T^2-*NER*: A Two-Stage Span-Based Framework for Unified Named Entity Recognition with Templates

Peixin Huang[1], **Xiang Zhao**[1], **Minghao Hu**[2*], **Zhen Tan**[1], **Weidong Xiao**[1]
[1]National University of Defense Technology, Changsha, China
[2]Information Research Center of Military Science, Beijing, China
{huangpeixin15, xiangzhao, tanzhen08a, wdxiao}@nudt.edu.cn,
huminghao16@gmail.com

Abstract

Named Entity Recognition (NER) has so far evolved from the traditional flat NER to overlapped and discontinuous NER. They have mostly been solved separately, with only several exceptions that concurrently tackle three tasks with a single model. The current best-performing method formalizes the unified NER as word-word relation classification, which barely focuses on mention content learning and fails to detect entity mentions comprising a single word. In this paper, we propose a two-stage span-based framework with templates, namely, T^2-*NER*, to resolve the unified NER task. The first stage is to extract entity spans, where flat and overlapped entities can be recognized. The second stage is to classify over all entity span pairs, where discontinuous entities can be recognized. Finally, multi-task learning is used to jointly train two stages. To improve the efficiency of span-based model, we design grouped templates and typed templates for two stages to realize batch computations. We also apply an adjacent packing strategy and a latter packing strategy to model discriminative boundary information and learn better span (pair) representation. Moreover, we introduce the syntax information to enhance our span representation. We perform extensive experiments on eight benchmark datasets for flat, overlapped, and discontinuous NER, where our model beats all the current competitive baselines, obtaining the best performance of unified NER.

1 Introduction

Named entity recognition (NER) is the task of recognizing mentions that represent entities in text. It has been a fundamental task in natural language processing (NLP), due to its wide application in various knowledge-based tasks like entity linking and data mining (Le and Titov, 2018; Cao et al., 2019).

Research on NER has evolved early from flat NER (Sang and Meulder, 2003; Pradhan et al., 2013b), later to overlapped NER (Doddington et al., 2004; Walker et al., 2006), and recently to discontinuous NER (Karimi et al., 2015; Pradhan et al., 2013a). As shown in Figure 1, flat NER simply detects the entity mentions and their types, while the problems of overlapped and discontinuous NER are more complicated, i.e., overlapped entities contain overlapping fragments, and discontinuous entities may contain several non-adjacent fragments. Regarding unified NER, this refers to recognize all types of named entities in the input text, regardless of whether they are flat, overlapping, or discontinuous.

Many methods have been developed to solve three NER tasks (Lu and Roth, 2015; Wang and Lu, 2018; Ju et al., 2018; Wang et al., 2018). The majority of them focus on flat and overlapped NER, with only several exceptions that center on unified NER. Yan et al. (2021) adopt a generative method to obtain position indexes of entity spans. Yet generative models potentially suffer from the exposure bias issue. Li et al. (2022) achieve the current best performance. They use convolution neural networks to obtain two types of word pair relation, and respectively fill them into the upper and lower triangular regions of a word-pair grid for decoding. However, their method decodes entities word by word, without integrally modeling the entity content, which is important for entity classification. It also fails to detect one-word entities as they are only assigned one relation in the grid diagonal. Moreover, it needs to classify over all word pairs, including redundant non-entity ones. Therefore, designing an effective unified NER model is still challenging.

*Corresponding author.

Transactions of the Association for Computational Linguistics, vol. 11, pp. 1265–1282, 2023. https://doi.org/10.1162/tacl_a_00602
Action Editor: Miguel Ballesteros. Submission batch: 3/2023; Revision batch: 6/2023; Published 10/2023.

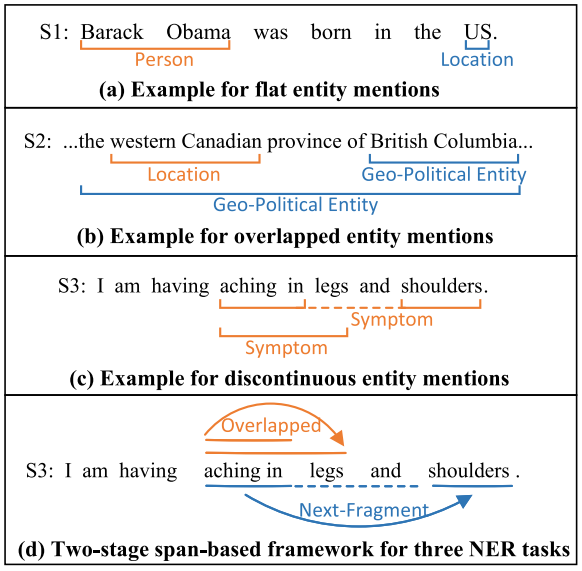

Figure 1: (a)–(c) Examples of three kinds of entity mentions. (d) We propose a two-stage span based framework to solve NER tasks in a unified way.

By contrast, span-based methods (Luan et al., 2019; Sohrab and Miwa, 2018) directly model the span content for entity classification and naturally recognize one-word entities. In light of this, we investigate an alternative unified NER formalism with a two-stage span-based framework. This framework resolves the unified NER by modeling it as span pair relation classification. Such relation is pivotal for recognizing overlapped and discontinuous entities as they describe the semantic relations between entity fragments. To illustrate: In Figure 1(d), in order to recognize the discontinuous entity ‘‘aching in shoulders’’, effectively capturing the discontinuous relation between spans ‘‘aching in’’ and ‘‘shoulders’’ is indispensable.

Specifically, the proposed framework works as follows. In the first stage, Span Extraction classifies the enumerated spans to find all entity spans, which are defined as text spans that either form entity mentions on their own or present as fragments of discontinuous entity mentions. In the second stage, Span Pair Classification classifies entity span pair relation to merge spans into entities. We define two types of relations for this goal: `Next-Fragment` and `Overlapped`, which are used for discontinuous and overlapped mentions respectively, as shown in Figure 1(d). We adopt multi-task learning to jointly train these two stages. This framework naturally solves the problems in Li et al. (2022).

However, it is still a toy framework, as we find three key issues that need to be considered, which may greatly facilitate unified NER.

1. Improving the efficiency: This toy framework needs to classify over all candidate spans and entity span pairs, which inevitably suffers from the inefficiency issue and considerable model complexity.

2. Modeling the discriminative span boundary: Span boundary is essential to discriminate different entity spans. However, this toy framework uses embeddings learned from span content to classify spans, which is inadequate in learning from other spans to acquire discriminative span boundary information.

3. Learning the discriminative span pair representation: Span pair representation directly affects the results of span pair classification. However, this toy framework reuses representations of candidate span pairs from the first stage for relation classification, failing to learn from other pairs to get discriminative pair representation.

Based on the above observations, we introduce the full model T^2-*NER*, which defuses these issues from the following aspects:

1. To speed up the inference process, we are inspired by Zhong and Chen (2021) and propose to equip the toy model with templates, which are packs of markers highlighting the corresponding spans. Templates enable the model to re-use the computations of text tokens and realize an efficient batch computation.

2. To model the discriminative span boundary, we design an adjacent packing strategy in the first stage. This strategy integrally models adjacent spans (i.e., spans with the same start tokens) by packing adjacent markers into a training instance. As the model learns from adjacent spans, more precise span boundary information can be learned.

3. To learn the discriminative span pair representation, we design a latter packing strategy in the second stage. This strategy integrally models the interrelation between span pairs

by packing the former markers with the related latter ones into a training instance. It enables the model to compare same-former spans to learn discriminative span pair representation. Moreover, we utilize syntax information to enhance the span (pair) representation.

Our main contributions are as follows:

- We identify the deficiency of existing NER methods in solving the unified NER task, and propose a new solution *T^2-NER*, to boost the performance. This is done by (1) modeling the unified NER as span pair relation classification, and offering a two-stage span-based framework to realize it; (2) exploiting templates to realize accelerating; and (3) designing adjacent packing strategy and latter packing strategy to get better span (pair) representations.
- We empirically evaluate our proposal on flat, overlapped, and discontinuous NER tasks against 13 baselines. The comparative results demonstrate the superiority of *T^2-NER*.

2 Related Work

Sequential Labeling-based Methods These solve NER tasks through various tagging schemes. These studies usually use neural models such as CNN (Collobert et al., 2011) and Transformer (Yan et al., 2019) for representation, followed by a CRF layer (Lafferty et al., 2001) for classification. However, it is hard for them to directly detect overlapped or discontinuous entity mentions. Shibuya and Hovy (2020) try to decode the tags in a layered manner for overlapped entity mentions. Tang et al. (2018) adopt a BIOHD tagging scheme to resolve the discontinuous NER task. Despite the fact that sequence labeling is reconciled with various NER tasks, it fails to solve these tasks with a unified scheme.

Hypergraph-based Methods These are first introduced into the NER task in Lu and Roth (2015), where they construct hypergraphs by the structure of overlapped mentions and exponentially represent them. Muis and Lu (2016) further explore the application of hypergraphs on discontinuous NER. Wang and Lu (2018) utilize deep neural networks to enhance the hypergraph representation, and decode overlapped entity mentions with hypergraphs. Although hypergraphs can represent all types of entity mentions, they require careful manual design of graph nodes and edges to avoid the structural ambiguity issue. Moreover, these models gradually generate graphs along the words, which may lead to error propagation issue.

Transition-based Methods These are first proposed for nested NER in Wang et al. (2018). They design transition actions, and maintain a stack as well as a buffer to store entities, enabling the representation of nested mentions. Follow-up work includes Dai et al. (2020), which further extends this model for discontinuous NER, through using multiplicative attention to capture the discontinuous dependency. However, these methods need manual intervention for the design of transition actions. And they also face the error propagation issue as transitions are conducted word by word along the sentence.

Generative Methods These adopt generative models such as BART (Lewis et al., 2020) and pointer networks to directly get structured results. Cui et al. (2021) design templates and employ BART for pre-training and fine-tuning on templates to obtain entity types of given spans. Fei et al. (2021) use a generative model with pointer network for discontinuous NER, directly getting a list of entity mentions. Yan et al. (2021) solve the unified NER with BART and pointer network, aiming to generate a sequence of entity start-end indexes and types. Generative methods unavoidably face the decoding efficiency issue as well as the exposure bias issue.

Span-based Methods Recently, NER has been frequently formulated as a span enumeration and classification task. To overcome the enumeration inefficiency issue, Sohrab and Miwa (2018) propose to set the maximum span length. Li et al. (2020) convert NER to a machine reading comprehension (MRC) task and extract entity spans with a MRC model. Shen et al. (2021) design a filter and a regressor to select span proposals. Li et al. (2021) formalize discontinuous NER as a subgraph finding task. Yu et al. (2020) use dependent embeddings as input to a multi-layer BiLSTM and a biaffine model to score spans in a sentence. Although the span-based framework has the innate ability to cope with the overlapped NER, the above methods are subject to enumeration nature, failing to balance the performance and efficiency well.

Other Methods Wang et al. (2021) formulate discontinuous NER as a task of discovering maximal cliques in a segment graph. Li et al. (2022) model the unified NER as word-word relation classification. They achieve the current state-of-the-art (SOTA) performance on the unified NER. However, this work fails to learn the mention content integrally and suffers from the efficiency issue as it needs to classify over all word pairs, including redundant ones.

To sum up, existing work mainly focuses on the flat and overlapped NER, only a few studies seek to resolve the unified NER (Yan et al., 2021; Li et al., 2022). We reconcile the span-based framework to the unified NER with a formalism as span pair relation classification. Our model substantially avoids the drawbacks in previous baselines, by effectively modeling span boundary and learning better span pair representation.

3 Preliminaries

In this section we first introduce the definition of entity span pair relation and the marker. Then we formalize the unified NER problem.

Definition 1 (Entity Span Pair Relation). We define the following three kinds of entity span pair relation and we also give an example as demonstrated in Figure 1(d) for better understanding.

- `OTHER`, indicating that the entity span pair has an other relation or does not have any relation defined in this paper.
- `Next-Fragment`, indicating that two entity spans belong to a single entity mention and they are successive.
- `Overlapped`, indicating that two entity spans are overlapped.

Definition 2 (Span Marker). We explicitly insert a pair of span markers before and after the candidate span to highlight it. We define those of the ith span as <Mi> and </Mi>.

Definition 3 (Span Pair Marker). We explicitly insert two pairs of typed span markers before and after two candidate spans to highlight them. We define these markers as $<\mathrm{F}:e_x>$, $</\mathrm{F}:e_x>$ and $<\mathrm{Li}:e_i>$, $</\mathrm{Li}:e_i>$, where $e_x, e_i \in \mathcal{E}$ and $\mathcal{E}$ denotes a pre-defined entity type set.

Problem 1 (T^2-*NER*: Unified NER as A Two-Stage Span-Based Framework). The unified NER can be formalized as follows: Given an input text of N tokens $X = \{x_1, x_2, \ldots, x_N\}$, the first stage aims to perform span extraction, i.e., representing and classifying the span markers to detect out all entity spans in X of up to length L, e.g., $s_{a,b} = \{x_a, x_{a+1}, ...x_b\}$, as well as its entity type $e \in \mathcal{E}$. After this stage, both the flat and overlapped entity spans could be recognized.

Then, the second stage aims to perform span pair classification by taking $s_{a,b}$ and $s_{c,d}$ as input and utilizing span pair markers to find their relation $r \in \mathcal{R}$, where $\mathcal{R}$ is predefined, including `Next-Fragment`, `Overlapped`, and `OTHER`. Through this stage, the discontinuous entity mentions could be detected. Also, the flat and overlapped entity mentions would be double checked by recognizing the `Overlapped` and `Other` relations between span pairs.

4 T^2-*NER* Model

As shown in Figure 2, T^2-*NER* contains four main components, which are elaborated by the following subsections.

4.1 Span Extraction with Grouped Templates

Span extraction aims to find all text spans and determine whether these spans are entity spans. Different from prior span-based work which simply follows the line of enumerating and classifying (Lee et al., 2017; Luan et al., 2019), we resort to construct templates for each input text to realize an approximate operation, thus accelerating the inference process. In form, these templates are composed of span markers.

Specifically, given an input text with N tokens, $X = \{x_1, x_2, \ldots, x_N\}$, and a maximum span length L, we first enumerate all text spans in X, obtaining a candidate span set as $S(X) = \{s_{(1,1)}, \ldots, s_{(1,L)}, \ldots, s_{(N-L+1,N)}, \ldots, s_{(N,N)}\}$.

Dispersedly training and inferring for all candidate spans requires large computation costs (Luan et al., 2019). To alleviate this issue, we neatly pack these spans into multiple instances at first. To fully utilize the boundary characteristics of spans, we propose an *adjacent packing strategy*. It clusters adjacent spans with the same start tokens in order into a group. For instance,

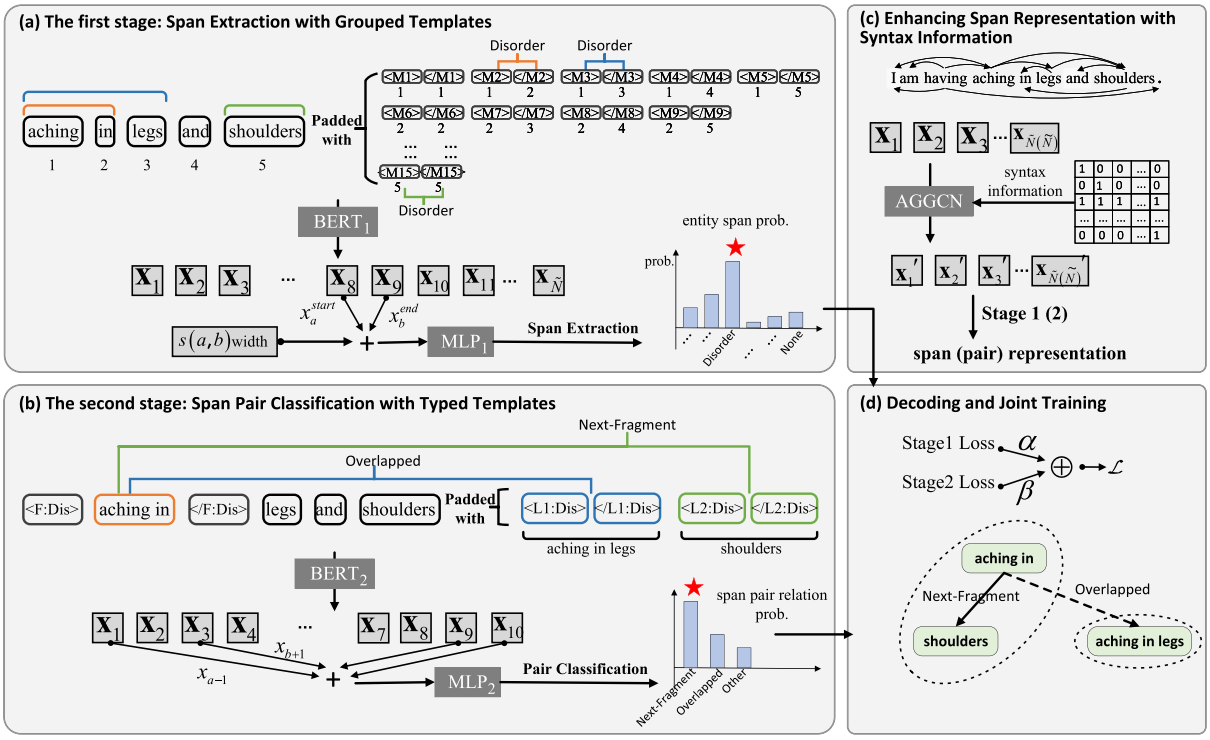

Figure 2: Model overview. (a) The first stage which adopts the adjacent packing strategy to construct templates. (b) The second stage which adopts the latter packing strategy to construct templates. (c) We employ the AGGCN module to encode the syntax information into span (pair) representations. (d) We jointly train two stages and decode the results with `Next-Fragment` relations.

we cluster spans, $\{s_{(1,1)}, s_{(1,2)} \ldots, s_{(1,L)}\}$ into the group S_1. As shown in Figure 2(a), for the sample sentence ''aching in legs and shoulders'' with a maximum span length 5, the group $S_1 = \{aching, aching\ in, aching\ in\ legs, aching\ in\ legs\ and, aching\ in\ legs\ and\ shoulders\}$.

Then we construct a template for each group, which is the sequentially concatenation of marker pairs of all spans in this group. Specifically, for the candidate span s_i, we make the corresponding marker pairs share the same position embeddings with the start and end tokens of this span, i.e., $p(<\text{Mi}>), p(</\text{Mi}>) := p(x_{\text{start}(i)}), p(x_{\text{end}(i)})$. In this way, the position embeddings of original tokens will remain unchanged after the template insertion. As shown in Figure 2(a), the template for group S_1 is <M1></M1><M2></M2><M3></M3><M4></M4><M5></M5>.

Finally, we separately append each template to the input text and feed the sequence into a pretrained BERT module. In order to re-use the representations of text tokens, we harness a directional attention mask matrix in the attention layer (Zhong and Chen, 2021). Specifically, the text tokens only attend to text tokens and not attend to span markers while a span marker can attend to all the text tokens and its partner marker associated with the same span. As a result, we can dispersedly process all groups in multiple runs, and batch spans in each group in one run.

After the BERT calculation, we obtain the representation of the whole sequence, which corresponds to a matrix $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{\tilde{N}}\}$, where $\tilde{N}$ is the length of the appended sequence. We denote contextualized representations of span markers of $s_{(a,b)}$ as $\mathbf{x}_a^{start}$ and $\mathbf{x}_b^{end}$. The span representation is the concatenation of them:

$$\mathbf{h}(s_{(a,b)}) = [\mathbf{x}_a^{start}; \mathbf{x}_b^{end}; \mathbf{w}], \quad (1)$$

where $\mathbf{w}$ is the learned embeddings of span width features and $[;]$ is the concatenation operation.

To recognize all candidate entity spans, we use a MLP layer with ReLU activations for prediction:

$$\mathbf{p}_1(e|s_{(a,b)}) = \text{Softmax}(\text{MLP}_1(\mathbf{h}(s_{(a,b)})), \quad (2)$$

where $\mathbf{p}_1$ denotes the probability distribution of the entity span type $e \in \mathcal{E} \cup \{none\}$, $\mathcal{E}$ denotes a set of predefined entity types and $none$ indicates that the span $\mathbf{s}_{(a,b)}$ is not an entity span.

4.2 Span Pair Classification with Typed Templates

Span pair classification takes candidate span pairs as the input and determine relations for them. Prior work simply re-uses and shares span representations for classifying span pair relations (Luan et al., 2019; Li et al., 2021), failing to learn discriminative representations of different span pairs. Considering this, we propose to compensate by constructing typed templates for each input sequence and learning span pair representations from typed templates.

Similarly, we propose to pack all candidate span pairs into multiple groups. To learn the discriminant span pair representation, we propose a *latter packing strategy*. As shown in Figure 2(b), it clusters span pairs with the same former span into a group. This strategy enables an integral modeling for the same-former spans. Thus, these same-former spans can be compared to obtain discriminative representations.

Specifically, given all of the recognized entity spans in the input text, we sequentially take one of them as the former span, and the others that appear after the former one as the latter spans. Then, we cluster the former span $s_{(a,b)}$ and the corresponding latter spans $\{s_{(c_1,d_1)}, s_{(c_2,d_2)}, \ldots, s_{(c_m,d_m)}\}$ into a group. As shown in Figure 2(b), the former span $aching\ in$ and the latter spans $\{aching\ in\ legs, shoulders\}$ are packed into a group.

We then construct a typed template for each group, which is the sequential concatenation of span pair markers of all pairs in this group. Specifically, for the group $\{s_{(a,b)}, s_{(c_1,d_1)}, s_{(c_2,d_2)}, \ldots, s_{(c_m,d_m)}\}$, we keep the markers $<\mathrm{F}:e_x>$, $</\mathrm{F}:e_x>$ of the former span $s_{(a,b)}$ fixedly located before and after it in the original sentence. We then concatenate marker pairs of the latter spans, i.e., $<\mathrm{L1}:e_1>\ </\mathrm{L1}:e_1>\ <\mathrm{L2}:e_2>\ </\mathrm{L2}:e_2>\ \ldots\ <\mathrm{Lm}:e_m>\ </\mathrm{Lm}:e_3>$. These marker pairs also share the same position embedding with the start and end tokens of corresponding entity spans. The typed template is composed of the fixed markers and the concatenated markers. We argue that these particular templates could capture the dependencies between candidate span pairs, bridging the problem of the prior work that only capture contextual information around each individual candidate span.

Then, we append each typed template to the input text. The whole sequence is denoted as:

$...<\mathrm{F}:e_x>x_a...x_b</\mathrm{F}:e_x>...x_{c_1}...x_{d_1}...x_{c_2}...$
$x_{d_2}...<\mathrm{L1}:e_y></\mathrm{L1}:e_y><\mathrm{L2}:e_z></\mathrm{L2}:e_z>...$

Correspondingly, the appended sequence for the sample sentence in Figure 2(b) is $<\mathrm{F}:Dis>\ aching\ in\ </\mathrm{F}:Dis>\ legs\ and\ shoulders$ $<\mathrm{L1}:Dis></\mathrm{L1}:Dis><\mathrm{L2}:Dis></\mathrm{L2}:Dis>$.

We feed the sequence into another BERT module and we harness a similar attention mask matrix. After the calculation, we obtain the contextualized representation of the whole sequence, which corresponds to a matrix $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{\tilde{N}}\}$, where $\tilde{N}$ is the length of the appended sequence. We denote representations of the markers of $s_{(a,b)}$ as $\mathbf{x}_{a-1}$ and $\mathbf{x}_{b+1}$, and that of the markers of $s_{(c,d)}$ as $\mathbf{x}_c^{start}$ and $\mathbf{x}_d^{end}$. The span pair representation is the concatenation of them:

$$\mathbf{h}(s_{(a,b)}, s_{(c,d)}) = [\mathbf{x}_{a-1}; \mathbf{x}_{b+1}; \mathbf{x}_c^{start}; \mathbf{x}_d^{end}], \quad (3)$$

where $[;]$ denotes the concatenation operation.

Finally, we harness another MLP with ReLU activations for prediction:

$$\mathbf{p}_2(r|X) = \mathrm{Softmax}(\mathrm{MLP}_2(\mathbf{h}(s_{(a,b)}, s_{(c,d)})), \quad (4)$$

where $\mathbf{p}_2$ is probability distribution of span pair relation type $r \in \mathcal{R}$, and $\mathcal{R}$ is the relation type set.

Moreover, we introduce a reverse setting from the latter to the former for a bidirectional prediction. For the candidate span pair $s_{(a,b)}, s_{(c,d)}$, we follow the above operation, obtaining former marker representations $\mathbf{x}_a^{start}$ and $\mathbf{x}_b^{end}$, and latter marker representations $\mathbf{x}_{c-1}$ and $\mathbf{x}_{d+1}$. These embeddings are added in (3) to get the final span pair representation. We argue that this reverse setting would also provide supplemental information with integral modeling of the same-latter spans.

4.3 Enhancing Span Representation with Syntax Information

Dependency syntax information is commonly neglected in the unified NER work. It has been explored in flat NER (Finkel and Manning, 2009).

In this paper, we use it to enhance our model. Specifically, we harness a dependency parser to transform the appended sequence into an adjacency matrix $\mathbf{A}$, where $\mathbf{A}_{ij} = 1$ indicates that there is a dependency edge going from token x_i to token x_j, otherwise $\mathbf{A}_{ij} = 0$. Notably, each marker token shares the same syntax information with the corresponding text token.

We then harness an attention-guided GCN (AGGCN) (Guo et al., 2019) to encode the syntax graph. To illustrate the AGGCN module, we start with the traditional GCN (Kipf and Welling, 2017). Given the representations of the appended text $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{\tilde{N}(\tilde{N})}\}$, where $\tilde{N}(\tilde{N})$ is the length of the appended text, the GCN module updates them with syntax information as follows:

$$\mathbf{x}_i^{(l)} = \sigma(\sum_{j=1}^{\tilde{N}(\tilde{N})} \mathbf{A}_{ij}\mathbf{W}^{(l)}\mathbf{x}_j^{(l-1)} + \mathbf{b}^{(l)}), \quad (5)$$

where $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the weight matrix and bias vector of the lth layer.

AGGCN improves the original GCN by transforming the adjacency matrix $\mathbf{A}$ into an attention guided adjacency matrix $\widetilde{\mathbf{A}}$, where $\widetilde{\mathbf{A}}_{ij}$ is the weight of the edge going from token x_i to the token x_j. $\widetilde{\mathbf{A}}$ is computed through multi-head self-attention (Vaswani et al., 2017):

$$\widetilde{\mathbf{A}}^t = \text{Softmax}(\frac{\mathbf{X}^t\mathbf{W}_Q^t \times (\mathbf{X}^t\mathbf{W}_K^t)^T}{\sqrt{d}}), \quad (6)$$

where $\mathbf{W}_Q^t$ and $\mathbf{W}_K^t$ are the matrices which project $\mathbf{X}^t$ to the query and the key, d is the input feature dimension. $\widetilde{\mathbf{A}}^t$ is the attention guided adjacency matrix of the tth head and $t \leq N_{head}$ where N_{head} is a hyper-parameter.

Then AGGCN adopts densely connected layers to update $\mathbf{X}$ with $\widetilde{\mathbf{A}}$, following the similar operation in (5). The output is $\widetilde{\mathbf{X}}^t$. Then the information from different layer is integrated:

$$\widetilde{\mathbf{X}} = [\widetilde{\mathbf{X}}^1, \widetilde{\mathbf{X}}^2, \ldots, \widetilde{\mathbf{X}}^{N_{head}}]\mathbf{W}_1, \quad (7)$$

where $\mathbf{W}_1$ is the weight. $\widetilde{\mathbf{X}}$ is then concatenated with the original token representations to obtain the final token representations $\mathbf{X}' = [\mathbf{X}, \widetilde{\mathbf{X}}\mathbf{W}_2]$, where $\mathbf{W}_2$ denotes a linear transformation for dimension reduction. The final syntax-enhanced representations $\mathbf{X}'$ would be used in (1) and (3) for the following span (pair) representation.

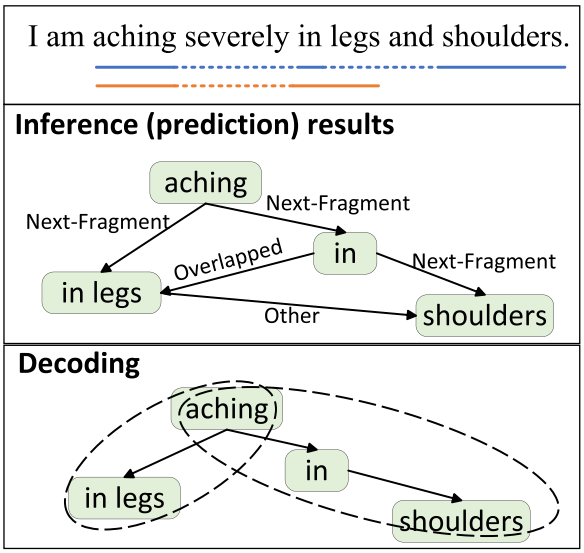

Figure 3: Inference and decoding cases of the sample sentence. In this sentence, ''aching in shoulders'' is made of three spans.

4.4 Joint Training and Decoding

We jointly train two stages through multi-task learning:

$$\mathcal{L} = -\sum \alpha \log \mathbf{p}_1(e^*) + \beta \mathbf{p}_2(r^*), \quad (8)$$

where e^* and r^* denote the corresponding gold-standard labels for entity spans and span pairs, α and β are the hyper-parameters to control the importance of two stages.

In the training process, three types of relations are all needed. In the inference process, the flat and overlapped entities would be recognized in the first stage, and the discontinuous entities would be recognized through `Next-Fragment` relation in the second stage. Thus, we choose `Next-Fragment` relation for decoding.

The predictions of our model are the entity spans and their relations, which can be considered as a directional span graph. The decoding object is to find sub-graphs in which each entity span connects with any other span by the `Next-Fragment` relation. Each sub-graph corresponds to an entity mention and the entities which are made of more than two spans are also covered in the decoding process. The sub-graph containing a single entity span composes an entity mention by itself. Figure 3 gives the cases for the inference and decoding of a sample sentence.

Datasets	Sentences				Entity Mention			
	# Train	# Dev	# Test	# Avg.Len	# All	# Ovlp.	# Dis.	# Avg.Len
CoNLL2003	17291	–	3453	14.38	35089	–	–	1.45
OntoNotes5.0	59924	8528	8262	18.11	104151	–	–	1.83
ACE2004	6802	813	897	20.12	27604	12626	–	2.50
ACE2005	7606	1002	1089	17.77	30711	12404	–	2.28
GENIA	15023	1669	1854	25.41	56015	10263	–	1.97
CADEC	5340	1097	1160	16.18	6316	920	679	2.72
ShARe13	8508	1250	9009	14.86	11148	663	1088	1.82
ShARe14	17404	1360	15850	15.06	19070	1058	1656	1.74

Table 1: Statistics of NER datasets. ''#'' denotes the amount. ''Ovlp.'' and ''Dis.'' denote overlapped and discontinuous mentions, respectively.

5 Experimental Setting

5.1 Datasets and Evaluations

To evaluate T^2-*NER* for three NER tasks, we experiment on eight benchmark datasets. Statistics of these datasets are presented in Table 1.

- *Flat NER Datasets*, include CoNLL2003 (Sang and Meulder, 2003) and OntoNotes 5.0 (Pradhan et al., 2013b). CoNLL2003 is an English dataset with four types of flat entities. We follow the data processing in Lin et al. (2019). For OntoNotes 5.0, we use the same dataset settings as Yan et al. (2021).
- *Overlapped NER Datasets*, include ACE2004[1] (Doddington et al., 2004), ACE2005[2] (Walker et al., 2006), and GENIA (Kim et al., 2003). ACE2004 and ACE2005 are derived from various domains, such as newswire and online forums. We follow Lu and Roth (2015), splitting the train/dev/test as 8:1:1. For GENIA, we follow Yan et al. (2021), collapsing all entity subtypes into five types and splitting the train/dev/test as 8.1:0.9:1.
- *Discontinuous NER Datasets*, include CADEC (Karimi et al., 2015), ShARe13 (Pradhan et al., 2013a), and ShARe14 (Mowery et al., 2014), all of which are collected from biomedical or clinical domain. We use the same data processing as Dai et al. (2020).

We use strict evaluations that a predicted entity is counted as true positive mention if both its span and type match those of a gold entity. As for a discontinuous entity, each span should match a span of the gold entity. We use span-level micro-averaged *Precision* (*P*), *Recall* (*R*), and *F1 score* (*F1*) as evaluation metrics.

5.2 Implementation Details

Considering the dataset domains, we use BioBERT (Lee et al., 2020) for GENIA and CADEC, ClinicalBERT (Alsentzer et al., 2019) for ShARe13 and ShARe14, and vanilla BERT (Devlin et al., 2019) for the other datasets. To obtain the dependency syntax information, we harness the Stanford CoreNLP parser 4.5.4 (Manning et al., 2014), which is based on Shift-Reduce parsing neural model (Zhu et al., 2013). This parser is trained with a collection of syntactically annotated data, i.e., the Penn Treebank corpus.[3] We directly apply it without using additional training datasets to train this parser. The parser training data do not overlap with the evaluation datasets. Thus, there is no data contamination issue in our work.

We incorporate cross-sentence information by expanding the input to a fixed window size with its context and ensuring that each text is located in the middle of the expanded sequence. In practice, we use grid search to find the best experimental configuration. Concretely, the window size is chosen from [128, 256, 384], the size of span width feature $\mathbf{w}$ from [150, 200], the loss weight α and β from [0.6, 0.8, 1.0], the MLP layer from [1, 2, 3]. As for the AGGCN part, the GCN layer l is chosen from [1, 2], the GCN head N_{head} from [2, 4]. Balancing effectiveness and efficiency, we choose the following configurations to produce the experimental results reported in the following sections. The window size is set to 256 for both stages, the span width feature size $\mathbf{w}$ to 150, the loss weight α and β to 1.0, the MLP layer to 2, the MLP size to 150, the GCN layer l to 2 and the GCN head N_{head} to 4. These hyperparameters are the same for all datasets. In the enumeration of the span, we set the maximum span length L as 16 for OntoNotes5.0 and GENIA, and 8 for the other datasets.

[1] https://catalog.ldc.upenn.edu/LDC2005T09.
[2] https://catalog.ldc.upenn.edu/LDC2006T06.
[3] https://catalog.ldc.upenn.edu/docs/LDC99T42.

During the training process, we adopt the Adam Weight Decay optimizer (Loshchilov and Hutter, 2019) with a learning rate of 2e-5 to finetune BERT and 1e-3 to finetune other parts of the model. For the batch size, we search from [4,8,16,32] and choose 8 for all datasets. For the training epochs, we search from [3,5,8,10,15,50], and finally choose 8 for CoNLL2003, 5 for OntoNotes5.0, 15 for ACE2004, 10 for ACE2005, 50 for GENIA and CADEC, and 10 for ShARe13 and ShARe14. We run each experiment for 5 times and report the averaged score.

5.3 Baselines

Sequential Labeling-based Methods assign a tag to each token with various tagging schemes, including *Seq2Seq* (Straková et al., 2019) and *Second-Path* (Shibuya and Hovy, 2020). **Hypergraph-based methods** construct hypergraphs to represent and extract entities, including *Seg-Graph* (Wang and Lu, 2018) and *Two-Stage* (Wang and Lu, 2019). **Transition-based methods** maintain a stack and a buffer to represent and infer entity mentions, including *Transition* (Dai et al., 2020) and *ARN* (Lin et al., 2019). **Generative methods** generate entity index or word sequences with the decoder, including *BART-Large* (Yan et al., 2021) and *MAPtr* (Fei et al., 2021). **Span-based methods** enumerate spans and combine them into entities, including *BERT-MRC* (Li et al., 2020), *Locate-Label* (Shen et al., 2021), and *Extract-Select* (Huang et al., 2022). **Other methods** include *MAC* (Wang et al., 2021) and W^2*NER* (Li et al., 2022) approaches.

We directly adopt the best parameter setup reported in papers that originally introduced the methods listed. The best results on each dataset are denoted in **bold**. We use the two-tailed t-test to measure the statistical significance. Significant improvements of T^2*-NER* over the second best model for $p < 0.05$ are marked with ★.

6 Results and Analyses

6.1 Results for Flat NER

Table 2 presents the experimental results on two flat NER datasets. As seen: (1) T^2*-NER* achieves the SOTA performance with at least 1.06% and 0.87% *F1 score* improvements on CoNLL2003 and OntoNotes 5.0 datasets, respectively. (2)

	CoNLL2003			OntoNotes 5.0		
Model	*P*	*R*	*F1*	*P*	*R*	*F1*
Seq2Seq (Straková et al., 2019)	–	–	92.98	–	–	–
Seg-Graph (Wang and Lu, 2018)	–	–	90.50	–	–	–
BART-Large (Yan et al., 2021)	92.56	93.56	93.05	89.62	90.92	90.27
BERT-MRC (Li et al., 2020)	92.33	**94.61**	93.04	**92.98**	89.95	91.11
Locate-Label (Shen et al., 2021)	92.13	93.73	92.94	–	–	–
Extract-Select (Huang et al., 2022)	92.10	94.03	93.05	–	–	–
W^2NER (Li et al., 2022)	92.71	93.44	93.07	90.03	90.97	90.50
T^2*-NER*	**93.78**	94.48	**94.13**★	91.83	**92.13**	**91.98**★

Table 2: Results on the flat NER datasets (%).

T^2*-NER* outperforms other unified NER models (i.e., *BART-Large* and W^2*NER*) for the flat NER task. We attribute this to the two-stage architecture, which further double-checks the flat NER results in the second stage. (3) T^2*-NER* outperforms other span-based models (i.e., *BERT-MRC*, *Locate-Label* and *Extract-Select*). The reason may be that we introduce the syntax information to enhance the span representation, and our templates are more advantageous in learning span-wise representation for span extraction.

6.2 Results for Overlapped NER

Table 3 presents the result comparisons on three overlapped NER datasets. As seen: (1) T^2*-NER* can effectively deal with the nested NER task, achieving the SOTA performances with 1.01%, 2.58%, and 0.43% *F1 score* improvements on ACE2004, ACE52005, and GENIA datasets, respectively. (2) Different from other unified NER models, T^2*-NER* achieves higher *Precision* while relatively lower *Recall*, since it double-checks the entity spans with ''Overlapped'' relation and further filter spans that it believes to be low confident. In contrast, the results of T^2*-NER* on the flat NER task do not show a similar trend. We argue that this occurs because the second stage does not conduct particular classification of flat entities. (3) Moreover, the superiority of T^2*-NER* over other span-based models also proves the effectiveness of T^2*-NER* in obtaining better span representations, which we attribute to the *adjacent packing strategy* and the utilization of syntax information.

6.3 Results for Discontinuous NER

Table 4 shows the performance comparison on three discontinuous NER datasets. As seen, T^2*-NER* outperforms the previous best model W^2*NER* by 2.48%, 1.28%, and 0.37% in *F1 score*

Model	ACE2004			ACE2005			GENIA		
	P	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*
Seq2Seq (Straková et al., 2019)	–	–	84.33	–	–	83.42	–	–	78.20
Second-Path (Shibuya and Hovy, 2020)	83.73	81.91	82.81	82.98	82.42	82.70	78.07	76.45	77.25
Seg-Graph (Wang and Lu, 2018)	78.00	72.40	75.10	76.80	72.30	74.50	77.00	73.30	75.10
ARN (Lin et al., 2019)	–	–	–	76.20	73.60	74.90	75.80	73.90	74.80
BART-Large (Yan et al., 2021)	87.27	86.41	86.84	83.16	86.38	84.74	78.87	79.60	79.23
BERT-MRC (Li et al., 2020)	85.05	86.32	85.98	87.16	86.59	86.88	**85.18**	81.12	83.75
Locate-Label (Shen et al., 2021)	87.44	87.38	87.41	86.09	87.27	86.67	80.19	80.89	80.54
W^2*NER* (Li et al., 2022)	87.33	87.71	87.52	85.03	88.62	86.79	83.10	79.76	81.39
Extract-Select (Huang et al., 2022)	88.26	**88.53**	88.39	87.15	88.37	87.76	83.64	**84.41**	84.02
T^2*-NER*	**90.88**	87.97	**89.40***	**90.53**	**90.15**	**90.34***	85.09	83.82	**84.45***

Table 3: Results on the overlapped NER datasets (%). For a fair comparison, we only present results of *Second-Path* with the BERT model.

Model	CADEC			ShARe13			ShARe14		
	P	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*
Two-Stage (Wang and Lu, 2019)	72.10	48.40	58.00	83.80	60.40	70.30	79.10	70.70	74.70
Transition (Dai et al., 2020)	68.90	69.00	69.00	80.50	75.00	77.70	78.10	81.20	79.60
BART-Large (Yan et al., 2021)	70.08	71.21	70.64	82.09	77.42	79.69	77.20	83.75	80.34
MAPtr (Fei et al., 2021)	75.50	71.80	72.40	**87.90**	77.20	80.30	–	–	–
MAC (Wang et al., 2021)	70.50	72.50	71.50	84.30	78.20	81.20	78.20	**84.70**	81.30
W^2*NER* (Li et al., 2022)	74.09	72.35	73.21	85.57	79.68	82.52	79.88	83.71	81.75
T^2*-NER*	**78.33**	**73.22**	**75.69***	87.41	**80.48**	**83.80***	**80.53**	83.77	**82.12***

Table 4: Results on the discontinuous NER datasets (%).

on CADEC, ShARe13, and ShARe14, respectively, obtaining new SOTA results. Although some methods outperform *T*^{2}*-NER* for *Precision* or *Recall*, they sacrifice another score, which results in lower *F1 score*. This observation can also be found in the flat and overlapped NER results.

Recall that three discontinuous NER datasets also contain flat and overlapped entities. Only around 10% of entity mentions in these datasets are discontinuous. To truly understand how *T*^{2}*-NER* behaves on data with only discontinuous entities, we follow Dai et al. (2020) and experiment on a subset of the test set where only sentences with at least one discontinuous entity mention are included (Figure 4(a)). Since sentences in this subset sometimes contain flat and overlapped mentions as well, we further report the test results when only discontinuous entity mentions are considered (Figure 4(b)). We compare with several discontinuous NER models and report the results on the subset of three datasets in Figure 4. We can see that our model can predict the discontinuous entity mentions and consistently defeat the baseline models in both settings.

6.4 Model Ablation Study

To elucidate the contribution of the main components, we design five internal baselines for comparison. We only report the results on CoNLL2003, ACE2005, and CADEC datasets as the findings on the other datasets are qualitatively similar.

- *w/o. Adjacent Packing*: This variation removes *adjacent packing strategy* and orderly clusters spans into multiple groups of equal size K. For instance, the group S_1 would be $\{s_{(1,1)}, s_{(1,2)} \ldots, s_{(\lceil \frac{K}{L} \rceil, K-\lfloor \frac{K-1}{L} \rfloor * L)}\}$. In practice, we set K to 128. Accordingly, the grouped templates would also be different.

- *w/o. Latter Packing*: This variation leaves out *latter packing strategy* and clusters all of

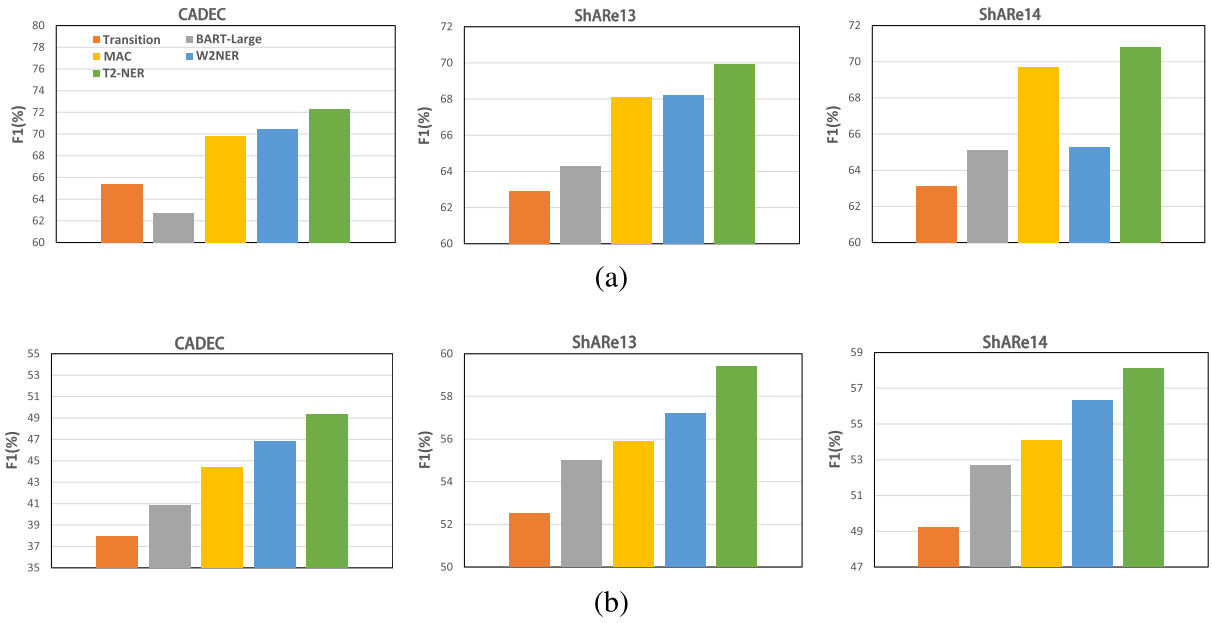

Figure 4: Results of discontinuous entity mentions. (a) The *F1 score* on sentences that include at least one discontinuous entity mention. (b) The *F1 score* only considering discontinuous entity mentions.

the span pairs into multiple groups. Specifically, we batch candidate pairs by appending 4 markers for each pair to the end of the sentence, until the total number of tokens exceeds 250. Accordingly, the appended markers form one template.

- *w/o. Reverse Setting*: This variation adopts the uni-directional prediction, leaving out the reverse span pair representation and only using equation (3) for span pair classification.
- *w/o. Syntax Information*: This variation removes syntax graph-guided AGGCN module. The syntax-enhanced token representation is removed accordingly, without influencing the major goal.
- *w. Untyped Span Pair Marker*: This variation apply untyped span pair markers to construct templates in the second stage.

Results are shown in Table 5. As seen: (1) T^2*-NER* greatly or comparably outperforms five internal baselines on the test set of three datasets. Compared to T^2*-NER*, *w/o. Adjacent Packing* drops 0.02%, 0.52%, and 0.94% *F1 score* on CoNLL2003, ACE2005, and CADEC. The results demonstrate that it is sub-optimal to simply pack spans equally into multiple groups. The reason may be that the modeling of these groups

Model	CoNLL2003	ACE2005	CADEC
w/o. Adjacent Packing	94.11(−0.02)	89.82(−0.52)	74.75(−0.94)
w/o. Latter Packing	94.12(−0.01)	88.36(−1.98)	74.28(−1.41)
w/o. Reverse Setting	94.08(−0.05)	90.17(−0.17)	75.37(−0.32)
w/o. Syntax Information	94.00(−0.13)	89.84(−0.50)	75.65(−0.04)
w. Untyped Span Pair Marker	94.13(0)	90.14(−0.20)	75.36(−0.33)
T^2*-NER*	**94.13**	**90.34**	**75.69**

Table 5: F1 scores of internal baselines (%).

does not produce meaningful information, whereas the adjacent packing strategy can learn discriminative boundary information through the integral modeling of same-start-token spans. (2) *w/o. Latter Packing* shows huge performance drop on ACE2005 and CADEC. This result demonstrates that the integral modeling of same-former spans contributes to discriminative representations of span pairs, impelling better `Next-Fragment` and `Overlapped` relation classification. Moreover, *w/o. Latter Packing* suffers from a slight *F1 score* decrease on CoNLL2003, which reveals that the double check of `Other` relation in the second stage is also effective. (3) When the reverse setting is removed, the *F1 scores* on all datasets decrease, indicating the significance of modeling the information from the latter span to the former span. (4) Experimental results of *w/o. Syntax Information* and *w. Untyped Span Pair Marker* demonstrate that, after removing

Model	*P*	*R*	*F1*
Span Extraction	90.85	86.57	88.66
Span Extraction (+*Span Pair Classification*)	**91.10**	**87.09**	**89.05**

Table 6: Effect of joint training of two stages. *P*, *R* and *F1* are the results of span extraction stage (%).

Model	*F1*	*Speed* (Sentence/s)
Transition (Dai et al., 2020)	69.00	66.5
BART-Large (Yan et al., 2021)	70.64	19.2
MAC (Wang et al., 2021)	71.50	109.7
W^2NER (Li et al., 2022)	73.21	**365.7**
T^2-*NER*	**75.69**	190.3

Table 7: Running speed comparisons.

either AGGCN module or entity type information, the *F1 scores* go down. This observation suggests that syntax information and entity type information are both effective for our model. (5) After removing the Syntax information, the performance drop on CoNLL2003 and ACE2005 datasets is more significant compared to CADEC. The reason may be that the parser performs better in news domain than in biomedical domain, as it is trained over the news text.

6.5 Analysis of Joint Training

To explore the influence of jointly training two stages, we present the performance changes of the first stage (i.e., Span Extraction) before and after adding the second stage (i.e., Span Pair Classification). The performance comparisons on CADEC are shown in Table 6. We find that the *F1 scores* of span extraction increases by 0.39% after adding the span pair classification task. This observation shows that the second stage could benefit the first stage, revealing that joint training is more advisable. Moreover, the double-checking in the second stage may benefit the extraction of flat and overlapped spans in the first stage.

6.6 Analysis of Complexity

Along with the significant performance improvements, pre-trained language models (e.g., BERT) usually face the issue of high computational costs. Even worse, this issue become more severe as the sequence length continues to increase. In this paper, we insert markers and concatenate them with the original texts. It is obvious that these markers extend the length of input text.

For both stages, we group these markers into several batches, which can control the length of appended sequences. For the first stage, we enumerate spans in a small-length text and then use its contexts to expand the text to 256 tokens, for which the number of candidate spans in a text is usually less than the context length. Hence, with grouped templates consisting of a small number of span markers, the complexity of T^2-*NER* is still near-linearly to that of the model without templates. For the second stage, after filtering non-entity text spans in the first stage, the number of candidate spans is relatively small, thus the increased computation is limited.

We further present the comparison of inference speed between three baselines and T^2-*NER* in Table 7. For fair comparison, all of these models are implemented by PyTorch and ran on a NVIDIA RTX 3090 GPU environment. As we can see, the inference speeds of T^2-*NER* are around 3 times faster than *Transition* and 10 times faster than *BART-Large*. By contrast, our model achieves a 2.48% F1 improvement on CADEC but sacrifices 48% speed compared to the W^2-*NER* model. Considering the effectiveness and efficiency of our model, we expect it to be effective in practice.

6.7 Case Study

We find that most mistakes are caused by incorrectly recalling some entities. As shown in Table 8, the span ‘‘laceration in esophagus’’ in case 3 is incorrectly recognized as an entity. Our model may be confused by the preposition ‘‘in’’, and tend to classify phrases such as ‘‘blood in stomach’’ and ‘‘laceration in esophagus’’ as entities. Nonetheless, three cases illustrate that T^2-*NER* can recognize the flat, overlapped and discontinuous entities. From case 1, we find that three overlapped entities from inside to outside are all accurately recognized, revealing that T^2-*NER* can recognize multi-level overlapped entities. From case 2, we find that T^2-*NER* correctly recognizes the reference phrase ‘‘both side’’, whereas W^2*NER* incorrectly recognizes it as a PER entity, revealing that T^2-*NER* can resolve ambiguous entity reference.

Sentence 1	$^{3}[^{2}[^{1}$[United nations]$^{1}_{\text{ORG}}$ secretary general]$^{2}_{\text{PER}}$ kofi annan]$^{3}_{\text{PER}}$ today discussed plans for the summit with ^{4}[the host]$^{4}_{\text{PER}}$, $^{7}[^{6}[^{5}$[egyptian]$^{5}_{\text{GPE}}$ president]$^{6}_{\text{PER}}$ hosni mubarak]$^{7}_{\text{PER}}$.
W²NER	$^{3}[^{2}[^{1}$[United nations]$^{1}_{\text{ORG}}$ ^{4}[secretary general]$^{2}_{\text{PER}}$ kofi annan]$^{3}_{\text{PER}}]^{4}_{\text{PER}}$ today discussed plans for the summit with ^{5}[the host]$^{5}_{\text{PER}}$, $^{7}[^{6}[$ egyptian president]$^{6}_{\text{PER}}$ hosni mubarak]$^{7}_{\text{PER}}$.
T²-NER	$^{3}[^{2}[^{1}$[United nations]$^{1}_{\text{ORG}}$ secretary general]$^{2}_{\text{PER}}$ kofi annan]$^{3}_{\text{PER}}$ today discussed plans for the summit with ^{4}[the host]$^{4}_{\text{PER}}$, $^{7}[^{6}[^{5}$[egyptian]$^{5}_{\text{GPE}}$ president]$^{6}_{\text{PER}}$ hosni mubarak]$^{7}_{\text{PER}}$.
Sentence 2	^{1}[The US Supreme Court]$^{1}_{\text{ORG}}$ will hear arguments from ^{2}[both sides]$^{2}_{\text{ORG}}$ on Friday and $^{4}[^{3}$[Florida]$^{3}_{\text{GPE}}$ ' s ^{5}[Leon County]$^{5}_{\text{GPE}}$ Circuit Court]$^{4}_{\text{ORG}}$ will consider the arguments on disputed ^{6}[state]$^{6}_{\text{GPE}}$ ballots on Saturday .
W²NER	^{1}[The US Supreme Court]$^{1}_{\text{ORG}}$ will hear arguments from ^{2}[both sides] PER on Friday and $^{3}[$ Florida ' s ^{4}[Leon County]$^{4}_{\text{GPE}}$ Circuit Court]$^{3}_{\text{ORG}}$ will consider the arguments on disputed state ballots on Saturday .
T²-NER	^{1}[The US Supreme Court]$^{1}_{\text{ORG}}$ will hear arguments from ^{2}[both sides]$^{2}_{\text{ORG}}$ on Friday and $^{4}[^{3}$[Florida]$^{3}_{\text{GPE}}$ ' s ^{5}[Leon County]$^{5}_{\text{GPE}}$ Circuit Court]$^{4}_{\text{ORG}}$ will consider the arguments on disputed ^{6}[state]$^{6}_{\text{GPE}}$ ballots on Saturday .
Sentence 3	EGD showed ^{1}[hiatal hernia]$^{1}_{\text{Disorder}}$ and vertical ^{2}[laceration]$^{2}_{\text{Disorder}}$ in distal ^{2}[esophagus]$^{2}_{\text{Disorder}}$ with ^{3}[blood in ^{4}[stomach]$^{4}_{\text{Disorder}}]^{3}_{\text{Disorder}}$ and overlying ^{4}[lac]$^{4}_{\text{Disorder}}$.
W²NER	EGD showed ^{1}[hiatal hernia]$^{1}_{\text{Disorder}}$ and vertical ^{2}[laceration]$^{2}_{\text{Disorder}}$ in distal ^{2}[esophagus]$^{2}_{\text{Disorder}}$ with ^{3}[blood in ^{4}[stomach]$^{4}_{\text{Disorder}}]^{3}_{\text{Disorder}}$ and overlying ^{4}[lac]$^{4}_{\text{Disorder}}$.
T²-NER	EGD showed ^{1}[hiatal hernia]$^{1}_{\text{Disorder}}$ and vertical $^{3}[^{2}$[laceration]$^{2}_{\text{Disorder}}$ in]$^{3}_{\text{Disorder}}$ distal $^{3}[^{2}$[esophagus]$^{2}_{\text{Disorder}}]^{3}_{\text{Disorder}}$ with ^{4}[blood in ^{5}[stomach]$^{5}_{\text{Disorder}}]^{4}_{\text{Disorder}}$ and overlying ^{5}[lac]$^{5}_{\text{Disorder}}$.

Table 8: Examples of predicted results of our model and W^2NER. Words annotated with the same index are part of the same entity, subscripts indicate entity types, red brackets indicate golden entities, blue brackets indicate entities predicted by the models, red highlights indicate wrong predictions of models, blue highlights indicate entities that models fail to recognize.

7 Conclusion

In this paper, we introduce a two-stage span-based framework with templates, named T^2-*NER* for the unified NER task. T^2-*NER* formulate the NER task as span pair relation classification, thus naturally tackling the overlapped and discontinuous NER. Thanks to this formulation, T^2-*NER* is quite effective for various NER tasks, achieving the SOTA performances on eight benchmark datasets. As future work, T^2-*NER* will be improved through discovering more advanced span features and knowledge derived from external sources such as Wikipeidia.

Acknowledgments

We would like to thank our action editor, Miguel Ballesteros, and the anonymous reviewers for their invaluable suggestions and feedback. This work is partially supported by National Key R&D Program of China Nos. 2022YFB3102600, NSFC under grants Nos. 62002373, 62006243, 62272469, and 71971212, and the Science and Technology Innovation Program of Hunan Province under grant No. 2020RC4046.

References

Emily Alsentzer, John R. Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew B. A. McDermott. 2019. Publicly available clinical BERT embeddings. *CoRR*, abs/1904.03323v3. https://doi.org/10.18653/v1/W19-1909

Yixin Cao, Zikun Hu, Tat-Seng Chua, Zhiyuan Liu, and Heng Ji. 2019. Low-resource name tagging learned with weakly labeled data. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 261–270, Hong Kong, China. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1025

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using BART. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1835–1845, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.findings-acl.161

Xiang Dai, Sarvnaz Karimi, Ben Hachey, and Cécile Paris. 2020. An effective transition-based model for discontinuous NER. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5860–5870, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.520

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

George R. Doddington, Alexis Mitchell, Mark A. Przybocki, Lance A. Ramshaw, Stephanie M. Strassel, and Ralph M. Weischedel. 2004. The automatic content extraction (ACE) program - Tasks, data, and evaluation. In *Proceedings of the Language Resources and Evaluation Conference, May. 2004*.

Hao Fei, Donghong Ji, Bobo Li, Yijiang Liu, Yafeng Ren, and Fei Li. 2021. Rethinking boundaries: End-to-end recognition of discontinuous mentions with pointer networks. In *Proceedings of the AAAI Conference February. 2021*, pages 12785–12793. https://doi.org/10.1609/aaai.v35i14.17513

Jenny Rose Finkel and Christopher D. Manning. 2009. Nested named entity recognition. In *Proceedings of the EMNLP Conference August. 2009*, pages 141–150. https://doi.org/10.3115/1699510.1699529

Zhijiang Guo, Yan Zhang, and Wei Lu. 2019. Attention guided graph convolutional networks for relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 241–251, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1024

Peixin Huang, Xiang Zhao, Minghao Hu, Yang Fang, Xinyi Li, and Weidong Xiao. 2022. Extract-select: A span selection framework for nested named entity recognition with generative adversarial training. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 85–96, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.findings-acl.9

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A neural layered model for nested named entity recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1446–1459. New Orleans, Louisiana. Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1131

Sarvnaz Karimi, Alejandro Metke-Jimenez, Madonna Kemp, and Chen Wang. 2015. CADEC: A corpus of adverse drug event annotations. *Journal of Biomedical Informatics*, 55:73–81. https://doi.org/10.1016/j.jbi.2015.03.010, PubMed: 25817970

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA corpus—A semantically annotated corpus for bio-textmining. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology, June 29–July 3. 2003*, pages 180–182. https://doi.org/10.1093/bioinformatics/btg1023, PubMed: 12855455

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th ICLR Conference April. 2017*. OpenReview.net.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings*

of the ICML Conference June 28 – July 1. 2001, pages 282–289.

Phong Le and Ivan Titov. 2018. Improving entity linking by modeling latent relations between mentions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1595–1604, Melbourne, Australia. Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1148

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240. https://doi.org/10.1093/bioinformatics/btz682, PubMed: 31501885

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics. https://doi.org/10.18653/v1/D17-1018

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.703

Fei Li, Zhichao Lin, Meishan Zhang, and Donghong Ji. 2021. A span-based model for joint overlapped and discontinuous 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)*, pages 4814–4828, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.372

Jingye Li, Hao Fei, Jiang Liu, Shengqiong Wu, Meishan Zhang, Chong Teng, Donghong Ji, and Fei Li. 2022. Unified named entity recognition as word-word relation classification. In *Proceedings of the AAAI Conference February 22 – March 1. 2022*, pages 10965–10973. https://doi.org/10.1609/aaai.v36i10.21344

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.519

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2019. Sequence-to-nuggets: Nested entity mention detection via anchor-region networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5182–5192, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1511

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proceedings of the ICLR Conference May. 2019*. OpenReview.net.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 857–867, Lisbon, Portugal. Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1102

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. 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)*, pages 3036–3046, Minneapolis, Minnesota. Association for Computational Linguistics.

https://doi.org/10.18653/v1/N19-1308

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford coreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics. https://doi.org/10.3115/v1/P14-5010

Danielle L. Mowery, Sumithra Velupillai, Brett R. South, Lee M. Christensen, David Martínez, Liadh Kelly, Lorraine Goeuriot, Noémie Elhadad, Sameer Pradhan, Guergana K. Savova, and Wendy W. Chapman. 2014. Task 2: ShARe/CLEF eHealth evaluation lab 2014. In *Working Notes for CLEF Conference September. 2014*, volume 1180, pages 31–42.

Aldrian Obaja Muis and Wei Lu. 2016. Learning to recognize discontiguous entities. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 75–84. Austin, Texas. Association for Computational Linguistics.

Sameer Pradhan, Noémie Elhadad, Brett R. South, David Martínez, Lee M. Christensen, Amy Vogel, Hanna Suominen, Wendy W. Chapman, and Guergana K. Savova. 2013a. Task 1: ShARe/CLEF eHealth evaluation lab 2013. In *Working Notes for CLEF Conference September. 2013*, volume 1179.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013b. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the 17th Conference Computational Natural Language Learning, August. 2013*, pages 143–152.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the 7th Conference Natural Language Learning, May 31 – June 1. 2003*, pages 142–147. https://doi.org/10.3115/1119176.1119195

Yongliang Shen, Xinyin Ma, Zeqi Tan, Shuai Zhang, Wen Wang, and Weiming Lu. 2021. Locate and label: A two-stage identifier for nested named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2782–2794, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.216

Takashi Shibuya and Eduard H. Hovy. 2020. Nested named entity recognition via second-best sequence learning and decoding. *Transactions of the Association for Computational Linguistics*, 8:605–620. https://doi.org/10.1162/tacl_a_00334

Mohammad Golam Sohrab and Makoto Miwa. 2018. Deep exhaustive model for nested named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2843–2849. Brussels, Belgium. https://doi.org/10.18653/v1/D18-1309

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested NER through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1527

Buzhou Tang, Jianglu Hu, Xiaolong Wang, and Qingcai Chen. 2018. Recognizing continuous and discontinuous adverse drug reaction mentions from social media using LSTM-CRF. *Wireless Communications and Mobile Computing*, 2018. https://doi.org/10.1155/2018/2379208

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems December. 2017*, pages 5998–6008.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 multilingual training corpus LDC2006T06. *Web Download. Philadelphia: Linguistic Data Consortium*, 110(110):261–276.

Bailin Wang and Wei Lu. 2018. Neural segmental hypergraphs for overlapping mention recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 204–214, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1019

Bailin Wang and Wei Lu. 2019. Combining spans into entities: A neural two-stage approach for recognizing discontiguous entities. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6215–6223, Hong Kong, China. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1644

Bailin Wang, Wei Lu, Yu Wang, and Hongxia Jin. 2018. A neural transition-based model for nested mention recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1011–1017, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1124

Yucheng Wang, Bowen Yu, Hongsong Zhu, Tingwen Liu, Nan Yu, and Limin Sun. 2021. Discontinuous named entity recognition as maximal clique discovery. 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)*, pages 764–774, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.63

Hang Yan, Bocao Deng, Xiaonan Li, and Xipeng Qiu. 2019. TENER: Adapting transformer encoder for named entity recognition. *CoRR*, abs/1911.04474v3. https://doi.org/10.48550/arXiv.1911.04474

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. 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)*, pages 5808–5822, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.451

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.577

Zexuan Zhong and Danqi Chen. 2021. A frustratingly easy approach for entity and relation extraction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 50–61, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.5

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 434–443, Sofia, Bulgaria. The Association for Computer Linguistics.

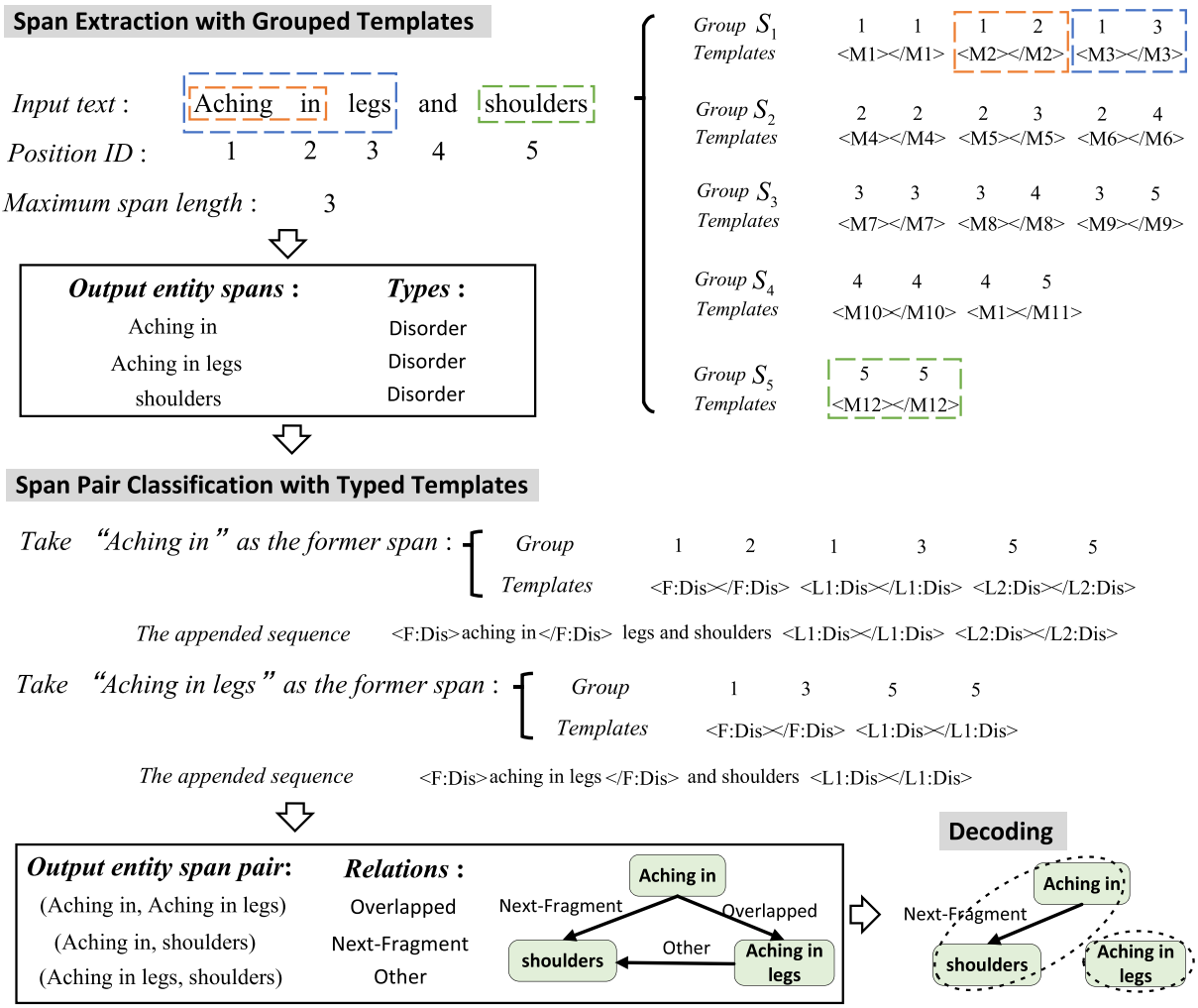

Figure 5: Step-by-step examples of two stages and the result decoding process.

A Step-by-step Examples of Our Work

The detailed examples of our work step by step are shown in Figure 5.