1
<i>FAC</i>	64.1	58.3	61.1	67.7	75.3	71.3
overall	79.5	81.6	80.5	79.5	79.2	79.3

Table 6. Precision, recall, and F1-score of HTLinker for extracting different categories of entities from the GENIA dataset.

Category	GENIA		
	Precision	Recall	F1-Score
<i>DNA</i>	68.2	71.3	69.7
<i>RNA</i>	84.8	83.9	84.4
<i>Protein</i>	73.7	75.2	74.5
<i>Cell_Line</i>	77.5	69.0	73.0
<i>Cell_Type</i>	74.6	75.9	75.3
overall	76.0	76.8	76.4

Figure 3 demonstrates the impact of the thresholds on extracting entities. The thresholds of the two taggers remained consistent in the experiments. First, as the thresholds of the two taggers increased, precision increased and recall decreased. The precision and recall for extracting entities were above 70% for the threshold range tested experimentally. Second, the F1-score increased and then decreased as the threshold was increased. When

the threshold was 0.6, the F1-score reached the highest values of 81.0% and 79.6% on the ACE2004 and ACE2005 datasets, respectively. When the threshold was 0.5, the F1-score reached the highest value of 76.4% on the GENIA dataset.

Figure 3. Precision, recall, and F1-score of extracting entities for the taggers with different thresholds.

5. Related Work

5.1. Named Entity Recognition

NER is an essential task in information extraction (IE) and has attracted the interest of numerous researchers. Early works [11–13] relied on hand-crafted features to extract entities. Hidden Markov model (HMM) [30] and conditional random field (CRF) [31] were applied in these NER models. Later, deep learning approaches were widely used. CNN-CRF [14] automatically captures the semantic features of text using a convolutional neural network (CNN), and is combined with CRF to extract entities. Then, bidirectional LSTM (BiLSTM) [15] was used for learning the semantic features of text combined with CRF to learn the correlation between entity labels. In addition to word-level representation, character-level information was considered to better use the information in the given text. LSTM [32] and CNN [33,34] were employed to learn character-level features, which overcame the out-of-vocabulary (oov) issue. However, these methods often only assign a label to each token, which ignores the nested structure that exists between entities.

5.2. Nested Named Entity Recognition

Work in recent years has paid more attention to the existence of nested structures of entities in text.

To efficiently extract nested entities, hypergraphs [16,17] were constructed to identify the nested entities in texts. However, the ambiguous structure of the hypergraph affects the effectiveness of extracting nested entities during the inference process. To overcome this issue, Seg-Graph [18] applies a segmental hypergraph structure. In addition, the construction of hypergraphs relies on hand-crafted design. To more effectively construct the hypergraph structure, Hyper-Graph [27] is a novel hypergraph model based on the BIOES tagging scheme, using LSTM to automatically learn the structure of the hypergraph.

Nested entities can also be extracted from inside to outside or from outside to inside by stacking sequence labeling models. HMM [11,12,35] and SVM [36] were employed to construct multi-layer sequence labeling models to extract nested entities. However, these methods extract outside and inside entities independently, ignoring the dependencies between nested entities. Alex et al. [37] constructed two modules based on the CRF sequence labeling model and cascaded CRFs from inside to outside and from outside to inside to enhance the correlation between nested entities. However, one issue is that this model struggles to handle nested entities of the same category. BiLSTM-CRF [34] is a stable and effective model for flair NER, so Ju et al. [25] extracted nested entities from text by stacking BiLSTM-CRF. However, stacking different layers of sequence labeling models causes error propagation.

Instead of sequence labeling models, span-based methods have also been used to identify nested entities. MGNER [19] is a novel framework that divides nested named entity recognition into two parts: entity span detection and span classification. The span representation is obtained by first locating the entity span, and then the entity category classification is performed based on the entity span information. This method achieved promising performance in extracting entities, but the boundaries of the extracted entities are susceptible to deviations. Zheng et al. [20] accurately located the boundary using the boundary-aware method and combined the two subtasks of boundary detection and span classification by parameter sharing.

6. Conclusions and Future Work

In this paper, we presented a head-to-tail named entity recognition model to extract nested or normal entities from a given text. The proposed model is a sequence-based tagging approach that identifies entity boundaries and entity categories using two correlated steps. The entity boundary is divided into the entity head and entity tail, and it is easier to identify the entity tail for different entity categories by making each entity head the prior condition. In addition, dividing entity head and tail into two steps in cascade for identification facilitates the more accurate localization of entity boundaries. Specifically, the positioning of entity boundaries requires two cascading steps, and more stringent conditions improve the accuracy of the model in extracting entity boundaries. The experimental results demonstrated the effectiveness of the proposed method, which achieved the best performance in comparison with nine baselines.

In the future, we will explore more effective methods for fusing the extracted entity head information to improve the accuracy of extracting entity tails for different entity categories.

Author Contributions: Conceptualization, J.Y.; methodology, H.L.; software, X.L.; investigation, X.L.; resources, J.Y.; writing—original draft preparation, X.L.; writing—review and editing, P.H.; supervision, J.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Anhui Provincial Natural Science Foundation (No. 1908085MF202) and the Independent Scientific Research Program of National University of Defense Science and Technology (No. ZK18-03-14).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Guo, S.; Chang, M.W.; Kiciman, E. To link or not to link? a study on end-to-end tweet entity linking. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Atlanta, GA, USA, 9–15 June 2013; pp. 1020–1030.
2. Gupta, N.; Singh, S.; Roth, D. Entity linking via joint encoding of types, descriptions, and context. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 7–11 September 2017; pp. 2681–2690.
3. Martins, P.H.; Marinho, Z.; Martins, A.F. Joint Learning of Named Entity Recognition and Entity Linking. In Proceedings of the ACL 2019, Florence, Italy, 28 July–2 August 2019; pp. 190–196.
4. Miwa, M.; Bansal, M. End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Berlin, Germany, 7–12 August 2016; pp. 1105–1116.
5. Liu, L.; Ren, X.; Zhu, Q.; Zhi, S.; Gui, H.; Ji, H.; Han, J. Heterogeneous Supervision for Relation Extraction: A Representation Learning Approach. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 7–11 September 2017; pp. 46–56.
6. Wei, Z.; Su, J.; Wang, Y.; Tian, Y.; Chang, Y. A novel cascade binary tagging framework for relational triple extraction. In Proceedings of the ACL 2020, Online, 5–10 July 2020; pp. 1476–1488.
7. Wang, Y.; Yu, B.; Zhang, Y.; Liu, T.; Zhu, H.; Sun, L. TPLinker: Single-stage Joint Extraction of Entities and Relations Through Token Pair Linking. In Proceedings of the 28th International Conference on Computational Linguistics, Barcelona, Spain, 8–13 December 2020; pp. 1572–1582.
8. Li, Q.; Ji, H.; Huang, L. Joint event extraction via structured prediction with global features. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Sofia, Bulgaria, 4–9 August 2013; pp. 73–82.

9. Nguyen, T.H.; Cho, K.; Grishman, R. Joint event extraction via recurrent neural networks. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, CA, USA, 12–17 June 2016; pp. 300–309.
10. Molla, D.; van Zaanen, M.; Cassidy, S. Named Entity Recognition in Question Answering of Speech Data. In Proceedings of the Australasian Language Technology Workshop 2007, Melbourne, Australia, 10–11 December 2007; pp. 57–65.
11. Zhang, J.; Shen, D.; Zhou, G.; Su, J.; Tan, C.L. Enhancing HMM-based biomedical named entity recognition by studying special phenomena. *J. Biomed. Inform.* **2004**, *37*, 411–422. [[CrossRef](#)] [[PubMed](#)]
12. Zhou, G.; Zhang, J.; Su, J.; Shen, D.; Tan, C. Recognizing names in biomedical texts: A machine learning approach. *Bioinformatics* **2004**, *20*, 1178–1190. [[CrossRef](#)] [[PubMed](#)]
13. Zhou, G. Recognizing names in biomedical texts using mutual information independence model and SVM plus sigmoid. *Int. J. Med. Inform.* **2006**, *75*, 456–467. [[CrossRef](#)] [[PubMed](#)]
14. Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* **2011**, *12*, 2493–2537.
15. Huang, Z.; Xu, W.; Yu, K. Bidirectional LSTM-CRF models for sequence tagging. *arXiv* **2015**, arXiv:1508.01991.
16. Lu, W.; Roth, D. Joint mention extraction and classification with mention hypergraphs. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 857–867.
17. Muis, A.O.; Lu, W. Labeling Gaps Between Words: Recognizing Overlapping Mentions with Mention Separators. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 7–11 September 2017; pp. 2608–2618.
18. Wang, B.; Lu, W. Neural Segmental Hypergraphs for Overlapping Mention Recognition. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 204–214.
19. Xia, C.; Zhang, C.; Yang, T.; Li, Y.; Du, N.; Wu, X.; Fan, W.; Ma, F.; Philip, S.Y. Multi-grained Named Entity Recognition. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 1430–1440.
20. Zheng, C.; Cai, Y.; Xu, J.; Leung, H.F.; Xu, G. 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), Hong Kong, China, 3–7 November 2019; pp. 357–366.
21. Yu, B.; Zhang, Z.; Shu, X.; Wang, Y.; Liu, T.; Wang, B.; Li, S. Joint Extraction of Entities and Relations Based on a Novel Decomposition Strategy. In Proceedings of the ECAI 2020, Santiago de Compostela, Spain, 29 August–8 September 2020.
22. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
23. Doddington, G.R.; Mitchell, A.; Przybocki, M.A.; Ramshaw, L.A.; Strassel, S.M.; Weischedel, R.M. Automatic Content Extraction (ACE) program-task definitions and performance measures. In Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC-2004), Lisbon, Portugal, 26–28 May 2004; pp. 837–840.
24. Kim, J.D.; Ohta, T.; Tateisi, Y.; Tsujii, J. GENIA corpus—A semantically annotated corpus for bio-textmining. *Bioinformatics* **2003**, *19*, i180–i182. [[CrossRef](#)] [[PubMed](#)]
25. Ju, M.; Miwa, M.; Ananiadou, S. 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), New Orleans, LA, USA, 1–6 June 2018; pp. 1446–1459.
26. Wang, B.; Lu, W.; Wang, Y.; Jin, H. A Neural Transition-based Model for Nested Mention Recognition. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 1011–1017.
27. Katiyar, A.; Cardie, C. 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), New Orleans, LA, USA, 1–6 June 2018; pp. 861–871.
28. Lin, H.; Lu, Y.; Han, X.; Sun, L. Sequence-to-Nuggets: Nested Entity Mention Detection via Anchor-Region Networks. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 5182–5192.
29. Luo, Y.; Zhao, H. Bipartite Flat-Graph Network for Nested Named Entity Recognition. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 6408–6418.
30. Zhou, G.; Su, J. Named entity recognition using an HMM-based chunk tagger. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, PA, USA, 7–12 July 2002; pp. 473–480.
31. Ratinov, L.; Roth, D. Design challenges and misconceptions in named entity recognition. In Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009), Boulder, CO, USA, 4–5 June 2009; pp. 147–155.
32. Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; Dyer, C. 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, San Diego, CA, USA, 12–17 June 2016; pp. 260–270.
33. Chiu, J.P.; Nichols, E. Named entity recognition with bidirectional LSTM-CNNs. *Trans. Assoc. Comput. Linguist.* **2016**, *4*, 357–370. [[CrossRef](#)]

-
34. Ma, X.; Hovy, E. 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), Berlin, Germany, 7–12 August 2016; pp. 1064–1074.
 35. Shen, D.; Zhang, J.; Zhou, G.; Su, J.; Tan, C.L. Effective adaptation of hidden markov model-based named entity recognizer for biomedical domain. In Proceedings of the ACL 2003 Workshop on Natural Language Processing in Biomedicine, Sapporo, Japan, 11 July 2003; pp. 49–56.
 36. Gu, B. Recognizing nested named entities in GENIA corpus. In Proceedings of the HLT-NAACL BioNLP Workshop on Linking Natural Language and Biology, New York, NY, USA, 8 June 2006; pp. 112–113.
 37. Alex, B.; Haddow, B.; Grover, C. Recognising nested named entities in biomedical text. In *Biological, Translational, and Clinical Language Processing*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2007; pp. 65–72.