Finding Cycles in Graph: A Unified Approach for Various NER Tasks

Liang Qiao*
Zhejiang University
Hangzhou, China
qiaoliang@zju.edu.cn

Pengfei Li*
Xi'an Jiaotong Univerisity
Xi'an, China
lpf641081298@stu.xjtu.edu.cn

Ting Jin
Hainan Univerisity
Haikou, China
jinting@hainanu.edu.cn

Xi Li
Zhejiang University
Hangzhou, China
xilizju@zju.edu.cn

***Abstract*—Named Entity Recognition (NER) is the task of recognizing the entities' locations and types in text, which can be generally categorized into flat NER, overlapped NER, and discontinuous NER. Most previous methods are usually designed specifically for one of the tasks, such as sequence labeling approaches for flat NER and span-based models for overlapped NER. Recently, some new work has begun to propose the unified NER framework that can addresses all three scenarios simultaneously. However, there still has room for improvement in some complex scenarios (long/discontinuous entity). In this paper, we propose a concise framework that supports all types of NER tasks, where entities can be represented by unique cycles that are formed by the directed edges among tokens in the graph. The model integrates a Graph Feature Enhancement module to extract correlations at both the node-level and edge-level. At the node-level, the features are enhanced in the binary and ternary token relations. In edge-level, the model will go further to enhance the relations among token pairs using deformable convolutions. Furthermore, to benefit the completeness of cycle formation, we also propose a novel Cycle Loss that optimizes the independent edge classification in the group of cycles from a global perspective. Experimental results show that our model can achieve competitive and even new state-of-the-art performance on eight popular NER benchmarks, including flat NER, overlapped NER, and discontinuous NER.**

I. Introduction

Named Entity Recognition (NER) intends to identify entities with predefined semantic types (*e.g.*, person, locations, organization, etc.) from the text, which is a fundamental task in natural language processing (NLP), and plays an essential role in many downstream tasks, including information retrieval [1], question answering [2], etc. Generally, NER tasks can be divided into three categories [3], [4] based on whether containing overlapped or nonadjacent entities, named flat NER, overlapped NER[1], and discontinuous NER. The overlapped NER contains some entities that share the same tokens, and the discontinuous NER contains entities composed of several fragments.

Most of the previous methods are designed for one specific type of NER task. They can be roughly divided into three types: *sequence-labeling*, *span-based*, and *hypergraph-based* methods. Sequence labeling [5], [6] is the most commonly used method in flat NER, where each token is assigned a

* Authors contributed equally.

[1] 'Nested NER' can be treated as a special case of 'overlapped NER'.

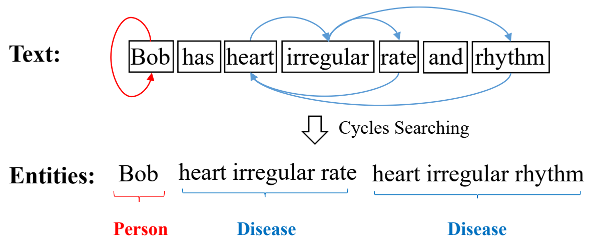


Fig. 1: Illustration of the proposed schema that represents entities in cycles. In this example, 'Bob' is a flat entity. 'heart irregular rate' and 'heart irregular rhythm' belong to both overlapped and discontinuous types.

label that represents its entity type. These methods are usually challenging to apply to overlapped and discontinuous entities directly. Spans prediction [7], [8] is one of the dominating methods used in handling the problem of overlapped entities. [4], [9] extend the span-based methods into the two-staged framework that predicts the entities' spans and their relations to fit discontinuous NER. The hypergraph-based methods [10], [11] are more flexible to overlapped and discontinuous NER, but they usually depend on the manually defined graph structure and suffer from the structure ambiguity problem [12].

Although the methods designed for discontinuous NER [4], [9], [13] can theoretically support all forms of tasks, they rarely test the model on the flat NER benchmarks. Recently, some works have been proposed to solve the problem in a unified framework, achieving the competitive performance on all three types of NER benchmarks. [3] proposes a generative sequence-to-sequence framework to predict the entity span sequences. However, the generative schema cannot parallel decode and is hard to train if the data is insufficient. [14] reformulates the NER sub-tasks as the problem of word-word relation classification, which is flexible for all kinds of entities. Nevertheless, the optimization target of the relations for every two words is independent, also easily falling into the spurious structure problem.

Inspired by [14], we adopt a similar training target to predict the relations between every two words. Differently, in this paper, we approach this problem in a new light, that is, to formulate the NER problem as a task to find all cycles in a directed graph, as illustrated in Figure 1. Each token in the text can be regarded as a node in a graph. Tokens that belong to the same entity will be assigned with directed edges to their next tokens, and the last token will be linked to the first one. The token will point to itself in entities with a single token. Entities belonging to different types will be optimized in different channels. In this way, entities could be represented by unique cycles, and the NER task can be transferred into a single target of predicting the edges in different categories in a graph.

Aware that the spurious structure may become the main challenge in graph edge learning, where a single missed edge will cause an entity's false recall. In this paper, we propose the framework named CycleNER to benefit the complete formation of cycles in the graph. The CycleNER integrates the strategies in the implicit feature enhancement and the explicit cycle-level optimization constraint. Specifically, we adopt a Graph Feature Enhancement (GFE) module after the BERT-based encoder [15] to construct the correlation information among tokens. From the perspective of the graph, it enhances the correlation features in both the one-dimensional node level and two-dimensional edge level. Moreover, we propose a Cycle Loss to constrain the prediction to form the same number of cycles with the ground truth, making the model more inclined to form edges in terms of complete cycles and suppressing the prediction of spurious edges. The proposed framework is one-staged that effectively eliminates the error accumulation problem.

The major contributions of this paper are as follows: (1) We propose a concise and uniform task schema to predict cycles in the graph that can handle any NER subtasks. (2) We design a Graph Feature Enhancement Module to capture the correlation features in the graph, including node-level and edge-level. (3) We propose a Cycle Loss to benefit the formation of the complete cycles from the global perspective. (4) Extensive experiments show that our method achieves competitive and even state-of-the-art results on different types of NER benchmarks.

II. Related Work

A. Named Entity Recognition

Named Entity Recognition (NER) has been studied for decades. Among them, sequence labeling, span-based, and hypergraph-based methods are three mainstream types.

1) Sequence labeling methods.: These models consider the NER task as a token-wised classification problem [5], [6], [16]. These methods assign each token a tag with the position marker such as BIO or BIEOS. CRF methods [17], [18] are usually used in decoding to better capture the tokens' correlations. Some sequence labeling methods [16], [19], [20] can also be customized to fit overlapped and discontinuous NER tasks by designing more complex label schemes like BIOHD.

2) Span-based methods.: Span-based methods [7], [8], [21]–[25] are the most common approach to handling the situation with overlapped entities by classifying the enumerated spans. To fit the discontinuous NER situation, they usually adopt another spans relations prediction module [4], [9]. This type of method usually requires enumerating all possible spans, resulting in the large model complexity.

3) Hypergraph-based methods.: [26] first proposes to cope with the overlapped NER problem by modeling arbitrary combinations of mentions in a hypergraph. After that, [12] extends the framework for overlapped and discontinuous entities, and [10], [11] utilize deep neural networks to enhance the model. However, these models usually rely on the manually designed graph and easily fall into the spurious structure problem.

Most methods are designed for the specific NER scenario. Some approaches also aim to solve NER problems in a unified framework. [3] proposes a unified generative framework that reformulates the NER subtasks as an entity span sequence generation task. However, it requires higher computing resources and also cannot parallel decode. [14] reformulates the NER subtasks as word-word relation classification, which has a similar optimization target to our proposed method. However, it optimizes the word-word relation independently, while our proposed method can benefit the formation of the complete cycles from a global perspective.

B. Edge Prediction in Graph

Modeling tasks as the edge prediction in a graph is commonly used to solve the correlation extraction problem. [27] treats the relationship between token pairs as edges for the relation extraction task and reformulates relation extraction as edge prediction in a graph. In the dependency parsing domain, [28] proposes a biaffine attention module to extend the graph-based method [29] and obtain the state-of-the-art performance. For the NER task, [7], [8], [14] predict the entity head and tail's link in the graph composed of tokens. Our method follows a similar task formulation and essentially enhances the model by considering the characteristics of the graph.

III. Methodology

A. Overview

This paper presents to transfer all NER tasks into a unified learning schema that finds all cycles in a graph. The model finally predicts the binary relations among tokens in a text sequence, denoted as edges. We assign each token with a directed edge to its next token for every entity, and the last token is pointed to the entity's head. The entity with a single token will be pointed to itself.

To predict the edges from the view of cycles, we propose the unified NER framework named CycleNER as illustrated in Figure 2. First, the input token sequence is fed into the BERT-based [15] encoder to obtain the embedded representations. Then, each token is treated as a node in the graph. A Graph Feature Enhancement (GFE) module is integrated to extract the feature further and build the correlation information among tokens. The GFE module enhances the correlations in both the node-level and the edge-level. Last, the network conducts the Binary Graph Prediction to obtain the final token relations in different categories. The model will predict the number

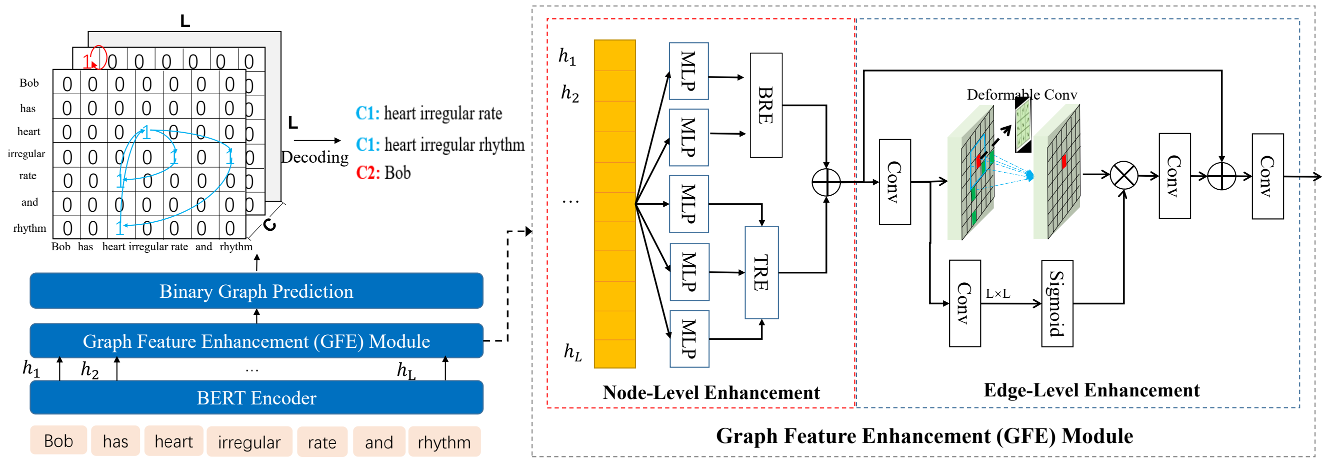


Fig. 2: Illustration of the proposed CycleNER framework. The input token sequence is first fed into the BERT-based encoder, then enhanced by a Graph Feature Enhancement (GFE) module, and finally predicted as a binary graph to represent tokens' relations. The right part shows the detailed GFE module, which includes node-level and edge-level correlation enhancements.

of channels' graphs for different entity types. Then we can conduct the cycle searching to obtain the final entities.

B. *Graph Feature Enhancement(GFE) Module*

The task of edge prediction is an independent task for each edge, easily falling into the spurious structure problem [12]. The missing of just one edge will result in a cycle not forming.

To benefit the completeness of the cycles, we need to enhance the correlation information for the elements belonging to the same entity. From the perspective of the graph, we have to make the network fully capture the correlations among the nodes and edges. Here, we propose a Graph Feature Enhancement(GFE) module to enhance both levels' correlations.

Specifically, we denote the hidden state of the BERT encoder's output as $H{=}\{h_1, h_2, ..., h_L\}{\in}\mathbb{R}^{L\times d}$, where L is the maximum sequence length, and d denotes the dimension. The GFE module then enhances the feature map in two phases, node-level and edge-level correlation enhancement. Finally, it outputs the feature with the shape of $\mathbb{R}^{C\times L\times L}$, where C is the number of entity categories.

The whole module can divided into two parts: node level enhancement and edge level enhancement.

1) Node-level Correlation Enhancement.: In the node-level correlation enhancement process, we define two operations: the *Binary Relation Extraction* (BRE) and *Ternary Relation Extraction* (TRE).

The BRE aims to model the basic correlation information between every two nodes, including the information aggregation by point-wise feature concatenation and the feature distance represented by the fused rotary embedding [30]. Specifically, the sequence feature is firstly fed into two independent Multi-Layer-Perceptrons (MLP):

$$\mathcal{F}^Q, \mathcal{F}^K = MLP^{Q,K}(H), \tag{1}$$

where $\mathcal{F}^Q, \mathcal{F}^K{\in}\mathbb{R}^{L\times C'\times d}$ and C' is the channel's number of the middle layers. We use $\mathcal{F}^A_{ij}, \mathcal{F}^D_{ij}{\in}\mathbb{R}^{C'\times 1\times 1}$ and separately denote the channel-wise vector of the aggregation and distance feature ($\mathcal{F}^A, \mathcal{F}^D{\in}\mathbb{R}^{C'\times L\times L}$) for every two nodes $0{\le}i, j{\le}L$, which can be formed as:

$$\mathcal{F}^A_{ij} = (\mathcal{F}^Q_i \oplus \mathcal{F}^K_j)W^A, \tag{2}$$

$$\mathcal{F}^D_{ij} = ((\mathcal{F}^Q_i R_i) \otimes (\mathcal{F}^K_j R_j))W^D, \tag{3}$$

where $\oplus$ is the concatenation operation, $\otimes$ is the element-wise inner product operation, $W^A{\in}\mathbb{R}^{2d\times 1}$ and $W^D{\in}\mathbb{R}^{d\times 1}$ are learnable parameters, and $R_i, R_j{\in}\mathbb{R}^{d\times d}$ are the rotary matrixes [30]. The BRE's output can be calculated by point-wised addition of the two features:

$$\mathcal{F}^{BRE} = \mathcal{F}^A + \mathcal{F}^D. \tag{4}$$

Inspired by significant accuracy gains using high-order modelling [31], we further conduct the TRE that adopts triaffine operation to get the relations for every three nodes $0{\le}i, j, k{\le}L$:

$$\mathcal{F}^X, \mathcal{F}^Y, \mathcal{F}^Z = MLP^{X,Y,Z}(H), \tag{5}$$

$$\mathcal{T}_{ijk} = {\mathcal{F}^X_i}^{\top} {\mathcal{F}^Y_j}^{\top} W^{\mathcal{T}} \mathcal{F}^Z_k, \tag{6}$$

where $W^{\mathcal{T}}{\in}R^{d\times d\times d}$ is a three-way tensor, and $\mathcal{T}_{ijk}$ is the channel-wise feature vector of the triaffine feature $\mathcal{T}{\in}\mathbb{R}^{C'\times L\times L\times L}$. We reduce the feature's dimension into $\mathbb{R}^{C'\times L\times L}$ by conducting the Global Max Pooling operation:

$$\mathcal{F}^{TRE} = GlobalMaxPooling(\mathcal{T}). \tag{7}$$

Finally, the enhanced node-level features can be represented as the addition of these two features:

$$\mathcal{F} = \mathcal{F}^{BRE} + \mathcal{F}^{TRE}. \tag{8}$$

2) Edge-level Correlation Enhancement.: After the node-level correlation enhancement module, we can get the rich characteristics of the edges for every two nodes in the graph. For an entity represented by a cycle, the edges should also not be considered independently. For example, if the network perceives edges from token A to B and B to C, there will be a higher probability of producing the relation between C and A. Therefore we introduce a *Deformable Attention* module to further capture the correlations among edges, which is illustrated in the right part of Figure 2.

Specifically, we firstly use a 1×1 convolution to get the initial edge feature:

$$\mathcal{F}_{edge} = conv_{1\times1}(\mathcal{F}) \tag{9}$$

Observing that the related positions for entities vary in shapes and sizes, we conduct two 3×3 deformable convolution layers [32] to further enhance the feature:

$$\mathcal{F}_{dcn} = dconv_{3\times3}(dconv_{3\times3}(\mathcal{F}_{edge})) \tag{10}$$

Motivated by the channel-wise attention technique in SENet [33], we want our network can dynamically learn the importance of different edges, i.e., improving the useful edge features. Here, we integrate a point-wise attention module on the feature map as follows:

$$\mathcal{F}_{att} = \mathcal{F}_{dcn} \otimes (sigmoid(conv_{3\times3}(\mathcal{F}_{edge}))), \tag{11}$$

where $conv_{3\times3}$ denotes the 3×3 convolution operation, which will generate a feature map with shape of $\mathbb{R}^{L\times L}$ to serve as attention map.

Finally, the output of the edge-level correlation enhancement process as well as the GFE module $\mathcal{F}_{out}\in\mathbb{R}^{C\times L\times L}$ can be calculated as follows,

$$\mathcal{F}_{out} = conv_{1\times1}(conv_{1\times1}(\mathcal{F}_{att}) + \mathcal{F}). \tag{12}$$

C. Binary Graph Prediction and Decoding

In the Binary Graph Prediction, the model predicts the output score map by simply conducting *sigmoid* operation:

$$\mathcal{Y} = sigmoid(\mathcal{F}_{out}). \tag{13}$$

The final binary map can be obtained using the binarization threshold of 0.5.

In the decoding stage, we adopt [34] to find all cycles in graphs with the time complexity of $O((n+e)(c+1))$, where n, e, c are the number of nodes, edges, cycles in the graph, respectively. Every cycle will be transferred into the final entity in different categories. For each cycle, we take the smallest position index as the entity head and follow the edges' directions to form the final entity.

Specially, if the prior information exists that all entities are continuous, we may also simplify the cycle searching process by finding the right-to-left edges. This would fall into a similar schema with span-based methods. Nevertheless, the model still provides strong correlation information in entities.

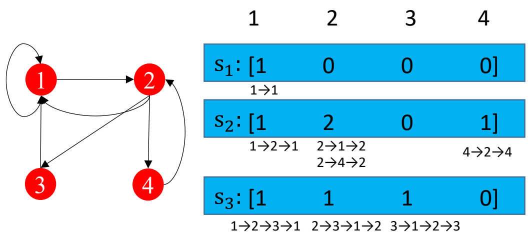


Fig. 3: An example of $\boldsymbol{s}_k$ that represents the vector that contains the number of cycles (exclude nested cycles) with the length k.

D. Optimization with Cycle Loss

The CycleNER aims to learn a binary classification task for every two tokens, whose ground truth can be easily generated according to the entities and their categories.

The binary-cross-entropy (BCE) loss is usually adopted in graph edges classification, which can be formed as:

$$L_{bce} = -\frac{1}{NCL^2}\sum_{i=1}^{N}\sum_{c=1}^{C}\sum_{m=1}^{L}\sum_{n=1}^{L}\boldsymbol{y}_{icmn}\log\hat{\boldsymbol{y}}_{icmn}, \tag{14}$$

where N denotes the number of samples, $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ represent the binary vectors of ground truth and the model's output logits, respectively.

1) Cycle Loss.: The BCE function calculates the loss for each edge independently, which usually causes the mismatch problem with the real optimization targets. Different edges' false positives or negatives will impact the final results in different degrees. For example, if the model falsely predicts another edge from '*rhythm*' to '*irregular*' in the example illustrated in Figure 1, it will generate another decoded entity of '*irregular rhythm*'. However, if the error edge is from '*rhythm*' to '*rate*', it will generate the same loss as the previous example but cannot form any other cycles. In fact, edges in the graph have different degrees of importance. Some edges are used multiple times in forming entities, while others might be unimportant. Therefore, we propose a *Cycle Loss* to optimize the network from the granularity of the complete cycles.

The basic idea of Cycle Loss is to match the predicted cycles number with the ground truth in all positions. Specifically, given a graph G composed of cycles, we denote its adjacency matrix as A_G. We use $\boldsymbol{s}_k\in\mathbb{R}^L$ to represent the vector that contains the number of cycles with the length k at each node position, as an example shown in Figure 3. It is easy to know that $\boldsymbol{s}_1$ is exactly the vector formed by A_G's diagonal elements, which is formed as $\boldsymbol{s}_1 = diagonal(A_G)$.

To compute the value of any index i in $\boldsymbol{s}_2$, we need to count all paths that transferred out from node i to another node j and then transferred back to i in one step. Similarly, to calculate $\boldsymbol{s}_k$ for any k, we can define the transition matrix $\mathcal{T}_k$. The element of row i and column j in $\mathcal{T}_k$ represents the number of existing paths with length k from node i to j, without passing through

i. It can be counted by all paths from i to another node m in $k-1$ steps if there is a 1-step edge from m to j, for any $m \neq i, j$. The process can be represented by the multiplication of the $(k-1)$-step transition states with a 1-step transition matrix as follows,

$$\mathcal{T}_k = (\mathcal{T}_{k-1} - M(diagonal(\mathcal{T}_{k-1})))\mathcal{T}_1, \tag{15}$$

$$\mathcal{T}_1 = (\mathcal{A}_G - M(diagonal(\mathcal{A}_G))), \tag{16}$$

where $M(\boldsymbol{s})$ returns a 2-D matrix that uses $\boldsymbol{s}$ as diagonal elements and fills others as 0. For the transition matrix in each step, it removes the diagonal elements to prevent from counting nested cycles. Then, we can obtain the $\boldsymbol{s}_k$ $(k>1)$ as:

$$\boldsymbol{s}_k = diagonal(\mathcal{T}_k). \tag{17}$$

Finally, we can define the *Cycle Loss* to make model be optimized to match with the ground truth's cycle numbers in each position as follows,

$$L_{cycle} = \sum_{k=1}^{K}\sum_{i=1}^{L}(\hat{\boldsymbol{s}}_{k,i} - \boldsymbol{s}_{k,i})^2, \tag{18}$$

where the K is a parameter to control the considered maximum cycle length and $\hat{\boldsymbol{s}}$ is the ground truth vector. Notice that the long entities need to be formed by more edges. The cycle loss assigns them higher training weights since the global constraint is imposed on each cycle node. In this way, the cycle loss can effectively balance the learning weights of entities with different lengths.

The overall loss of CycleNER can be formed as:

$$L = L_{bce} + \lambda L_{cycle}. \tag{19}$$

where λ is the parameter to balance loss weight.

IV. Experiment

A. Datasets & Implementation Details

To evaluate our model for various NER subtasks, we conducted experiments on eight datasets included flat, overlapped and discontinuous NER datasets:

1) Flat NER datasets.: We adopt CoNLL-2003 [38] and OntoNotes [39] as the flat NER benchmarks. For CoNLL-2003, we follow [3], [7] to train the models on the concatenation of the training and development sets. For OntoNotes, we use the same data split as [3], [7], and the New Testaments portion was excluded since there is no entity in this portion.

2) Overlapped NER datasets.: We use ACE2004 [40], ACE2005 [41], and GENIA [42] to conduct our experiments. For ACE2004 and ACE2005, we follow [3], [7] to split the data as 8:1:1 for training, development, and testing, respectively. For GENIA, we follow [3] to use five types of entities, and the data split ratio is 8.1:0.9:1.0.

3) Discontinuous NER datasets.: We adopt CADEC [43], ShARe13 [44] and ShARe14 [45] to evaluate the model on discontinuous entities. We follow [3], [13] to only use Adverse Drug Events(ADEs) entities.

We set the maximum input sequence length for the GENIA dataset as 256 and the other as 160. In the training stage, all models are trained by the AdamW [46] optimizer with $batch\_size$=8. We set the initial learning rate as $5e{-}5$ for the encoder and other parts as $1e{-}3$. We use the slanted triangular learning rate warm-up for the first 10 epochs of total 30 epochs. The parameter of channel's number of the middle layer C' is set as 64, maximum cycle length K is set as 10 and the loss weight λ=1. Besides, we adopt the vanilla BERT-Large [15] as the encoder and the corresponding *WordPiece* tokenizer to convert each word into word pieces. All experiments are conducted on two 32GB Tesla-V100 GPUs.

B. Results

We adopt the span-level F1 score as the evaluation metric on all datasets, and the results are as follows.

1) Results on Flat NER.: First, we conduct the experiments on two popular flat NER datasets, whose results are shown in Table I. Although our model does not achieve the best performances, the results are still competitive, and the precisions of our model obtain the highest value. All edges on the cycles need to be recalled in the decoding stage of our model. It benefits the complete entities precision, but it will also affect the recall to some extent.

2) Results on overlapped NER.: The experiments on three overlapped NER benchmarks are compared with some recent advances including *sequence-labeling*, *span-based* and *unified* methods. Since some methods adopt another domain-specific pre-trained model, we report the re-implemented results for fair comparison using the same pre-trained model. As the results illustrated in Table II, our model achieves the best Precision and F1 score on the ACE2004 and ACE2005 compared with other methods. It outperforms the previous best F1 scores by 0.56% and 0.58%, respectively. Our model does not achieve the state-of-the-art performance on the GENIA dataset. It is mainly because entities in GENIA have a longer average word-pieces length (41) compared with ACE2004 (33) and ACE2005 (29). If there are more nodes and edges in the graph, the positive edges that need to be predicted are more sparse, and thus the network cannot be optimized sufficiently unless more weighting balanced strategies are adopted.

Since the above datasets also include flat entities, we further investigate the performances of our model on recognizing only overlapped entities. We additionally evaluate the performance on the test set that is composed of sentences including overlapped entities, and the performance that only considers overlapped entities. As the results are shown in Table III, our method achieves much better performance in handling overlapped entities on all datasets.

3) Results on discontinuous NER.: Table IV shows the comparison results between our model and others on three

Types	Methods	Pretrained-models	CoNLL2003			OntoNotes		
			P	R	F1	P	R	F1
Sequence-labeling	(Peters et al., 2018) [35]	[ELMO]	-	-	92.22	-	-	-
Sequence-labeling	(Devlin et al., 2019) [15]	[BERT-Large]	-	-	92.80	-	-	-
Span-based	(Yu et al., 2020) [7]†	[BERT-Large]	92.85	92.15	92.50	89.92	89.74	89.83
Span-based	(Li et al.,2020) [8]]†	[BERT-Large]	92.47	93.27	92.87	91.34	88.39	89.84
Hypergraph-based	(Wang and Lu, 2018) [10]	[Glove]	-	-	90.50	-	-	-
Unified	(Yan et al., 2021) [3]	[BART-Large]	92.61	**93.87**	**93.24**	89.99	90.77	90.38
Unified	(Li et al.,2022) [14]	[BERT-Large]	92.71	93.44	93.07	90.03	**90.97**	**90.50**
Unified	CycleNER(ours)	[BERT-Large]	**93.50**	91.91	92.70	**91.69**	88.68	90.16

TABLE I: Results for flat NER datasets. Results with "†" are reported from [3].

Types	Methods	Pretrained-models	ACE2004			ACE2005			GENIA		
			P	R	F1	P	R	F1	P	R	F1
Sequence-labeling	(Straková et al., 2019) [36]	[BERT-Large]	-	-	84.33	-	-	83.42	-	-	78.20
Sequence-labeling	(Shibuya and Hovy, 2020) [20]	[BERT-Large]	-	-	-	83.30	84.69	83.99	77.46	76.65	77.05
Span-based	(Yu et al., 2020) [7]†	[BERT-Large]	85.42	85.92	85.67	84.50	84.72	84.61	79.43	78.32	78.87
Span-based	(Li et al., 2020) [8]†	[BERT-Large]	85.83	85.77	85.80	85.01	84.13	84.57	**81.25**	76.36	78.72
Span-based	(Shen et al., 2021) [25]∗	[Glove&BERT-Large]	87.20	87.26	87.23	86.24	86.54	86.39	78.47	79.19	78.83
Span-based	(Tan et al., 2021) [37]∗	[Glove&BERT-Large]	87.86	85.63	86.73	86.88	86.15	86.51	80.73	77.20	78.93
Span-based	(Li et al., 2021a) [4]∗	[BERT-Large]	86.58	86.10	86.34	83.11	85.39	84.23	78.88	77.31	78.09
Hypergraph-based	(Wang and Lu, 2018) [10]	[Glove]	78.00	72.40	75.10	76.80	72.30	74.50	77.00	73.30	75.10
Unified	(Yan et al., 2021) [3]	[BART-Large]	87.27	86.41	86.84	83.16	86.38	84.74	78.87	**79.60**	**79.23**
Unified	(Li et al., 2022) [14]∗	[BERT-Large]	87.90	87.08	87.49	84.78	**88.04**	86.38	80.55	77.32	78.90
Unified	CycleNER(ours)	[BERT-Large]	**88.45**	**87.99**	**88.22**	**87.48**	86.47	**86.97**	79.23	77.48	78.35

TABLE II: Results for overlapped NER datasets. Results with "†" are reported from [3], and "∗" means our re-implemented result with the BERT-Large pretrained model.

Models	ACE2004	ACE2005	GENIA
(Yan et al., 2021) [3]	70.64/-/-	79.69/-/55.0	80.34/-/52.7
(Dai et al., 2020) [13]	69.0/65.4/37.9	77.7/62.9/52.5	79.6/63.1/49.2
(Li et al., 2022) [14]∗	87.49/87.34/82.41	86.38/87.05/77.62	**78.90**/75.45/38.52
CycleNER (ours)	**88.22/88.53/83.25**	**86.97/87.79/78.51**	78.35/**75.50/38.83**

TABLE III: Performance on Overlapped Entities, '/' separates the overall results, the result of sentences including overlapped entity, and the result only considers overlapped entities.'∗' means our re-implemented result with the BERT-Large pre-trained model.

discontinuous NER datasets. We can see that our CycleNER achieves the new state-of-the-art performances on all three dataset. Our method surpasses the best results by 4.74%/1.16%/0.34% in Precision and 1.27%/0.74%/0.86% in terms of the F1 score on three datasets, respectively. The method of [3] obtains the highest Recall. However, it depends on a different pre-trained model adapted for generative framework, which is not comparable to a certain degree.

Since only about 10% of entities are discontinuous on the three datasets, we also evaluate the effectiveness of our method on recognizing complicated discontinuous entities. Following the same evaluation setting as [3], [13], we additionally evaluate the performance on the test set that is composed of sentences with as least one discontinuous entity, and the performance that only considers discontinuous entities. As the results are shown in Table V, our method achieves much better performance in handling discontinuous entities.

In summary, our model can achieve competitive and even new state-of-the-art performance on various NER datasets especially overlapped and discontinuous NER datasets. For the flat NER, the task is relatively easy, and almost all previous methods' performances are similar. The data distribution has determined the performance upper limits to some extent unless introduces outside knowledge or heavier structure. Our model's performance in those simple NER scenarios is competitive, while the complex situation can more reflect the model's performance advantages.

C. Ablation Studies & Analysis

1) Components Ablation.: We conducted the components ablation experiments by removing the individual components separately and re-evaluating the performance on three discontinuous NER datasets. The result is shown in Table VI. In the GFE module, since the node-level correlation enhancement serves the role of transferring the sequence in a two-dimensional matrix, we do the ablation on the BRE/TRE separately. The model without BRE will have a severe performance drop from the result. It is because BRE provides the basic information required by the edge classification. The TRE captures high-order correlation information that can improve the whole model's performance, but it has difficulty in convergence when used solely. Besides, the edge-level correlation enhancement also contributes to the performance since consistent performance drops exist after excluding it. The same result also occurs by removing the Cycle Loss, and the F1 score will decrease by 1.58%/0.71%/0.50% on three datasets, respectively.

Moreover, we find that the proposed components contribute a higher performance gain in CADEC than the other two datasets. This is because entities in CADEC have a larger average length (2.74) than the others (ShARe13 with 1.72 and ShARe14 with 1.63). It can somehow reflect that the proposed modules can effectively extract the integrated correlations among long entities.

Types	Methods	Pretrained-models	CADEC			ShARe 13			ShARe14		
			P	R	F1	P	R	F1	P	R	F1
Sequence-labeling	(Tang et al., 2018) [16]	[Glove]	67.80	64.99	66.36	-	-	-	-	-	-
Span-based	(Wang et al., 2021) [9]∗	[BERT-Large]	70.50	72.50	71.50	83.25	76.46	79.71	78.20	83.65	80.83
Span-based	(Li et al., 2021a) [4]∗	[BERT-Large]	70.10	69.05	69.57	82.50	76.79	79.54	79.23	81.10	80.15
Hypergraph-based	(Wang and Lu, 2019) [11]	[Word-Embedding]	72.10	48.40	58.00	83.80	60.40	70.30	79.10	70.70	74.70
Others	(Dai et al., 2020) [13]	[ELMO]	68.90	69.00	69.00	80.50	75.00	77.70	78.10	81.20	79.60
Others	(Fei et al., 2021) [47]∗	[BERT-Large]	73.11	70.25	71.65	83.68	76.23	79.78	78.23	82.62	80.37
Unified	(Yan et al., 2021) [3]	[BART-Large]	70.08	71.21	70.64	82.09	**77.42**	79.69	77.20	**83.75**	80.34
Unified	(Li et al., 2022) [14]∗	[BERT-Large]	72.02	70.28	71.14	82.63	76.75	79.58	79.96	80.14	80.05
Unified	CycleNER(ours)	[BERT-Large]	**73.49**	**71.49**	**72.48**	**84.96**	76.35	**80.43**	**80.30**	82.13	**81.20**

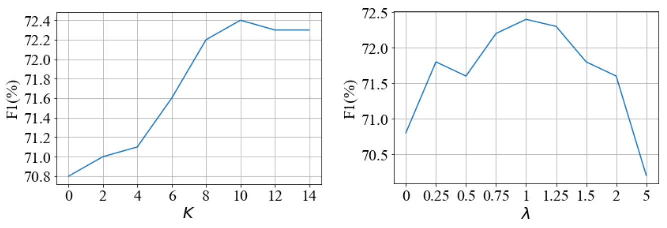
TABLE IV: Results for discontinuous NER datasets. '∗' means our re-implemented with the BERT-Large pretrained model.

Models	CADEC	ShARe13	ShARe14
(Yan et al., 2021) [3]	70.64/-/-	79.69/-/55.0	80.34/-/52.7
(Dai et al., 2020) [13]	69.0/65.4/37.9	77.7/62.9/52.5	79.6/63.1/49.2
(Li et al., 2022) [14]∗	71.1/67.8/45.1	79.6/65.8/56.7	80.1/65.5/49.9
CycleNER (ours)	**72.4/71.3/48.8**	**80.4/66.6/57.5**	**81.2/69.3/53.9**

TABLE V: Performance on Discontinuous Entities, '/' separates the overall results, the result of sentences with at least one discontinuous entity, and the result only considers discontinuous entities.'∗' means our re-implemented result with the BERT-Large pretrained model.

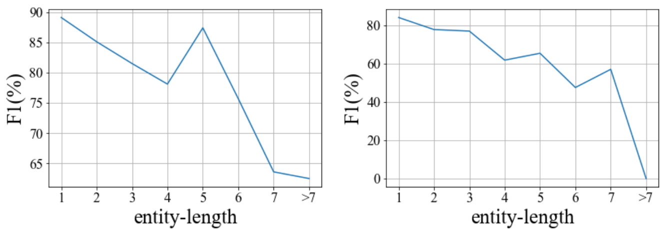
	CADEC	ShARe13	ShARe14
CycleNER (ours)	**72.48**	**80.43**	**81.20**
-BRE	14.49(↓57.99)	19.08(↓61.35)	10.39 (↓70.63)
-TRE	71.41(↓1.07)	79.90(↓0.53)	80.78(↓0.24)
-Edge-level Enhancement	71.85(↓0.63)	79.84(↓0.59)	80.64(↓0.38)
-Cycle Loss	70.90(↓1.58)	79.72(↓0.71)	80.50(↓0.50)

TABLE VI: Ablation results on discontinuous NER datasets, '-' denotes remove the component alone.



(a) Impact of K in Eq. (18). (b) Impact of λ in Eq. (19).

Fig. 4: The impact of hyper-parameters K and λ on CADEC.



(a) ACE2005 (b) ShARe13

Fig. 5: Impact of entity length on ACE2005 and ShARe13

2) Hyper-parameters Ablation.: Here, we conduct the following experiments to evaluate the influence of parameters.

First, we explore the effect of the hyper-parameter K representing the maximum cycle length in Equation (18), and Figure 4(a) presents the result on the CADEC. We can see that with the increases of K, the F1 score first goes up and then tends to be stable after $K\geq10$. The larger K means more cycles (entities composed of more pieces) could be included during training, but the values in $\boldsymbol{s}_k$ will also become more and more sparse. The optimal values of K are different according to the characteristics of the datasets.

Next, We ablate the hyper-parameter of the loss balancing weight λ in Equation (19), and the result is show in Figure 4 (b). In our setting, the BCE loss trains the model in line with the main optimization objectives, while the Cycle Loss serves as auxiliary optimization that makes the model with better global awareness. We can see that small λ weakens the impact of global optimization from complete cycles, while larger λ puts Cycle Loss in the dominant position, making the network difficult in sufficient convergence. In this experiment, the model obtains the highest performance when λ=1.

3) Impact of Entity Length: The predicted entities are constructed based on the complete cycle prediction in the graph, which requires all of the word-level edges to be predicted in inference. It would bring challenges to our method when meeting the long entities, *e.g.*, there would be even more than 15-word pieces for an entity with a length larger than 7. We separately calculate the model's performance on different entity lengths of two datasets that contain a certain number of long entities (ACE2005 and ShARe13). The results are illustrated in Figure 5. From the results, we can find that the model's performance goes down along with the entity's length increased. It is because there are very few long entities in the training set, and it is also a common challenge to all current NER methods [13]. Nevertheless, from the results demonstrated in the ablation experiments in the main paper, we can see our proposed module obtains more performance gain in the dataset with a longer average entity length.

4) Model Complexity Comparison: The parameter number of current CycleNER model is 321.5M, where the encoder of BERT-Large is 319.6M and other parts only cost 1.9M. For [3], it uses BART-Large as pre-trained model, and the total parameter number is more than 400M. For [14], if it also uses BERT-Large as encoder, the parameter number would be about 328M. It can be seen that the main calculation of the current model is mainly form the encoder. Our decoder is currently a very lightweight design compared with other methods. The parameters are mainly come from 1) the MLP layers, the rotary matrix in node level (biaffine and triaffine operations will not involves new parameters), and 2) the convolution layers in edge level.

V. Conclusion

This paper proposed a concise framework named CycleNER to uniformly tackle flat, overlapped, discontinuous NER by converting the entity index sequences into cycles prediction in the graph. To get the rich graph feature, we proposed a Graph Feature Enhancement (GFE) module composed of the correlation information extraction in both node-level and edge-level. Furthermore, we design a Cycle Loss that aims to overcome the spurious graph structure problem and prompt complete cycles formation. We evaluate our method on eight datasets, including all three types of NER subtasks. The results show that our method achieves competitive and even new state-of-the-art performance, and the ablation studies demonstrate the effectiveness of the proposed modules.

References

[1] D. Petkova and W. B. Croft, "Proximity-based document representation for named entity retrieval," in *CIKM*, 2007, pp. 731–740.

[2] D. M. Aliod, M. van Zaanen, and D. Smith, "Named entity recognition for question answering," in *ALTA*, 2006, pp. 51–58.

[3] H. Yan, T. Gui, J. Dai, Q. Guo, Z. Zhang, and X. Qiu, "A unified generative framework for various NER subtasks," in *ACL*, 2021, pp. 5808–5822.

[4] F. Li, Z. Lin, M. Zhang, and D. Ji, "A span-based model for joint overlapped and discontinuous named entity recognition," in *ACL*, 2021.

[5] L. Liu, J. Shang, X. Ren, F. F. Xu, H. Gui, J. Peng, and J. Han, "Empower sequence labeling with task-aware neural language model," in *AAAI*, 2018, pp. 5253–5260.

[6] Y. Cao, Z. Hu, T. Chua, Z. Liu, and H. Ji, "Low-resource name tagging learned with weakly labeled data," in *EMNLP/IJCNLP*, 2019, pp. 261–270.

[7] J. Yu, B. Bohnet, and M. Poesio, "Named entity recognition as dependency parsing," in *ACL*, 2020, pp. 6470–6476.

[8] X. Li, J. Feng, Y. Meng, Q. Han, F. Wu, and J. Li, "A unified MRC framework for named entity recognition," in *ACL*, 2020.

[9] Y. Wang, B. Yu, H. Zhu, T. Liu, N. Yu, and L. Sun, "Discontinuous named entity recognition as maximal clique discovery," in *ACL*, 2021.

[10] B. Wang and W. Lu, "Neural segmental hypergraphs for overlapping mention recognition," in *EMNLP*, 2018, pp. 204–214.

[11] ——, "Combining spans into entities: A neural two-stage approach for recognizing discontiguous entities," in *EMNLP*, 2019, pp. 6215–6223.

[12] A. O. Muis and W. Lu, "Labeling gaps between words: Recognizing overlapping mentions with mention separators," in *EMNLP*, 2017, pp. 2608–2618.

[13] X. Dai, S. Karimi, B. Hachey, and C. Paris, "An effective transition-based model for discontinuous NER," in *ACL*, 2020, pp. 5860–5870.

[14] J. Li, H. Fei, J. Liu, S. Wu, M. Zhang, C. Teng, D. Ji, and F. Li, "Unified named entity recognition as word-word relation classification," in *AAAI*, 2022, pp. 10 965–10 973.

[15] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019, pp. 4171–4186.

[16] B. Tang, J. Hu, X. Wang, and Q. Chen, "Recognizing continuous and discontinuous adverse drug reaction mentions from social media using LSTM-CRF," *Wirel. Commun. Mob. Comput.*, 2018.

[17] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *ICML*, 2001, pp. 282–289.

[18] X. Liu, S. Zhang, F. Wei, and M. Zhou, "Recognizing named entities in tweets," in *ACL*, 2011, pp. 359–367.

[19] M. Ju, M. Miwa, and S. Ananiadou, "A neural layered model for nested named entity recognition," 2018, pp. 1446–1459.

[20] T. Shibuya and E. Hovy, "Nested named entity recognition via second-best sequence learning and decoding," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 605–620, 2020.

[21] M. Xu, H. Jiang, and S. Watcharawittayakul, "A local detection approach for named entity recognition and mention detection," in *ACL*, 2017, pp. 1237–1247.

[22] M. G. Sohrab and M. Miwa, "Deep exhaustive model for nested named entity recognition," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2843–2849.

[23] Y. Luan, D. Wadden, L. He, A. Shah, M. Ostendorf, and H. Hajishirzi, "A general framework for information extraction using dynamic span graphs," in *NAACL-HLT*, 2019, pp. 3036–3046.

[24] D. Wadden, U. Wennberg, Y. Luan, and H. Hajishirzi, "Entity, relation, and event extraction with contextualized span representations," in *EMNLP/IJCNLP*, 2019, pp. 5783–5788.

[25] Y. Shen, X. Ma, Z. Tan, S. Zhang, W. Wang, and W. Lu, "Locate and label: A two-stage identifier for nested named entity recognition," in *ACL/IJCNLP*, 2021, pp. 2782–2794.

[26] W. Lu and D. Roth, "Joint mention extraction and classification with mention hypergraphs," in *EMNLP*, 2015, pp. 857–867.

[27] Y. Wang, B. Yu, Y. Zhang, T. Liu, H. Zhu, and L. Sun, "Tplinker: Single-stage joint extraction of entities and relations through token pair linking," in *COLING*, 2020, pp. 1572–1582.

[28] T. Dozat and C. D. Manning, "Deep biaffine attention for neural dependency parsing," in *ICLR*, 2017.

[29] E. Kiperwasser and Y. Goldberg, "Simple and accurate dependency parsing using bidirectional LSTM feature representations," *Trans. Assoc. Comput. Linguistics*, vol. 4, pp. 313–327, 2016.

[30] J. Su, Y. Lu, S. Pan, B. Wen, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *CoRR*, 2021.

[31] Y. Zhang, Z. Li, and M. Zhang, "Efficient second-order treecrf for neural dependency parsing," in *ACL*, 2020, pp. 3295–3305.

[32] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *ICCV*, 2017, pp. 764–773.

[33] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.

[34] D. B. Johnson, "Finding all the elementary circuits of a directed graph," *SIAM J. Comput.*, vol. 4, no. 1, pp. 77–84, 1975.

[35] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *NAACL-HLT*, 2018, pp. 2227–2237.

[36] J. Straková, M. Straka, and J. Hajic, "Neural architectures for nested ner through linearization," in *ACL*, 2019, pp. 5326–5331.

[37] Z. Tan, Y. Shen, S. Zhang, W. Lu, and Y. Zhuang, "A sequence-to-set network for nested named entity recognition," 2021, pp. 3936–3942.

[38] E. F. T. K. Sang and F. D. Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," in *CoNLL/HLT-NAACL*, 2003, pp. 142–147.

[39] S. Pradhan, A. Moschitti, N. Xue, H. T. Ng, A. Björkelund, O. Uryupina, Y. Zhang, and Z. Zhong, "Towards robust linguistic analysis using ontonotes," in *CoNLL*, 2013, pp. 143–152.

[40] G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. M. Strassel, and R. M. Weischedel, "The automatic content extraction (ACE) program - tasks, data, and evaluation," in *LREC*, 2004.

[41] C. Walker and L. D. Consortium., "Ace 2005 multilingual training corpus." *LDC corpora. Linguistic Data Consortium.*, 2005.

[42] J. Kim, T. Ohta, Y. Tateisi, and J. Tsujii, "GENIA corpus - a semantically annotated corpus for bio-textmining," in *ISMB*, 2003, pp. 180–182.

[43] S. Karimi, A. Metke-Jimenez, M. Kemp, and C. Wang, "Cadec: A corpus of adverse drug event annotations," *J. Biomed. Informatics*, vol. 55, pp. 73–81, 2015.

[44] S. Pradhan, N. Elhadad, B. R. South, D. Martinez, L. M. Christensen, A. Vogel, H. Suominen, W. W. Chapman, and G. K. Savova, "Task 1: Share/clef ehealth evaluation lab 2013." in *CLEF (Working Notes)*, 2013, pp. 212–31.

[45] D. L. Mowery, S. Velupillai, B. R. South, L. Christensen, D. Martinez, L. Kelly, L. Goeuriot, N. Elhadad, S. Pradhan, G. Savova *et al.*, "Task 2: Share/clef ehealth evaluation lab 2014," in *Proceedings of CLEF 2014*, 2014.

[46] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *ICLR*, 2019.

[47] H. Fei, D. Ji, B. Li, Y. Liu, Y. Ren, and F. Li, "Rethinking boundaries: End-to-end recognition of discontinuous mentions with pointer networks," 2021.