Unveiling Spatial Relations in Nested Named Entity Recognition:A CNN-Driven Approach

BenYuan Chen,30920231154340,AI
Zheng Fang,30920231154343,AI
Jie Hu,36920231153193,AI
XinJie Peng,309J20231154356,AI

Abstract

Named entity recognition (NER) is the task to detect and classify the entity spans in the text. But as the complexity of downstream tasks increases, Research on nested named entity recognition is becoming increasingly popular. Nested Named Entity Recognition is an extended form of NER. In nested NER, unlike traditional NER, the boundaries between entities can contain each other, forming a nested or hierarchical structure. Span-based methods have been widely used to tackle the nested NER. Most of these methods will get a score matrix, where n means the length of sentence, and each entry corresponds to a span. However, in previous work, researchers mainly focused on generating this fractional matrix, while ignoring the spatial relationships in the fractional matrix. The spatial relationship includes the positional relationship between spans, that is, whether one span overlaps within, outside, or partially within another span.In this task, we propose using Convolutional Neural Network (CNN) to model these spatial relations in the fractional matrix. We hope to validate the performance of the model in handling nested named entity recognition tasks on multiple datasets by combining the fractional matrix obtained based on span method and spatial relationships simulated using CNN.

Introduction

Named Entity Recognition (NER) is a natural language processing (NLP) task that focuses on identifying and categorizing named entities within a given text.Named entities are specific words or phrases that represent real-world objects, such as people's names, locations, organizations, dates, numerical values, and more.

Nested Named Entity Recognition (Nested NER) is a more advanced and complex variant of traditional Named Entity Recognition (NER) in natural language processing. 'Nested Entities' are named entities containing references to other named entities as in [University of[Tokyo]], in which both [Tokyo] and [University of Tokyo] are named entities.Such nested entities are frequent in data sets like ACE 2004, ACE 2005 and GENIA(e.g., 17% of NEs in GENIA are nested (Finkel and Manning 2009))

Traditionally, solving this task involved primarily using the sequence labeling paradigm, where a label is assigned to each token(Yan et al. 2019).This method isn't suitable for nested NER, where tokens can belong to multiple entities.To address this concern, the span-based approach, which assigns labels to individual spans, was introduced(Eberts and Ulges 2019; Li et al. 2019).

Eberts and Ulges 2019.used a pooling method over token representations to get the span representation, and then conducted classification on this span representation.Li et al. 2019.transformed the NER task into a Machine Reading Comprehension form, they used the entity type as the query, and asked the model to select the spans that belong to this entity type.

The spans adjacent to a given span exhibit special relationships with the central span. Exploiting these spatial correlations could prove advantageous.In this paper, we use the Biaffine decoder(Dozat and Manning 2016) to obtain a 3D feature matrix where each entry represents a span. We then treat this feature matrix as an image and use convolutional neural network (CNN) to model the local interactions between spans.

We evaluated our method on three nested NER benchmarks (ACE 2004, ACE 2005, GENIA) and compared this simple method with recently proposed methods. Despite the simplicity of our method, it yielded relatively large performance gains in three widely used nested NER datasets.We believe that this way of viewing the spann feature matrix as an image will provide some insight into future exploration of span-based methods for nested NER tasks.

related work

Currently, there are generally four frameworks used for nested named entity recognition: sequence labeling framework, using hypergraph for efficient span representation, sequence-to-sequence framework, and span classification methods. However, each of these four methods has its own shortcomings:

Sequence Labeling Framework(Wang and Lu 2018): In nested named entity recognition, a token can contain multiple entities, which necessitates the use of Cartesian products in entity labels. However, Cartesian labels are inevitably affected by the long tail problem.

Using hypergraph for Efficient Span Representation(Wang and Lu 2018): Using hypergraph is an innovative approach in natural language processing that aids in

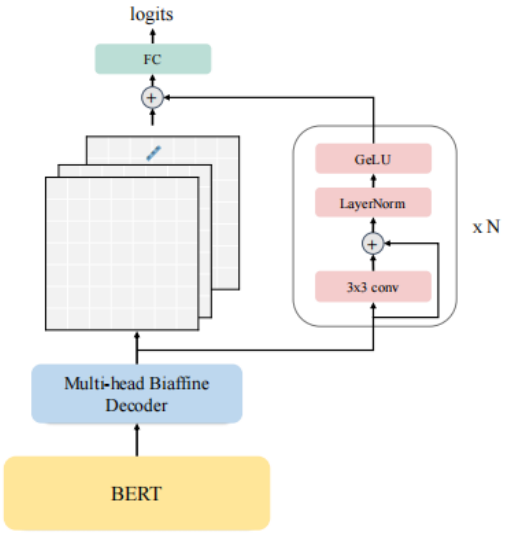

Figure 1: The proposed method in this paper. Use several blocks of CNN to model the spatial correlations between neighbor spans.

handling complex entity nesting structures and span relationships more flexibly and efficiently. In nested named entity recognition, hypergraph provide a more powerful representation to better capture and manage the complex relationships between spans. However, this method's decoding process is overly complex.

Sequence-to-Sequence Framework: Seq2Seq models are widely applied to various natural language processing and sequence generation tasks. A Seq2Seq model consists of two main components: an encoder and a decoder. It is used for generating entity sequences. Entity sequences can be either entity pointer sequences or entity text sequences. However, Seq2Seq methods require substantial decoding time.

Span Classification(Yu, Bohnet, and Poesio 2020): Span classification is a natural language processing task that involves identifying and classifying specific spans or segments in text. This method enumerates all possible spans in a sentence and uses pooling techniques to obtain span representations. Span methods are easily parallelizable and have a straightforward decoding process, making them widely adopted. However, prior work has overlooked the relationships between adjacent spans.

Proposed Method

In this section, we first introduce the nested NER task, then describe how to get the feature matrix.After that, we present the CNN module to model the spatial correlation on the feature matrix. At general framework of our proposed method can be viewed in Figure 1.

Nested NER Task

In many practical applications, it is common that the named entities have a nested structure(Strassel and Mitchell 2003; Finkel and Manning 2009). Specifically, an entity could contain other entities or be a part of other entities, which breaks the second assumption mentioned above. As the example shown in Figure 2, the outer entity "the Oakland Zoo" contains an inner entity, i.e., "Oakland". In the AnCora corpus of Spanish and Catalan newspaper text, nearly half of entities are embedded within another entity. Named entities with the nested structure are also prevalent in specific domains. For example, approximately 17% of entities in the GENIA corpus, a biomedical domain corpus labeled with entity categories such as protein and DNA, are embedded. Consequently, the NER task is required for further recognizing named entities with nested structures (i.e., both outer entities and inner entities), rather than the longest outer entity only. The inherent complexity of nested entities makes the nested NER a more challenging task than traditional flat NER. Given an input sentence $X = [x_1, x_2, ..., x_n]$ with n tokens, the nested NER task aims to extract all entities in X. Each entity can be expressed as a tuple (s_i, e_i, t_i). s_i, e_i are the start and end indices of the entity. $t_i \in \{1, ..., |T|\}$ is its entity type, and $|T|$ is the number of entity types. As the task name suggests, the entities may overlap with each other, but different entities are not allowed to have crossing boundaries. For a sentence with n tokens, there are $n(n+1)/2$ valid spans.

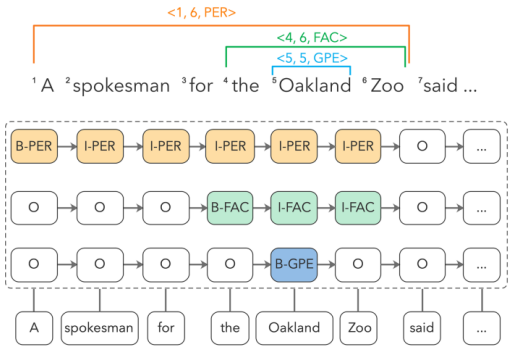

Figure 2: Nested NER Task

Span-based Method for Nested NER

We have transformed this task into a span classification task, where the model assigns an entity label to each valid span. The method begins by using an encoder to encode the input sentence, expressed as:

$$H = Encoder(X)$$

Here, $H \in R^{n \times d}$, and d is the size of the hidden layer. Various pre-trained models, such as BERT, are commonly utilized as the encoder. For words tokenized into multiple pieces, we employ max-pooling to aggregate their hidden states.

After obtaining the contextualized embeddings of tokens, previous approaches typically concatenate them with static word embeddings and character embeddings. The combined embedding is then input into a BiLSTM layer. To simplify

the model, we avoid using additional embeddings or the BiLSTM layer.

Next, we use a multi-head biaffine decoder to obtain the score matrix as follows:

$$H_s = LeakyReLU(HW_s),$$
$$H_e = LeakyReLU(HW_e),$$
$$R = MHBiaffine(H_s, H_e)$$

Here, $W_s, W_e \in R^{d \times h}$, where h is the size of the hidden layer, and $MHBiaffine(\cdot,\cdot)$ represents the multi-head biaffine decoder. The resulting matrix $R \in R^{n \times n \times r}$, where r is the size of the features.

The input of the Multi-head Biaffine decoder consists of two matrices $H_s, H_e \in R^{n \times h}$, and the output is $R \in R^{n \times n \times r}$. The formulation of the Multi-head Biaffine decoder is as follows:

$$S_1[i,j] = (H_s[i] \oplus H_e[j] \oplus w_{i-j})W$$

Here, $\oplus$ denotes vector concatenation, $w_{i-j} \in R^c$ is the span length embedding for length $i-j$, and $W \in R^{(2h+c) \times r}$. The detailed process of the decoder is as follows:

1. Firstly, H_s and H_e are split into a series of sub-matrices H_{s_k}, H_{e_k} using a Split operation, where k is the index of the head. The Split operation equally divides the matrix along its last dimension, resulting in sub-matrices of dimensions $n \times h_k$.

2. Compute the first part S_1, which is obtained by element-wise multiplication of the sub-matrices at corresponding positions for each head k. Specifically, for each head k, calculate $S_{(1)2k}[i,j] = H_{(s_k)}[i] \cdot U \cdot H_{(e_k)}[j]^T$, where $U \in R^{h_k \times r \times h_k}$ is a learnable parameter. Concatenate the results from all heads to obtain S_2, i.e., $S_2 = Concat(S_{(1)2}, ..., S_{(1)2K})$.

3. The final output R is the sum of S_1 and S_2, i.e., $R = S_1 + S_2$. Each element $R[i,j]$ in the matrix R can be regarded as the feature vector for the span, representing the span from the j-th to the i-th position.

Each cell (i,j) in the matrix R can be viewed as the feature vector $v \in R^r$ for the span. For the lower triangle of R (where $i > j$), the span encompasses words from the j-th to the i-th position.

CNN on Score Matrix

As illustrated in Figure 3, cells exhibit relationships with those in their vicinity. Consequently, we propose the utilization of Convolutional Neural Networks (CNN) to model these interactions. The following CNN block is iteratively applied in our model:

$$R_0 = Conv2d(R),$$
$$R_{00} = GeLU(LayerNorm(R_0 + R)),$$

Here, $Conv2d$, $LayerNorm$, and $GeLU$ represent 2D convolution, layer normalization (Ba, Kiros, and Hinton 2016), and the GeLU activation function(Hendrycks and Gimpel 2023), respectively. Layer normalization is performed in the feature dimension. It is noteworthy that due to varying token counts n in sentences, the shapes of their

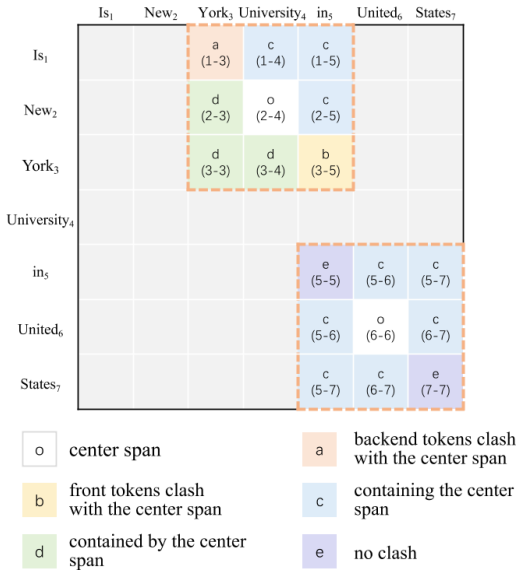

Figure 3: All valid spans of a sentence. We use the start and end tokens to pinpoint a span, for instance, "(2-4)" represents "New York University". Spans in the two orange dotted squares indicates that the center span can have the special relationship (different relations are depicted in different colors) with its surrounding spans. For example, the span "New York" (2-3) is contained by the span "New York University" (2-4). Therefore, the "(2-3)" span is annotated as "d".

corresponding R matrices differ. To ensure consistent results when processing R in batches, the 2D CNN excludes a bias term, and all paddings in R are zero-filled.

After undergoing several CNN blocks, the resulting R_{00} is further processed by another 2D CNN module.

In summary, this passage describes the application of CNN to the score matrix. Initially, the score matrix R undergoes convolutional operations (R_0), followed by further adjustments through layer normalization and the GeLU activation function, resulting in R_{00}. This process is repeated multiple times, and the output is further processed by another 2D CNN module. Such a design aims to capture local interactions in the score matrix, enhancing the model's ability to model relationships.

The Output

We utilize a perceptron to obtain prediction logits as follows:

$$P = Sigmoid(W_o(R + R_{00}) + b)$$

Here, $W_o \in R^{|T| \times r}$, $b \in R^{|T|}$, and $P \in R^{n \times n \times |T|}$.

Subsequently, we employ binary cross-entropy to calculate the loss:

$$L_{BCE} = -\sum_{0 \le i,j < n} y_{ij} \log(P_{ij})$$

In contrast to prior approaches that typically only use the upper triangular part to compute the loss (Yu, Bohnet, and Poesio 2020; Zhu and Li 2022), we incorporate both upper

	ACE2004			ACE2005		
	Precision	Recall	F1 score	Precision	Recall	F1 score
W2NER(BERT-large)	87.17	87.70	87.43	85.78	87.81	86.77
SG(RoBERTa-base)	86.70	85.93	86.31	84.37	85.87	85.11
Ours(BERT-large)	87.98	87.50	**87.74**	86.26	87.56	**86.91**
Ours without CNN(BERT-large)	86.60	86.48	86.54	84.91	87.39	86.13
Ours(RoBERTa-base)	87.33	87.29	**87.31**	86.70	88.16	**87.42**
Ours without CNN(RoBERTa-base)	86.09	86.88	86.48	85.17	88.0	86.56

Table 1: Results for the ACE2004, ACE2005, and Genia datasets. BERT-large or RoBERTa-base in parentheses means the pre-trained model used as the sentence encoder.

and lower triangles in the loss calculation. The reason is that, for batch computation, we cannot solely compute the upper triangle part. Since the lower triangle part has already been computed, we utilize it for the output as well. The tags in the score matrix are symmetric, meaning the tag in the (i, j)-th entry is the same as in the (j, i)-th entry.

During inference, we compute scores in the upper triangular part as:

$$\hat{P}_{ij} = \frac{P_{ij} + P_{ji}}{2}$$

where $i \leq j$. Subsequently, we exclusively use this upper triangular score to obtain the final prediction. The decoding process generally follows(Yuan et al.).'s method (2020). We initially prune out non-entity spans (those with scores below 0.5), then sort the remaining spans based on their maximum entity score. We select spans in accordance with this order, and if a span's boundary conflicts with selected spans, it is ignored.

Experiment

Data Sets

We conduct experiments in the following three widely used nested NER datasets: ACE2004(Doddington et al. 2004), ACE2005(Walker and Consortium 2005), Genia(Kim et al. 2003), to comprehensively testify the effectiveness of our method.

Besides, we choose some other methods for nested NER as our baselines to be compared, these methods are W2NER(Li et al. 2022) and SG(Wan et al. 2022). In order to strictly control variables and ensure rigor, for every method, we preprocess and split all the data sets in the same manner as suggested(Strassel and Mitchell 2003). For ACE2004, We divide the data set into three parts: training set, verification set, and test set. The number of sentences contained in each subset is 6297, 742, and 824 respectively. The average sentence length is 23.52. For ACE2005, we divide the data set in the same manner, and the number of sentences contained in each subset is 7178, 960, and 1051 respectively, and the average sentence length is 20.59. For the Genia data set, these numbers are 15038, 1765, and 1732, and the average sentence length is 26.47. We replicate each experiment five times and report its average performance. The preprocessing and tokenization of data for every experiment remain the same.

Results

Firstly we conducted an ablation experiment. As shown in Table 2, we tested two entity recognition tasks in the 3 data sets mentioned above, flat entity recognition and nested entity recognition. We can see that for both two tasks, the model with CNN always outperforms the model without CNN in all test data sets, which shows that the application of CNN can improve the effect of NER, especially on the nested NER task where the model using CNN is significantly better than the model without CNN. Next, we tested the performance of different models in ACE2004 and ACE2005. For our model proposed, we also tested the performance of the models with different sentence encoders used, and with or without CNN. The result shows that our method, with CNN, outperforms the other two methods on both data sets, and both BERT-large and RoBERTa-base as the sentence encoder achieve good scores. That proves that using CNN to model the interaction between neighbor spans can be beneficial to the nested NER task.

Our Intuitive Analysis and Explanation

Why CNN helps? Here we will not give rigorous mathematical proofs, but make qualitative analysis and explanations based on intuitive understanding. The usage of CNN based on a conventional nested NER task shows unexpectedly good results in a very, embarrassingly simple way. It outperforms some other recently proposed baseline methods which are more complex than ours. In order to understand this phenomenon, we conducted another ablation experiment, which showed that the improvement effect brought by CNN is very significant, the recall score of nested named entity recognition can be improved by at least two points. Conventional methods cannot do nested NER task well, largely because they do not explore the contextual semantic information between spans. As we mentioned before, we generate a score matrix from a multi-head Biaffine decoder. Focusing on a center element of the score matrix, we notice that the eight elements that are spatially close to the cen-

	FEP	FER	NEP	NER
ACE2004				
with CNN	86.9	87.3	88.4	**88.8**
without CNN	86.3	86.8	89.4	86.6
ACE2005				
with CNN	86.2	88.3	91.4	**89.0**
without CNN	85.2	87.9	91.3	86.2
Genia				
with CNN	81.7	79.4	71.7	**75.5**
without CNN	79.0	80.0	72.7	64.8

Table 2: Ablation experiment results on whether to CNN on score matrix. This table records the precision and recall for flat and nested entities in the test set of three datasets. FEP, FER, NEP and NER denote the flat entity precision, flat entity recall, nested entity precision and nested entity recall, respectively. Compared with models without CNN, the most improved metric is bold.

ter element also have a close relationship semantically. In other words, the spatiality of the score matrix is highly related to the hierarchical relationship of span semantics. That is to say, we can exploit the semantic information of spans through the spatial information of the score matrix. CNN is a good explorer of spatial information and can discover spatial semantic information of spans, thereby improving the recognition effect of named entities.

Conclusion

In our project, we use a method for span-based nested NER tasks, which uses CNN on the score matrix to improve the model's performance. It is very simple to apply CNN in a vanilla span-based NER model, however, experiments on different data sets and comparisons with some other models show that our method achieves comparable or better performance than some recently proposed methods. We analyzed the results and came to the following inferences: A nested relationship between spans can be regarded as a sort of spatial relationship. CNN is good at exploiting spatial relationships, hence, CNN can exploit the spatial correlation between neighbor spans, thereby it can help to find more nested named entities.

References

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer Normalization. arXiv:1607.06450.

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.

Dozat, T.; and Manning, C. D. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Eberts, M.; and Ulges, A. 2019. Span-based joint entity and relation extraction with transformer pre-training. *arXiv preprint arXiv:1909.07755*.

Finkel, J. R.; and Manning, C. D. 2009. Nested named entity recognition. In *Proceedings of the 2009 conference on empirical methods in natural language processing*, 141–150.

Hendrycks, D.; and Gimpel, K. 2023. Gaussian Error Linear Units (GELUs). arXiv:1606.08415.

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

Li, J.; Fei, H.; Liu, J.; Wu, S.; Zhang, M.; Teng, C.; Ji, D.; and Li, F. 2022. Unified named entity recognition as word-word relation classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 10965–10973.

Li, X.; Feng, J.; Meng, Y.; Han, Q.; Wu, F.; and Li, J. 2019. A unified MRC framework for named entity recognition. *arXiv preprint arXiv:1910.11476*.

Strassel, S.; and Mitchell, A. 2003. Multilingual resources for entity extraction. In *Proceedings of the ACL 2003 workshop on Multilingual and mixed-language named entity recognition*, 49–56.

Walker, C.; and Consortium, L. D. 2005. *ACE 2005 Multilingual Training Corpus*. LDC corpora. Linguistic Data Consortium. ISBN 9781585633760.

Wan, J.; Ru, D.; Zhang, W.; and Yu, Y. 2022. Nested named entity recognition with span-level graphs. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 892–903.

Wang, B.; and Lu, W. 2018. Neural Segmental Hypergraphs for Overlapping Mention Recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Yan, H.; Deng, B.; Li, X.; and Qiu, X. 2019. TENER: adapting transformer encoder for named entity recognition. *arXiv preprint arXiv:1911.04474*.

Yu, J.; Bohnet, B.; and Poesio, M. 2020. Named Entity Recognition as Dependency Parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Yuan, Z.; Tan, C.; Huang, S.; and Huang, F. ???? Fusing Heterogeneous Factors with Triaffine Mechanism for Nested Named Entity Recognition.

Zhu, E.; and Li, J. 2022. Boundary Smoothing for Named Entity Recognition.