Word Pair Information is Important for Nested Named Entity Recognition

No Author Given

No Institute Given

Abstract. Named Entity Recognition (NER) is the task of detecting and classifying entity spans in texts. When entity spans overlap with each other, the task is called nested NER. The span-based approach is highly efficient for handling nested NER. However, in previous studies, span representations were generated by integrating the endpoint word representations of the span or by integrating all the word representations within the span, without fully considering the local dependencies among different words within the span. Furthermore, when using feature matrices for entity prediction, the dependencies between entities and the contextual information of entities in a sentence were not considered. To address these issues, we propose a **S**pan-**En**hanced **Net**work (**SEnNet**), which utilizes word pair information within the span to construct initial span representations. These span representations are then gradually enriched through interactions between different spans and the introduction of contextual information, effectively capturing both internal and external span information. Experimental results show that the proposed model achieves the best performance compared to 10 advanced baseline models on three nested datasets and one non-nested dataset. Further experimental analysis shows that fully leveraging word pair information within spans helps enhance the recognition performance of entities of different lengths, especially long entities. The source code for our model is available at: https://anonymous.4open.science/r/SEnNet-228C

Keywords: nested named entity recognition· span representation· word pair information· conditional layer normalization· long entity identification· feature matrix.

1 Introduction

Named Entity Recognition (NER) is a fundamental information extraction task in Natural Language Processing (NLP) that identifies spans of entities and their semantic categories in text. Previously, this task was primarily addressed using a sequence labeling approach, whereby each token is assigned a label [3,6]. It cannot be directly applied to nested NER, where one entity is nested within another, since each token might belong to two or more entities.

Various approaches have been proposed to tackle nested NER, including optimization based on sequence labeling [19,20], hypergraphs [18,24], sequence-to-sequence [22,27], and span-based [7,25,26]. The span-based approach have be-

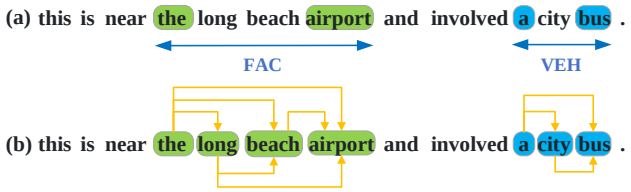


Fig. 1: A sentence from ACE05 dataset. (a) denotes the information utilized to represent the span by integrating its start and end information. (b) denotes the local dependency information between word pairs contained within the span, e.g. the links (*a, bus*) and (*city, bus*) in the span *a city bus* indicate that the meaning of *bus* in the span depends on the words *a* and *city*.

come one of the most mainstream approaches for handling nested NER, treating nested NER as a text span classification problem.

Despite the significant advancements achieved by the span-based approach in nested NER, two critical issues still require attention. Firstly, in previous studies [7,14,21,23,25], span representations are generated by integrating contextual information of span start and end words or by integrating information from all the words within the span, which ignores the local dependency information between word pairs within the span, as shown in Fig. 1. The dependency information of the same word in different spans varies, which helps improve the distinguishability of overlapping spans. Secondly, entities within the same sentence are interdependent, and the entity category of the same entity may vary in different contexts. However, in previous studies, when predicting entities using the feature matrix, the dependencies between entities and the contextual information of the entities were overlooked. Although some studies Yan et al. [23] used convolutional neural networks (CNNs) to mitigate this problem, CNNs can only model span dependencies within a single window at a time and cannot simultaneously capture span dependencies at different positions in the feature matrix, along with the introduction of contextual information.

Constructing effective span representations is crucial for span-based NER. In this paper, we utilize the word pair information within spans to construct the initial span representation. Then, we integrate the dependency information between spans and the contextual information of spans within sentences to form the final span representation. The initial span representation, incorporating both word pair and word information within the span, enhances the retention of original semantic details, crucial for NER tasks, as illustrated in Fig. 2.

Based on this observation, we propose a novel network model called **S**pan-**En**hanced **Net**work (**SEnNet**), which divides the nested NER process into three stages, as illustrated in Fig. 3: (1) span representation stage: constructing word pair information using Conditional Layer Normalization [7], on the basis of which the initial span representations is constructed. (2) enriching span representation stage: Introducing dependencies between spans and contextual information of spans in a sentence using the decoder structure in Transformers [16] (3) entity

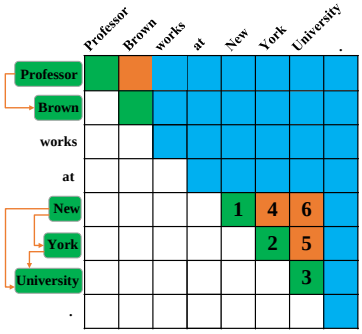


Fig. 2: Integrate the word pair information within the span as the initial representation of the span. Cell 4 represents the dependency information for the word pair (*New, York*), where *York* is conditioned on *New*. This cell illustrates the representation of the word *York* given the word *New*. Cell 2 represents the dependency information for the word pair (*York, York*), treating it as the word information of *York* within the span. We simultaneously integrate information from cells 1 to 6 to generate the initial representation of the span "*New York University*".

prediction stage: In the final prediction, we decode all entities in the sentence at once, and the decoded result is considered an unordered set. Our contributions follow:

- We suggest that local dependency information between word pairs within spans should be considered in span representations for span-based NER methods. To capture this information, we propose a word pair information fusion method to incorporate word pair information within spans.
- This paper proposes a novel span-based nested NER model that simultaneously integrates word pair information within spans, dependency information between spans, and contextual information of spans in a sentence to enhance span representation.
- Experimental results show that the proposed model can effectively utilize both internal and external span information, improving entity recognition performance and outperforming relevant baseline models. Detailed ablation experiments validate the effectiveness of these innovations and the impact of word pair information on recognizing long entities.

2 Related Work

In recent years, various paradigms have been proposed to tackle nested NER. Traditional sequence labeling approaches, such as [3,6], assign a predefined label to each token, but this is difficult to handle nested structures. Some works propose various improvement strategies. Hypergraph approaches transform sentences into hypergraphs, allowing multiple labels per token. For example [18,24].

Generation-based approaches such as [22,27] define named entity recognition as a sequence generation task can handle both nested and non-nested entities.

Span-based approaches have become one of the most mainstream approaches for handling NER tasks. These approaches define NER as a span-level classification task. Shen et al. [10] adopts the concept of object detection, performing boundary regression on the basis of enumerated spans. Wu et al. [21] generates span representations by constructing a pyramid structure, and models the relationships between spans on this basis. Yu et al. [25] utilizes the boundary representations of span start and end points to directly compute span category scores using a biaffine classifier. Yuan et al. [26] enhances span representations by integrating internal token representations, boundary representations, label representations, and associated span representations using a triaffine structure. Zhu and Li [28] borrows the concept of label smoothing to perform boundary smoothing on the basis of spans. Yan et al. [23] employs convolutional neural networks on feature matrix to model the relationships between central spans and surrounding spans.

The span-based method can effectively address nested NER without the need for intricate detection patterns, while demonstrating outstanding performance. However, the span representations are generated by integrating the endpoint word representations of the span or by integrating all the word representations within the span, without fully considering the dependencies among different words within the span. Furthermore, when using feature matrices for entity prediction, the dependencies between entities and the contextual information of the entities were not considered. Our SEnNet model addresses the aforementioned two issues by modeling the relationships between word pairs within spans and the dependencies between spans, while also incorporating the context of the spans within the sentence.

3 Model

In this section, we first explain how to construct the initial span representations and then discuss how to model the dependencies between spans and the contextual information of spans in a sentence. Next, we describe the model training and entity prediction process. The overview architecture of the model is illustrated in Fig 3.

3.1 Span Representation

Word Representation. Given an input sentence $X = \{x_1, x_2, \ldots, x_N\}$ of length N, we concatenate the contextual embedding x_i^{bert} generated by BERT [1], the GloVe [9] word embedding x_i^{glove}, the part-of-speech embedding x_i^{pos}, and the character embedding x_i^{char} for each word. The concatenated embeddings are then fed into a BiLSTM[8] to obtain the word-level representation X_i.

$$X_i = \text{BiLSTM}([x_i^{\text{bert}} \oplus x_i^{\text{glove}} \oplus x_i^{\text{pos}} \oplus x_i^{\text{char}}]) \tag{1}$$

where $\oplus$ denotes concatenation operation.

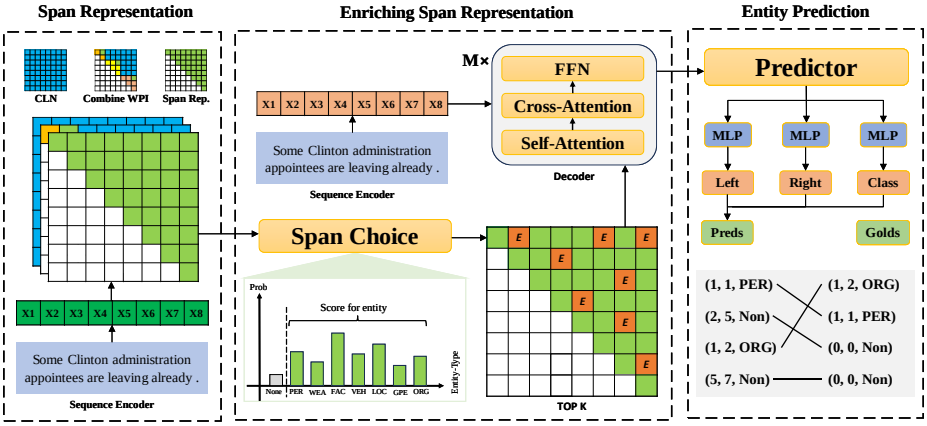


Fig. 3: The overview architecture of SEnNet. *CLN* represents the word pair information matrix formed by conditional layer normalization. *Span Rep.* represents the initial span representation matrix formed on the word pair information matrix. *Combine WPI* representation of a process of fused **W**ord **P**air **I**nformation (*WPI*) to form initial span representations. In the TOP K matrix, E represents the selected candidate entities.

Initial Span Representation. Initial span representation is obtained by pooling all word pair information within the span. It is important to utilize the word representations sequence $\mathbf{X}_{\text{sen}} = \{X_1, X_2, \ldots, X_N\}$, generate a high-quality word pair information matrix $V \in \mathbb{R}^{N \times N \times d}$. Following Li et al.[7], we adopt the Conditional Layer Normalization (CLN) to build matrix V, where V_{kl} represents the word pair (x_k, x_l), with word x_l conditioned on word x_k.

$$V_{kl} = CLN(X_k, X_l) \quad = \gamma_k \odot \left(\frac{X_l - \mu}{\delta} \right) + \lambda_k \tag{2}$$

where X_k and X_l correspond to the word representations of x_k and x_l, respectively. X_k is the condition for generating the gain parameter γ_k and λ_k of layer normalization, which are obtained by two different feed-forward networks: $\gamma_k = \text{FFN}(X_k)$ and $\lambda_k = \text{FFN}(X_k)$. μ and δ are the mean and standard deviation across the elements of X_l.

Based on the word pair information matrix V, the span feature matrix $S \in \mathbb{R}^{N \times N \times d}$ is computed, where $S_{ij} \in \mathbb{R}^{1 \times d}$ denotes the initial representation of the span (x_i, x_j), which is the span that starts from x_i and extends to x_j.

$$S_{ij} = MaxPooling(V_{kl}) \tag{3}$$

where V_{kl} for all k, l in $i \leq k \leq l \leq j$

3.2 Enriching Span Representation

To enrich span representation, we utilize M Transformer [16] decoder blocks to capture the dependency information between spans as well as the contextual information of spans within the sentence.

Span Selection. For a sentence of length N, the size of the generated feature matrix is $N \times N \times d$, and there are $N(N+1)/2$ valid spans, many of which include non-entity spans. To save computational resources, the top K initial span representations are selected as the candidate entity set $Q \in \mathbb{R}^{K \times d}$ for the subsequent process, where the number of K significantly exceeds the number of entities in the sentence.

First, the span-based predictor utilizes initial span representations to predict the probability distribution for the different categories of each span.

$$P_{ij}^{\text{cls}} = \text{Softmax}(\text{Linear}(S_{ij})) \tag{4}$$

Then, by summing the probability of all entity categories except the non-entity category, the result is used as the probability for the span being an entity.

$$P_{ij} = \sum_{t \neq \emptyset} P_{ij}^{\text{cls}}(t) \tag{5}$$

where $P_{ij}^{\text{cls}}(t)$ indicates the probability of the span S_{ij} belonging to category t. $\emptyset$ is a non-entity type which means this span is not an entity.

The initial span representations S_{ij} are sorted in descending order according to P_{ij}, and then the top K initial span representations, most likely to be entities, are selected as the candidate entity set $Q \in \mathbb{R}^{K \times d}$.

Span Interaction. Entities within the same sentence have interdependent relationships. To capture these dependencies, we utilize a self-attention mechanism for modeling, with the specific computations as follows.

$$U_L^{SA} = \text{MultiHeadAttn}(U_{L-1}, U_{L-1}, U_{L-1}) \tag{6}$$

where, U_L^{SA} denotes the output of the self-attention in the L-th decoding layer, while U_{L-1} represents the output of the feed-forward network from the preceding layer. The Q from the span selection phase is used as the input for the self-attention layer in the first decoding layer $U_0 = Q$.

Contextual Information. Overlapping spans contain common word pair information, which may reduce the distinguishability between overlapping spans. To improve the distinguishability between overlapping spans, we use a cross-attention mechanism to introduce contextual information of the spans within the sentence, thereby enhancing span representation.

$$U_L^{CA} = \text{MultiHeadAttn}(U_L^{SA}, X_{\text{sen}}, X_{\text{sen}}) \tag{7}$$

where U_L^{CA} denotes the output of the cross-attention mechanism in the L-th decoding layer, while X_{sen} represents the word representation sequence of the input sentence.

The candidate entity set Q are processed through the self-attention layer and cross-attention layer, and then enter the feed-forward layer, the output of which is used as the final output of the current decoding layer.

$$U_L = \text{FFN}(U_L^{CA}) \tag{8}$$

where U_L denotes the output of the feed-forward layer in the L-th decoding layer, and serves as the input to the $(L+1)$-th decoding layer.

3.3 Prediction

Now, we utilize the output $U_M \in \mathbb{R}^{K \times d}$ of the feed forward layer of the last transform decoder block, i.e., the set of candidate entities Q after M transform decoder blocks, to decode the candidate entity boundaries and categories. For candidate entities $E_i \in U_M$, we calculate its entity classification probability distribution as follows:

$$P_i^{\text{cls}} = \text{MLP}_{cls}(E_i) \tag{9}$$

where MLP consisting of one linear layers and a softmax activation layer.

For the identification of left and right boundaries, we first integrate E_i with X_{sen}, by concatenating E_i with the representation of each word in X_{sen} to generate the integrated features H_i^{fuse}.

$$H_i^{\text{fuse}} = [h_1, h_2, h_3, \ldots, h_N] \tag{10}$$

$$h_j = E_i \oplus X_j, j \in [1, N] \tag{11}$$

where $\oplus$ denotes the concatenation operation.

Then, we use two separate Multi-Layer Perceptrons (MLP) to classify the integrated features, thereby obtaining the probability of each word being the left or right boundary of span E_i.

$$P_i^{\alpha} = \text{Softmax}(\text{MLP}_{\alpha}(H_i^{\text{fuse}})) \tag{12}$$

where $\alpha \in \{\text{left}, \text{right}\}$. P_i^{left} represents the probability distribution of each word in the sentence being the left boundary of E_i, and similarly for P_i^{right}. MLP consisting of two linear layers and a tanh activation layer.

Finally, the entity triples η_i decoded from the E_i are defined as follows:

$$\eta_i = (\eta_i^{\text{cls}}, \eta_i^{\text{left}}, \eta_i^{\text{right}}) \tag{13}$$

$$\eta_i^{\theta} = \text{argmax}(P_i^{\theta}) \tag{14}$$

where $\theta \in \{cls, left, right\}$, *cls* is the entity type, *left* and *right* are the left and right boundaries.

3.4 Training Objective

The network loss computation comprises two components: span selection loss and prediction loss.

Span Selection Loss. We first calculate the span selection loss, which essentially evaluates the probability distribution P_{ij}^{cls} for span selection. Following Zhu and Li [28], we adopt boundary smoothing for the computation of loss:

$$L_{\text{BS}} = -\sum_{0 \le i \le j \le N} y_{ij}^{T} \log(P_{ij}^{\text{cls}}) \tag{15}$$

where y_{ij}^{T} represents the probability distribution of the true entity categories.

Prediction Loss. Due to the unordered nature of the predicted entities, it is difficult to pre-assign real entity labels to them, making the computation of the final prediction loss challenging. Following Tan et al. [14], we compute the loss between the final entity predictions and the gold entities. We denote the gold entities as $y = \{(l_i, r_i, \text{ty}_i)\}_{i=1}^{L}$, where L indicates the number of entities, and l_i, r_i, and ty_i respectively represent left boundary, right boundary, and type indices of the i-th entity. We pad y with $\emptyset$ to ensure that it matches the number K of selected candidate entity set $Q \in \mathbb{R}^{k \times d}$. Thus, the arrangement of the predicted entity set that best matches the real entities is as follows:

$$\hat{\pi} = \text{argmin}_{\pi \in \Gamma(k)} \sum_{i=1}^{K} L_{\text{match}}(y_i, \hat{y}_{\pi(i)}) \tag{16}$$

where $\Gamma(k)$ is the set of all K-length permutations, and $L_{\text{match}}(y_i, \hat{y}_{\pi(i)})$ is the pairwise match cost between the i-th entity and the prediction indexed by $\pi(i)$. We compute this optimal assignment efficiently using the Hungarian algorithm Kuhn et al. [5]. We define $L_{\text{match}}(y_i, \hat{y}_{\pi(i)})$ as:

$$L_{match}(y_i, \hat{y}_{\pi(i)}) = -1_{\text{ty}_i \neq \phi}[P_{\pi(i)}^{cls}(\text{ty}_i) + P_{\pi(i)}^{l}(l_i) + P_{\pi(i)}^{r}(r_i)] \tag{17}$$

where 1_{ω} denotes the indicator function that takes 1 when ω is true and 0 otherwise.

After obtaining the optimal matching sequence $\hat{\pi}$, we define the loss function for the entire prediction phase as the sum of the classification loss $L1$ and the boundary loss $L2$.

$$L1 = -\sum_{i=1}^{K} \log P_{\hat{\pi}(i)}^{\text{cls}}(\text{ty}_i) \tag{18}$$

$$L2 = -\sum_{i=1}^{K} 1_{\text{ty}_i \neq \emptyset}[\log P_{\hat{\pi}(i)}^{\text{l}}(l_i) + \log P_{\hat{\pi}(i)}^{\text{r}}(r_i)] \tag{19}$$

$$L_{\text{prediction}}(y, \hat{y}) = \lambda^{\text{cls}} L1 + \lambda^{\text{b}} L2 \tag{20}$$

where λ^{cls}, λ^{b} are loss weights. The total loss calculation is as follows:

$$L_{\text{total}} = L_{\text{BS}} + L_{\text{prediction}}(y, \hat{y}) \tag{21}$$

4 Experiments

4.1 Experimental Setup

Datasets. To verify the effectiveness of our proposed method, we conducted experiments on three widely used nested NER datasets: ACE04 [2], ACE05 [17], KBP17 [4] and one flat NER dataset: CoNLL03 [15]. We split the ACE04 and ACE05 samples into train, dev, and test set by 8:1:1. For KBP17, we divide all the documents into 866/20/167 documents for the train, dev, and test set, respectively. For CoNLL03, a flat NER dataset, we process the data according to Wu et al.[21].

Implementation Details. We use the pre-trained BERT-large-cased model [1] for contextual embeddings, and throughout the experiments, we employ GloVe (100d) [9] as the pre-trained word embeddings. The number of layers is set to 1 for both word-level and character-level BiLSTMs, where the hidden layer size for the former is half of the BERT hidden layer size, while the hidden layer size for the latter is set to 50. The part-of-speech embedding size for each word is set to 25. The batch size is chosen from $\{6, 8\}$. Dropout is chosen from $\{0.1, 0.2\}$. Learning rate is set to 2e-5. For all datasets, we train our model for 40 epochs with the AdamW optimizer and employ a linear warm-up decay learning rate scheduler.

Baselines. To demonstrate the effectiveness and advancements of the proposed SEnNet model in this paper, we selected various models for comparison on the ACE 2004, ACE 2005, KBP 2017, and CoNLL03 datasets. **Locate and label** [10]:Modeling the nested named entity recognition task as a joint task of entity boundary regression and span classification improves the training and inference efficiency of the model. **Sequence-to-Set:** [14]: ignored the order among predicted entities and adopted a non-autoregressive approach for set prediction. **PIQN** [13]: designed a parallel query network capable of querying all entityes in a sentence at once. **PnRNet** [21]: designed a two-stage ensemble prediction network involving span proposal and span optimization to alleviate the challenge of predicting long entities. **Triaffine** [26]: utilized a Triaffine mechanism to integrate four types of heterogeneous information to enrich span re-presentations, thereby improving the accuracy of entity recognition. **W2NER** [7]: constructed a unified entity recognition framework through word pair relationships, capable of simultaneously handling Nested NER, Flat NER, and Discontinuous NER. **PromptNER** [12]:utilized prompt learning for the NER task by constructing multiple prompt templates to achieve parallel querying of entities. **Diffusion-NER** [11]: applied a diffusion model to the NER task, completing entity recognition through the diffusion of entity boundaries. **CNNNER** [23]: utilized CNNs to model the spatial relationships between spans in the scoring matrix. **LHBN** [24]: It captures named entities by generating multiple simpler local hypergraphs, rather than generating a complex full-size hypergraph.

Table 1: Main Results on three nested NER datasets.

Model	ACE04			ACE05			KBP17		
	Pr.	Rec.	F1	Pr.	Rec.	F1	Pr.	Rec.	F1
Sequence-to-Set	88.46	86.10	87.26	**87.48**	86.63	87.05	84.91	83.04	83.96
Locate-and-Label	87.44	87.38	87.41	86.09	87.27	86.67	85.46	82.67	84.05
PnRNet	87.90	88.34	88.12	86.27	89.04	87.63	86.51	84.06	85.27
Triaffine	87.13	87.68	87.40	86.70	86.94	86.82	86.50	83.65	85.05
W2NER	87.33	87.71	87.52	85.03	88.62	86.79	-	-	-
PIQN	88.48	87.81	88.14	86.27	88.06	87.42	85.67	83.37	84.50
CNNNER	87.82	87.40	87.61	86.39	87.24	86.82	-	-	-
PromptNER	87.58	**88.76**	88.16	86.07	88.38	87.21	-	-	-
DiffusionNER	88.11	88.66	88.39	86.15	87.22	86.93	-	-	-
LHBN	88.78	88.13	88.46	86.76	88.93	87.83	**86.84**	86.52	86.55
SEnNet(Ours)	**88.94**	88.70	**88.82**	86.75	**89.07**	**87.90**	86.81	**86.60**	**86.70**

Table 2: Main Results on flat NER dataset CoNLL03. * signifies that the models were reproduced using the same pre-processed data and publicly available code.

Model	Pr.	Rec.	F1
PnRNet*	92.42	92.83	92.62
PIQN	**93.29**	92.46	92.87
W2NER	92.71	93.44	93.07
PromptNER	92.48	92.33	92.41
DiffusionNER	92.99	92.56	92.78
SEnNet(Ours)	93.10	**93.49**	**93.30**

4.2 Main Results

We adopt strict evaluation criteria, considering a predicted entity correct only if both its boundaries and type match exactly. Precision(Pr.), recall(Rec.), and F1 score are used as evaluation metrics. For fairness, we compared our method with baselines that use the same pre-trained model. Table 1 and 2 show the performance of SEnNet and the baseline model on the four datasets. Our method outperforms previous models on four datasets, with F1 scores of 88.82%, 87.90%, 86.70%, and 93.30% on ACE04, ACE05, KBP17, and CoNLL03, respectively. We believe our method is superior to previous methods for the following reasons: (1) The use of word pair information enables span representation to encompass more semantic information, which is beneficial for entity recognition. (2) Using the feature matrix for entity prediction, the dependency information between spans and the contextual information of spans within a sentence are incorporated to enrich span representations, thereby improving entity recognition performance.

4.3 Ablation Studies

To evaluate the effectiveness of several components in our method, we conducted ablation studies on the ACE04, ACE05, and KBP17 datasets. Table 3 lists the results of these studies.

Table 3: Ablation Studies.

Model	ACE04	ACE05	KBP17
SEnNet (Ours)	**88.82**	**87.90**	**86.70**
-w/o Word Pair Information	86.32	86.88	83.67
-w/o Span Interaction	88.61	87.29	85.00
-w/o Contextual Information	88.67	85.97	83.97
Transformer decoder blocks			
M = 0	88.02	87.09	86.33
M = 1	86.77	86.21	84.60
M = 2	88.59	86.58	86.33
M = 3 (Ours)	**88.82**	**87.90**	**86.70**
M = 4	87.98	87.16	86.66

Effectiveness of Word-pair information. In experiments without word pair information, the F1 scores on ACE04, ACE05, and KBP17 decreased by 2.5%, 1.02%, and 3.03%, respectively. One possible reason is that word pair information within the span retains more of the original semantic information, enriching the span representation and improving the performance of entity recognition.

Effectiveness of Span Interaction. In experiments without Span Interaction, the results show that the model performance decreased by 0.21%, 0.61%, and 1.7% on the ACE04, ACE05, and KBP17 datasets, respectively. This demonstrates that establishing dependencies between spans is beneficial for the final entity recognition.

Effectiveness of Contextual Information. In experiments without Contextual Information, the results show that the model performance decreased by 0.15%, 1.93%, and 2.73% on the ACE04, ACE05, and KBP17 datasets, respectively. This indicates that the introduction of contextual information can enrich span representation, which is beneficial for entity recognition.

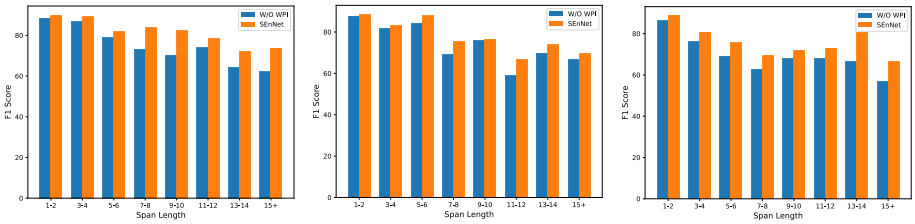
Effectiveness of Transformer decoder blocks. To validate the effectiveness of Transformer decoder blocks on model performance, we conducted experiments with different values of M. When $M = 0$, indicating direct entity prediction with the candidate entity set Q, the experimental results show a decrease of 0.8%, 0.81%, and 0.37% in F1 score for the three datasets ACE04, ACE05, and KBP17, respectively. As the number of decoding layers increases, the F1 score gradually improves, but more layers do not necessarily yield better results as we initially expected; the maximum value is achieved on all three datasets when $M = 3$.

4.4 Analysis of K-value.

We conducted experiments with different K values on the ACE 04 dataset and presented the results in Table 4. The experimental results show that as the K value increases, the F1 score improves. The best performance is achieved when K is set to 60. However, when K continues to increase, performance starts to decline. This can be explained by the fact that increasing K expands the

Table 4: Evaluate model performance using different K values on the ACE 04 dataset.

TOP K	Pr.	Rec.	F1
30	87.72	86.79	87.25
40	88.32	88.21	88.27
50	88.48	88.50	88.49
60	**88.94**	88.70	**88.82**
70	88.12	**88.90**	88.51
80	88.37	87.88	88.13



(a) ACE04 (b) ACE05 (c) KBP17

Fig. 4: F1 scores on spans of different lengths. "W/O WPI" stands for "Without Word Pair Information," indicating that word pair information within the span is not utilized.

interaction window, allowing access to more information and thus improving performance. But as K continues to increase, the larger window also includes more non-entity spans, which can negatively affect training and lead to a drop in performance. Therefore, K = 60 is the optimal value. The same results were observed on other datasets as well.

4.5 Analysis of the effects on long entities

To further analyze the effect of word pair information on long entity recognition, we conducted experiments on three datasets ACE04, ACE05, and KBP17 to compare the performance of entity recognition for entities of different lengths, with and without word pair information within spans, as shown in Fig. 4. The experimental results demonstrate that fully utilizing word pair information within spans leads to improved recognition performance for entities of different lengths. Longer entities contain more word pair information than shorter ones, which leads to a more significant improvement. Taking ACE04 as an example, the enhancement in F1 score for short entity recognition with lengths 1-6 is only 1-3%, while the improvement for long entity recognition can reach 5-12%. Similar effects were observed in the other two datasets as well. From the experimental results, it can be seen that representing spans solely based on their start and end information does not effectively preserve the original semantic information of the spans, especially in the case of long entities. Making full use of the word pair

information inside the span can effectively retain this information and improve the performance of entity recognition.

4.6 Case Study

For a detailed comparison between SEnNet and other span-based methods, we implement PnRNet, one of the advanced methods proposed by Wu et al.[21], which generates span representations by integrating all word information within a span. The F1 scores of the two models for entity recognition across different length ranges are listed in Table 5. The results show that our model achieves leading performance in the vast majority of length ranges, especially in recognizing long entities with lengths of over 10 or more. We believe that a possible reason is that the span representation is generated by simply integrating the intra-span word information without considering the dependency information between intra-span word pairs. Such dependency information helps preserve the original semantic content of the span, improving entity recognition performance, particularly as longer entities contain more word-pair dependency information.

Table 5: F1 scores of PnRNet and SEnNet over different length ranges.

Length	**ACE04**		**ACE05**		**KBP17**	
	PnRNet	SEnNet	PnRNet	SEnNet	PnRNet	SEnNet
1-4	90.14	**90.77**	89.07	**90.18**	**88.96**	88.89
5-9	80.97	**84.08**	**89.08**	87.34	74.62	**76.15**
10-14	57.78	**68**	82.61	**86.96**	68.71	**74.56**
15+	72.34	**73.47**	67.1	**68.66**	59.46	**66.25**

5 Conclusion

This paper proposes a span-based nested NER method. The method utilizes word pair information within spans to construct initial span representations, which are then enriched with dependency information between spans and contextual information to improve the performance of nested NER.The experimental results show that the proposed model effectively leverages both internal and external span information, improving named entity recognition performance. Compared to 10 advanced baseline models, the proposed model achieved the best results on three nested and one non-nested datasets. Additionally, we introduce a word pair information integrating network to capture word pair information within spans. Our analysis of the impact of word pair information on long entity recognition reveals that it helps retain the original semantic information of spans. Therefore, we recommend considering word pair information in span-based NER methods.

References

1. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: 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). pp. 4171–4186 (2019)
2. Doddington, G., Mitchell, A., Przybocki, M., Ramshaw, L., Strassel, S., Weischedel, R.: The automatic content extraction (ACE) program – tasks, data, and evaluation. In: Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04) (2004)
3. Huang, Z., Xu, W., Yu, K.: Bidirectional lstm-crf models for sequence tagging. ArXiv **abs/1508.01991** (2015)
4. Ji, H., Pan, X., Zhang, B., Nothman, J., Mayfield, J., McNamee, P., Costello, C.: Overview of tac-kbp2017 13 languages entity discovery and linking. Theory and Applications of Categories (2017)
5. Kuhn, H.W.: The Hungarian Method for the Assignment Problem, pp. 29–47 (2010)
6. 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. pp. 260–270 (2016)
7. Li, J., Fei, H., Liu, J., Wu, S., Zhang, M., Teng, C., Ji, D., Li, F.: Unified named entity recognition as word-word relation classification. Proceedings of the AAAI Conference on Artificial Intelligence **36**(10), 10965–10973 (2022)
8. 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). pp. 1064–1074 (2016)
9. Pennington, J., Socher, R., Manning, C.: GloVe: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543 (2014)
10. Shen, Y., Ma, X., Tan, Z., Zhang, S., Wang, W., Lu, W.: 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). pp. 2782–2794 (2021)
11. Shen, Y., Song, K., Tan, X., Li, D., Lu, W., Zhuang, Y.: DiffusionNER: Boundary diffusion for named entity recognition. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 3875–3890 (2023)
12. Shen, Y., Tan, Z., Wu, S., Zhang, W., Zhang, R., Xi, Y., Lu, W., Zhuang, Y.: PromptNER: Prompt locating and typing for named entity recognition. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 12492–12507 (2023)
13. Shen, Y., Wang, X., Tan, Z., Xu, G., Xie, P., Huang, F., Lu, W., Zhuang, Y.: 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). pp. 947–961 (2022)
14. Tan, Z., Shen, Y., Zhang, S., Lu, W., Zhuang, Y.: A sequence-to-set network for nested named entity recognition. In: Proceedings of the Thirtieth International

Joint Conference on Artificial Intelligence, IJCAI-21. pp. 3936–3942 (8 2021), main Track
15. Tjong Kim Sang, E.F., De Meulder, F.: Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003. pp. 142–147 (2003)
16. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 6000–6010. NIPS'17 (2017)
17. Walker, C., Strassel, S., Medero, J., Maeda, K.: Ace 2005 multilingual training corpus. Progress of Theoretical Physics Supplement **110**(110), 261–276 (2006)
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. pp. 204–214 (2018)
19. Wang, J., Shou, L., Chen, K., Chen, G.: Pyramid: A layered model for nested named entity recognition. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 5918–5928 (2020)
20. Wang, Y., Li, Y., Tong, H., Zhu, Z.: HIT: Nested named entity recognition via head-tail pair and token interaction. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 6027–6036 (2020)
21. Wu, S., Shen, Y., Tan, Z., Lu, W.: Propose-and-refine: A two-stage set prediction network for nested named entity recognition. In: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22. pp. 4418–4424 (7 2022), main Track
22. Yan, H., Gui, T., Dai, J., Guo, Q., Zhang, Z., Qiu, X.: A unified generative framework for various NER subtasks. 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). pp. 5808–5822 (2021)
23. Yan, H., Sun, Y., Li, X., Qiu, X.: An embarrassingly easy but strong baseline for nested named entity recognition. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 1442–1452 (2023)
24. Yan, Y., Cai, B., Song, S.: Nested named entity recognition as building local hypergraphs. Proceedings of the AAAI Conference on Artificial Intelligence **37**(11), 13878–13886 (2023)
25. Yu, J., Bohnet, B., Poesio, M.: Named entity recognition as dependency parsing. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 6470–6476 (2020)
26. Yuan, Z., Tan, C., Huang, S., Huang, F.: Fusing heterogeneous factors with triaffine mechanism for nested named entity recognition. In: Findings of the Association for Computational Linguistics: ACL 2022. pp. 3174–3186 (2022)
27. Zhang, S., Shen, Y., Tan, Z., Wu, Y., Lu, W.: De-bias for generative extraction in unified NER task. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 808–818 (2022)
28. Zhu, E., Li, J.: Boundary smoothing for named entity recognition. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 7096–7108 (2022)