BoningKnife: Joint Entity Mention Detection and Typing for Nested NER via prior Boundary Knowledge

Huiqiang Jiang♣, **Guoxin Wang**◇, **Weile Chen**♡, **Chengxi Zhang**♣, **Börje F. Karlsson**♦
♣Peking University, ♦Microsoft Research
◇Microsoft Azure AI, ♡Harbin Institute of Technology
{jhq, chengxi_zhang}@pku.edu.cn, {guow, borjekar}@microsoft.com, chen.weile7@gmail.com

ABSTRACT

While named entity recognition (NER) is a key task in natural language processing, most approaches only target flat entities, ignoring nested structures which are common in many scenarios. Most existing nested NER methods traverse all sub-sequences which is both expensive and inefficient, and also don't well consider boundary knowledge which is significant for nested entities. In this paper, we propose a joint entity mention detection and typing model via prior boundary knowledge (*BoningKnife*) to better handle nested NER extraction and recognition tasks. *BoningKnife* consists of two modules, MentionTagger and TypeClassifier. MentionTagger better leverages boundary knowledge beyond just entity start/end to improve the handling of nesting levels and longer spans, while generating high quality mention candidates. TypeClassifier utilizes a two-level attention mechanism to decouple different nested level representations and better distinguish entity types. We jointly train both modules sharing a common representation and a new dual-info attention layer, which leads to improved representation focus on entity-related information. Experiments over different datasets show that our approach outperforms previous state of the art methods and achieves 86.41, 85.46, and 94.2 F1 scores on ACE2004, ACE2005, and NNE, respectively.[1]

1 Introduction

Named Entity Recognition (NER) is a fundamental tasks in natural language processing (NLP), which aims to extract and recognize named entities, like person names, organizations, geopolitical entities, etc., in unstructured text. However, in addition to flat entity mentions, nested or overlapping entities are commonplace in natural language. Such nested entities bring richer entity knowledge and semantics and can be critical to facilitate various downstream NLP tasks and real-world applications. As an example of their frequency, nested entities account for 35.19%, 30.80%, and 66.14% of mentions in standard datasets like ACE2004 Doddington et al. [2004], ACE2005 Walker et al. [2006], and NNE Ringland et al. [2019], respectively.

Nonetheless, the standard method for classic NER treats the problem as a sequence labeling task which has difficulty recognizing entities with nested structures directly Alex et al. [2007], Lu and Roth [2015], Katiyar and Cardie [2018]. With that in mind, various approaches to recognizing nested entities have been proposed. From hyper-graph based methods Lu and Roth [2015], Wang and Lu [2018], Marinho et al. [2019] which design expressive tagging schemas, to span-based methods which classify the categories of sub-sequences Sohrab and Miwa [2018], Luan et al. [2019], Xia et al. [2019], Fisher and Vlachos [2019]. In order to improve output quality, most more recent approaches to nested NER adopt structures that require the enumeration or heuristic traversal of all sub-sequences, which leads to inefficiency, and lack effective use of boundary information, which is very significant for nested entities. Recently, Zheng et al. [2019] and Tan et al. [2020] explore using boundary knowledge to enhance recognition of nested entities. But both focus only on entity start/end information, which face limitations in handling long entity spans, the interaction of entity start/end(s), and lack region information. Moreover, decoupling the different nested levels of entity information remains a big problem in the nested NER task. For example, the spans *"the leader of the Hezbollah in Syrian-occupied*

---

[1]Work performed at Microsoft Research Asia between 2019/2020.

... the leader of the Hezbollah in Syrian-occupied Lebanon...

Figure 1: An example of nested mentions in ACE2004.

Lebanon", *"the Hezbollah in Syrian-occupied Lebanon"*, and *"Syrian-occupied Lebanon"* all share the same end token (see Fig 1). Shared contextual representations tend to focus on the outermost entity type (in the above case, PER).

In this paper, we propose a novel joint entity mention detection and typing model via prior boundary knowledge, *BoningKnife*, which can carve entity boundaries accurately and tease out type information more precisely. Our model consists of two main components, MentionTagger and TypeClassifier, which are jointly trained with a common encoder representation and a shared dual-info attention layer. MentionTagger performs mention detection by better leveraging boundary knowledge beyond just entity start/end to better handle nesting levels and longer spans. This improved representation of boundary knowledge both addresses limitations of previous systems and allows the generation of high quality mention candidates, which are critical for the overall system efficiency. TypeClassifier then utilizes a new two-level attention mechanism to decouple different nested level representations and better distinguish entity types. Moreover, the offshoots of MentionTagger entity token detection are further leveraged in the dual-info attention layer to improve joint training performance.

Experimental results on three datasets show that our approach achieves significant improvements over state-of-the-art methods across multiple nested NER datasets. Further analysis and case studies demonstrate the effectiveness of each component in the model and its different sub-task strategies of mention detection and typing attention layers. Moreover, our approach also achieves higher efficiency without drop in quality.

2 Related Work

Traditionally, most approaches formalize the NER task as a sequence labeling problem, which assigns a single label to each token in a sentence. Shen et al. [2003], Zhang et al. [2004], Zhou [2006] adopt bottom-up methods, which performs entity recognition from inner to outer mentions, following hand-crafted rules.

Lu and Roth [2015] introduces the idea of using a graph structure to connect tokens with multiple entities. While Muis and Lu [2018], Wang and Lu [2018], Katiyar and Cardie [2018], Wang and Lu [2019] propose hypergraphs and different methods to utilize graph information for nested NER.

Transition-based models, which assemble a shift-reduce structure to detect a nested entity, have also been proposed. Wang et al. [2018] builds a forest structure based on shift-reduce parsing. Marinho et al. [2019] uses a stack structure to construct the transition-shift-reduce model. And Ju et al. [2018] proposes a dynamically stacked multiple LSTM-CRF model to recognize the entity in an inside-out manner until no outer entity is plucked.

Span-based methods are another class of methods to recognize nested entities by classifying sub-sequences Xia et al. [2019]. Luan et al. [2019] proposes a graph-based model which leverages entity linking to improve NER performance. Fisher and Vlachos [2019] introduces a merge-and-label method which uses nested entity hierarchy features. Strakova et al. [2019] views the nested NER task as a seq2seq generator problem, in which the input is a list of sentences and output target entities list. Shibuya and Hovy [2020] introduces an improved CRF model that recursively decoder the entity from outside to inside.

However, most previous methods need to traverse all sub-sequences and lack boundary knowledge, which is significant for nested entities. To try and mitigate such shortcomings, Zheng et al. [2019] defines a boundary detection task to generate a mention candidate set based on the entity start/end, followed by typing all mentions in the candidate set. And Tan et al. [2020] splits boundary information into two sub-tasks (entity start and entity end), before classifying candidates. While both works use boundary knowledge, they focus only on entity start/end, which does not fully represent boundary information and lead to issues such as not handling long spans well. *BoningKnife* jointly trains entity mention detection and typing modules and utilizes an extended representation of boundary knowledge to address such limitations.
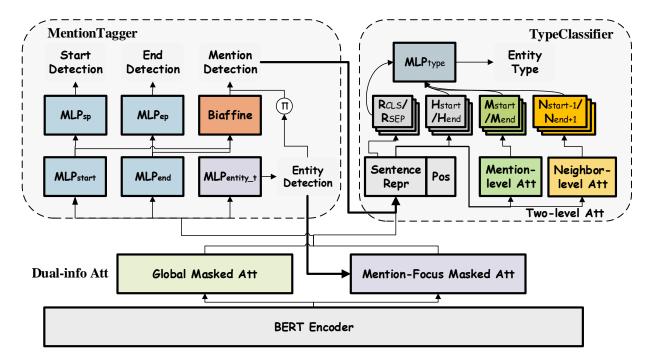
3 Problem and Methodology

Figure 2: Framework of the proposed *BoningKnife*, including Encoder Layer, MentionTagger, and TypeClassifier.

In this section, we define the nested NER task, and then elaborate on our proposed solution. Fig 2 illustrates the framework of the proposed (*BoningKnife*). Specifically, we jointly train tagger and classifier, where the former (MentionTagger) extracts potential mention spans and generates mention candidates by leveraging an improved representation of boundary knowledge, and the later (TypeClassifier) classifies mention candidates into predefined entity types.

3.1 Problem Statement

Let $s \in \mathbb{S}$ denote sentence data and $c \in \mathbb{C}$ denote entity label data, where $\mathbb{S}$ and $\mathbb{C}$ are the vector space of sentences and labels. Given a sentence $s = [t_i]_{1 \leq i \leq N}$, where N means the length of sentences s and t_i represents the i-th token of s, the NER task object is to extract all semantic elements $(l_e, r_e, c_e) \triangleq e \in \mathbb{E}$ where l_e, r_e are the element start/end indices, c_e means the element corresponding to a predefined label, and $\mathbb{E}$ is the entity space.

Essentially, nested NER aims to learn a $\mathcal{R}^{N \times N \times |\mathbb{C}|}$ space representation (only for the upper triangular matrix), where $|\mathbb{C}|$ is the size of entity categories and non-entity, and each value in the matrix represents the span type probability $\mathcal{P}(e|s)$. We decompose the target probability $\mathcal{P}(e|s)$ into the product of two conditional probabilities (detection and typing) with a latent parameter (span).

$$\mathcal{P}(e|s) = \sum_i^{|\mathbb{E}|} \mathcal{P}(l_i, r_i|s) \cdot \mathcal{P}(e|s, l_i, r_i) \tag{1}$$

In the formula, we discard the term with a small span probability $\mathcal{P}(l_i, r_i|s)$ in order to reduce the amount of calculations.

3.2 Encoder Layer

Because of the importance of context to entities, it is necessary to infuse information from different nested entities into one token. We propose a Dual-info attention structure to obtain entity semantic knowledge from both the token itself and others. This Dual-info attention representation is the input of two main sub-components of the model.
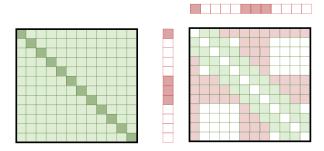
Figure 3: Mask matrix of Dual-info attention (windows size is 2). **Left**: global mask. **Right**: mention-focus mask which fuses "entity detection" subtasks and local context.

For the attention architecture, we use a pre-LayerNorm residual connection and multi-head attention mechanism.

$$\text{AttentionBlock}(\boldsymbol{s}, \text{MASK}_s) = \text{MultiHead}(\text{LayerNorm}(\boldsymbol{s}), \text{MASK}_s) + \boldsymbol{s} \tag{2}$$

where $\text{MASK}_s \in \mathbb{R}^{N \times N}$ is the attention mask matrix.

Dual-info Attention consists of Global Masked attention and Mention-focus Masked attention layers. Fig 3 shows the masked matrix example of Dual-info attention based on $\text{BERT}_{\text{base}}$ Devlin et al. [2018]. We use $\boldsymbol{b}_i$, the BERT representation of the i-th token in the sentence, to compute the Dual-info attention representation $\boldsymbol{R}_i$.

$$\boldsymbol{R}_i = \text{Linear}([\text{AttentionBlock}(\boldsymbol{b}_i, \text{MASK}_{\text{global}}); \text{AttentionBlock}(\boldsymbol{b}_i, \text{MASK}_{\text{focus}})]) \tag{3}$$

Global Masked Attention considers every token from the same sentence, which makes the representation more contextual, while Mention-focus Masked Attention uses entity detection from MentionTagger (Sec 3.3) and local context to construct attention weights. For tokens not in mention candidates, we encode them from the representation of mention candidates' tokens and local context. Otherwise, we encode mention candidates' tokens by calculating the attention weighted sum of all tokens except itself. This approach tries to emphasise information related to: entity to entity, token to entity, and entity types. The ablation experiments (Sec 5.1) and attention discussion (Sec 5.3) further showcase its effects.

3.3 MentionTagger

The mention tagger module aims to extract entity mention candidates and compute their corresponding probabilities in a sentence.

It is onerous to learn the mention detection matrix in $\mathbb{R}^{N \times N \times 2}$ space directly. A basic idea is to traverse all sub-sequences based on a shared representation. Building a high dimensional matrix concatenating the representation of entity's start/end token. However, this method misses the interaction between start and end tokens, and increases the risk of over-fitting in the training stage.

The other extreme is to treat entity boundary information as two completely independent variables, like recent mention detection models Zheng et al. [2019], Tan et al. [2020] that only consider entity start/end tokens, which lack enough region contextual information. MentionTagger circumvents these two problems and utilizes three types of boundary information: entity start/end token, entity token itself, and mention region; which we term *prior boundary knowledge*. To infuse prior boundary distribution, we use three sub-tasks (start/end detection, entity detection, and mention detection) in training the tagger module.

Start/End detection Inspired by the biaffine model Dozat and Manning [2017], we use two MLPs ($\text{MLP}_{\text{start}}$ and MLP_{end}) to get a low dimension start/end representation $\boldsymbol{h}_{\text{start}}, \boldsymbol{h}_{\text{end}}$ and compute the span representation $\boldsymbol{S}_e(i)$ for span $\boldsymbol{e}(l_i, r_i)$. We also use a start/end detection sub-task to enhance start/end representation $\boldsymbol{h}_{\text{start}}, \boldsymbol{h}_{\text{end}}$ and apply two other MLPs($\text{MLP}_{\text{startPoint}}$ and $\text{MLP}_{\text{endPoint}}$) to project $\boldsymbol{h}_{\text{start}}/\boldsymbol{h}_{\text{end}}$ to the category space. For span $\boldsymbol{e}(l_i, r_i)$, these are:

$$\begin{aligned}\boldsymbol{h}_{\text{start}}(i) &= \text{MLP}_{\text{start}}\left(\boldsymbol{R}_{l_i}\right) \\ \boldsymbol{h}_{\text{end}}(i) &= \text{MLP}_{\text{end}}\left(\boldsymbol{R}_{r_i}\right)\end{aligned} \tag{4}$$

where $\boldsymbol{R}_{l_i}$ and $\boldsymbol{R}_{r_i}$ are the Dual-info Attention representation of the span $\boldsymbol{e}(l_i, r_i)$'s start/end token.

And the span vector $\boldsymbol{S}_e(i)$ generated from the low dimension start/end representation $\boldsymbol{h}_{\text{start}}, \boldsymbol{h}_{\text{end}}$,

$$\boldsymbol{S}_e(i) = \boldsymbol{h}_{\text{start}}(i)\boldsymbol{U}_{\text{span}}\boldsymbol{h}_{\text{end}}(i) + \boldsymbol{h}_{\text{start}}(i)\boldsymbol{u}_{\text{start}} + \boldsymbol{h}_{\text{end}}(i)\boldsymbol{u}_{\text{end}} + \boldsymbol{b}_{\text{span}} \tag{5}$$

where $\boldsymbol{U}_{\text{span}} \in \mathbb{R}^{N \times N \times d_{\text{span}}}$, $u_{\text{start}} \in \mathbb{R}^{d_{\text{low}} \times d_{\text{span}}}$, $u_{\text{end}} \in \mathbb{R}^{d_{\text{low}} \times d_{\text{span}}}$ are self-learned parameters.

$$\mathcal{P}_\theta(\boldsymbol{Y}_{\text{start}}) = \text{softmax}(\text{MLP}_{\text{startPoint}}(\boldsymbol{h}_{\text{start}}(i)))$$
$$\mathcal{P}_\theta(\boldsymbol{Y}_{\text{end}}) = \text{softmax}(\text{MLP}_{\text{endPoint}}(\boldsymbol{h}_{\text{end}}(i))) \tag{6}$$

where $\mathcal{P}_\theta(\boldsymbol{Y}_{\text{start}})$ and $\mathcal{P}_\theta(\boldsymbol{Y}_{\text{end}})$ are the output probability of the start/end detection sub-task.

$$\mathcal{L}_{\text{start}} = \frac{1}{N} \sum_{i \in [1,N]} - \log \mathcal{P}_\theta(Y_{\text{start},i} = y_{\text{start},i})$$
$$\mathcal{L}_{\text{end}} = \frac{1}{N} \sum_{i \in [1,N]} - \log \mathcal{P}_\theta(Y_{\text{end},i} = y_{\text{end},i}) \tag{7}$$

where $y_{\text{start},i}$ are the ground truth labels of the start/end detection sub-task and $\mathcal{L}_{\text{start}}, \mathcal{L}_{\text{end}}$ are the training losses of the start/end sub-task, respectively.

Entity detection Notice that only using the start/end information does not define a span boundary. Two high probability start i/end j don't mean the probability of span $\boldsymbol{e}(l_i, r_i)$ is high. For example, in the sentence *"Joe went to school."*, the probability of start token *"Joe"* and end token *"school"* are both high, but the probability of span *"Joe went to school"* being an entity is low. Applying the entity detection information (verdict token w/o belonging to at least one entity) can help address this problem. We reduce the span probability with large spacing by accumulating the entity detection probability values.

For the token index k in a sentence, we have:

$$\mathcal{P}_\theta(\boldsymbol{Y}_{\text{enityDetection}}) = \text{softmax}(\text{MLP}_{\text{enityDetection}}(\boldsymbol{R}_k)) \tag{8}$$

where $\mathcal{P}_\theta(\boldsymbol{Y}_{\text{enityDetection}})$ means the output probability of token k belonging to a span.

In this sub-task, the entity detection loss function $\mathcal{L}_{\text{entityDetection}}$ is defined as:

$$\mathcal{L}_{\text{entityDetection}} = \frac{1}{N} \sum_{i \in [1,N]} - \log \mathcal{P}_\theta(Y_{\text{enityDetection},i} = y_{\text{enityDetection},i}) \tag{9}$$

where $y_{\text{entityDetection},i}$ are the ground truth labels of the entity detection sub-task.

Mention detection Using the span representation $\boldsymbol{S}_e(i)$ and entity detection probability $\mathcal{P}_\theta(\boldsymbol{Y}_{\text{enityDetection}})$, we compute the mention detection probability for all sub-sequences in the sentence. For the span $\boldsymbol{e}(l_i, r_i)$, these are:

$$\mathcal{P}_\theta(\boldsymbol{Y}_{\text{mention}}) = \text{softmax}(\text{MLP}_{\text{mention}}(\boldsymbol{S}_e(i) \cdot \prod_{k \in [l_i, r_i]} \mathcal{P}_\theta(\boldsymbol{Y}_{\text{enityDetection}}))) \tag{10}$$

where $\mathcal{P}_\theta(\boldsymbol{Y}_{\text{mention}})$ means the output probability of the mention detection sub-task.

The mention detection loss function $\mathcal{L}_{\text{mention}}$ is calculated as follows:

$$\mathcal{L}_{\text{mention}} = \frac{1}{N^2} \sum_{i \in [1,N]} \sum_{j \in [1,N]} - \log \mathcal{P}_\theta(Y_{\text{mention},i,j} = y_{\text{mention},i,j}) \tag{11}$$

where $y_{\text{entityDetection},i}$ are the ground truth labels of the mention detection sub-task. After getting the mention probability $\mathcal{P}_\theta(\boldsymbol{Y}_{\text{mention}})$ of each mention pair $\boldsymbol{e}(l_i, r_i)$, we use a threshold hyper-parameter to generate the mention candidate $\mathbb{M}_{\text{candidate}}$.

MentionTagger not only outputs a mention candidate set $\mathbb{M}_{\text{candidate}}$ as the input of TypeClassifier, but the offshoots of its internal entity token detection $\mathcal{P}_\theta(\boldsymbol{Y}_{\text{enityDetection}})$ are fed to the shared Dual-info attention layer as a mention-focus mask matrix $\text{MASK}_{\text{focus}}$, improving its representation of entity semantic information.

3.4 TypeClassifier

After obtaining mention candidates from MentionTagger, TypeClassifier aims to predict the probability of entity types for each candidate. We utilize a mention decoupling layer (MDL) to focus on the current mention semantic information, including two-level attention and a four-level representation (see Fig 2), for each span.

We apply a dimensional position embedding $\boldsymbol{P}_i$ over the Dual-info Attention encoder embedding $\boldsymbol{R}_i$, as position-wise token representation $\boldsymbol{H}_i$, and consider $\boldsymbol{H}_i$ as the input to the two-level attention component.

$$\boldsymbol{H}_i = \text{Linear}([\boldsymbol{R}_i; \boldsymbol{P}_i]) \tag{12}$$

where $\boldsymbol{P}_i$ is a learnable position embedding.

Two-level attention combines mention-level attention $\text{MentionAttention}(\boldsymbol{s})$ (to make semantic information focus on internal spans) and neighbor-level attention $\text{NeighborAttention}(\boldsymbol{s})$ (which emphasizes the knowledge from span boundaries and contextual tokens). Both utilize have different mask matrices. Mention-level attention can only see the mention tokens, while neighbor-level attention can only see the remaining tokens. This attention is defined as:

$$\text{MentionAttention}(\boldsymbol{s}) = \text{AttentionBlock}(\boldsymbol{H}, \text{MASK}_{\text{mention}})$$
$$\text{NeighborAttention}(\boldsymbol{s}) = \text{AttentionBlock}(\boldsymbol{H}, \text{MASK}_{\text{neighbor}}) \tag{13}$$

where $\text{MASK}_{\text{mention}}, \text{MASK}_{\text{neighbor}}$ are the difference mask matrices, and AttentionBlock is the attention architecture mentioned in eq (2).

Four-level representation For each span, we align and combine four fine-grained representations, thereby improving the model's understanding of entity boundaries and entity semantics. Taking the span $\boldsymbol{e}(l_i, r_i)$, the detail representation encompasses:

- Sentence-level Representation,

$$\boldsymbol{E}_{\text{sentence}}(\boldsymbol{s}) = \text{Linear}([\boldsymbol{R}_{\text{[CLS]}}; \boldsymbol{R}_{\text{[SEP]}}]) \tag{14}$$

- Position-wise Token Representation,

$$\boldsymbol{E}_{\text{position}}(\boldsymbol{s}) = \text{Linear}([\boldsymbol{H}_{\text{start}}; \boldsymbol{H}_{\text{end}}]) \tag{15}$$

- Mention-level Representation,

$$\boldsymbol{E}_{\text{mention}}(\boldsymbol{s}) = \text{Linear}([\text{MentionAttention}_{\text{start}}; \text{MentionAttention}_{\text{end}}]) \tag{16}$$

- Neighbor-level Representation,

$$\boldsymbol{E}_{\text{neighbor}}(\boldsymbol{s}) = \text{Linear}([\text{NeighborAttention}_{\text{previous}}; \text{NeighborAttention}_{\text{next}}]) \tag{17}$$

where $X_{\text{[CLS]}}, X_{\text{[SEP]}}, X_{\text{start}}, X_{\text{end}}, X_{\text{previous}}, and X_{\text{next}}$ are the [CLS], [SEP], start, end, previous mention, and next mention token representations, respectively.

These four different features are combined into TypeClassifer's representation:

$$\boldsymbol{T}(\boldsymbol{s}) = \text{Linear}([\boldsymbol{E}_{\text{sentence}}; \boldsymbol{E}_{\text{position}}; \boldsymbol{E}_{\text{mention}}; \boldsymbol{E}_{\text{neighbor}}]) \tag{18}$$

In the same way, the predicted classification probability is output through the Softmax function.

$$\mathcal{P}_\theta(\boldsymbol{Y}_{\text{type}}) = \text{softmax}(\text{MLP}_{\text{type}}(\boldsymbol{T}(\boldsymbol{s}))) \tag{19}$$

The corresponding loss function $\mathcal{L}_{\text{type}}$ is:

$$\mathcal{L}_{\text{type}} = \frac{1}{|\mathbb{M}_{\text{candidate}}|} \sum_{i \in [1, |\mathbb{M}_{\text{candidate}}|]} -\log \mathcal{P}_\theta(Y_{\text{type},i} = y_{\text{type},i}) \tag{20}$$

3.5 Optimization Objective

In MentionTagger, we joint optimize the above detection sub-tasks. All losses are based on cross entropy loss. To balance difference loss, we apply a focal loss style self-adjusting weight strategy. For larger losses, the weights will be correspondingly scaled up to improve the learning process.

$$\mathcal{L}_{\text{mentionTagger}} = \alpha_{\text{start}}\mathcal{L}_{\text{start}} + \alpha_{\text{end}}\mathcal{L}_{\text{end}} + \alpha_{\text{entityDetection}}\mathcal{L}_{\text{entityDetection}} + \alpha_{\text{mention}}\mathcal{L}_{\text{mention}} \tag{21}$$

where,

$$\text{LearnScore}_i = (1 - \exp(-\mathcal{L}_i))^\beta$$

$$\alpha_i = \frac{\text{LearnScore}_i}{\sum_j \text{LearnScore}_j} \tag{22}$$

where LearnScore_i is a score to judge degree of sub-task training, which borrows from focal loss Lin et al. [2017] and α_i is a normalization version of LearnScore_i, $j \in \{\text{start}, \text{end}, \text{entityDetection}, \text{mention}\}$.

We jointly train MentionTagger and TypeClassifier as a multi-task process alternately, where the shared representation layer and the entity detection prediction results in Dual-info attention enhance the connections between the two components. The overall optimization goal is:

$$\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{mentionTagger}} + \mathcal{L}_{\text{type}} \tag{23}$$

4 Experiments

4.1 Datasets

We evaluate our model on three nested NER datasets, ACE2004 Doddington et al. [2004], ACE2005 Walker et al. [2006], and NNE Ringland et al. [2019]; using the same splits as previous work Lu and Roth [2015], Wang and Lu [2018], Lin et al. [2019], Ringland et al. [2019].[2]. Table 1 shows the proportions of nested entities in the datasets range from 30.80% to 66.14%. ACE2004 and ACE2005 include 7 entity types, while NNE has 114 entity types. We report precision, recall, and micro-F1 metrics for all experiments.

	ACE2004	ACE2005	NNE
Documents	443	464	2,312
Sentences	8,507	9,311	49,208
Mentions	27,753	31,102	279,795
Entity overlaps	9,767	9,579	185,054
Overlap ratio	35.19%	30.80%	66.14%
% of overlaps over all sub-sequences	1.11%	1.30%	1.47%

Table 1: Statistics of the ACE2004, ACE2005, and NNE nested NER datasets.

4.2 Baseline and Experimental Settings

We compare *BoningKnife* with a set of representative models and the recent state of the art.

- Wang and Lu [2018], a graph-based model using LSTM to learn a feature encoder;
- Xia et al. [2019], which is a detect-classify model without boundary knowledge;
- Luan et al. [2019], a graph-based model which leverages entity linking to improve NER;
- Fisher and Vlachos [2019], a merge-and-label model with hierarchical features;
- Strakova et al. [2019], which treats the nested NER task as a seq2seq problem;
- Shibuya and Hovy [2020], which extracts nested entities recursively with CRF;
- Tan et al. [2020], which combines entity start/ end probabilities.

Moreover, we utilize $\text{BERT}_{\text{base}}$ itself as a lower-bound baseline for the BERT-based models.[3]

We perform a random search strategy for hyperparameter optimization and select the best settings on the development sets. We initialize the loss weight parameter as $\beta = 0.5$ in MentionTagger and set the max neighbor window size to

---

[2]ACE2004 / ACE2005 as in `https://statnlp-research.github.io/publications/`, and NNE as in `https://github.com/nickyringland/nested_named_entities`

[3]NNE results for Shibuya and Hovy [2020] are reported by using their public code in `https://github.com/yahshibu/nested-ner-2019-bert`.

Model	ACE2004			ACE2005		
	P(%)	**R(%)**	**F1(%)**	**P (%)**	**R(%)**	**F1 (%)**
Wang and Lu [2018]	78.0	72.4	75.1	76.8	72.3	74.5
Xia et al. [2019] [ELMO]	81.7	77.4	79.5	79.0	77.3	78.2
Luan et al. [2019]	-	-	84.7	-	-	82.9
BERT merge outmost & inmost[†]	80.55	79.23	79.88	78.12	82.71	80.35
Fisher and Vlachos [2019][†]	-	-	-	82.7	82.1	82.4
Strakova et al. [2019][†]	-	-	84.4	-	-	84.3
Shibuya and Hovy [2020][†]	85.23	84.72	84.97	83.30 ± 0.22	84.69 ± 0.37	83.99 ± 0.27
Tan et al. [2020][†]	85.8	84.8	85.3	83.8	83.9	83.9
BoningKnife[†]	$\mathbf{85.98 \pm 0.36}$	$\mathbf{86.86 \pm 0.39}$	$\mathbf{86.41 \pm 0.24}$	$\mathbf{84.77 \pm 0.31}$	$\mathbf{86.16 \pm 0.43}$	$\mathbf{85.46 \pm 0.32}$

Table 2: Results of the proposed *BoningKnife* and prior state-of-the-art methods over the ACE2004/2005 test sets. [†] denotes models utilizing BERT. '-' denotes results not reported.

Model	**P(%)**	**R(%)**	**F1(%)**
Wang et al. [2018]	77.4	70.1	73.6
BERT_{base} merge outmost & inmost	80.43	74.94	77.59
Wang and Lu [2018]	91.8	91.1	91.4
Shibuya and Hovy [2020][4]	93.03	93.34	93.19
BoningKnife	**93.74**	**94.75**	**94.24**
	($\pm$ 0.33)	($\pm$ 0.24)	($\pm$ 0.05)

Table 3: Results of the proposed model and baselines over the NNE dataset.

128 in neighbor-level attention block NeighborAttention to control memory size. The hidden sizes of low dimension start/end representation $\boldsymbol{h}_{\text{start}}, \boldsymbol{h}_{\text{end}}$ are 84, the number of attention heads is 16, and the windows size of mention-focus attention mask matrix $\text{MASK}_{\text{focus}}$ is 2. Except for $\text{BERT}_{\text{base}}$, our model has around 24M parameters. We employ AdamW Loshchilov and Hutter [2019] as optimizer during training. Experiments are repeated 5 times for different random seeds on each corpus.

4.3 Results and Discussion

Tables 2 and 3 report the results of our model and the different baselines on the ACE2004/ACE2005 and NNE datasets. It can be seen that our proposed method outperforms all previous state-of-the-art methods, reaching 86.41, 85.46, and 94.24 in average micro-F1 score, on ACE2004, ACE2005, and NNE respectively.

Compared with the latest boundary-enhanced method [Tan et al., 2020], our method achieves 1.11 and 1.56 absolute point gains on ACE2004 and ACE2005 [5]. The boost comes mainly from recall improvements (2.06 to 2.26 points). MentionTagger is able to produce more precise mention candidates, which allows TypeClassifier to focus on distinguishing entity types, instead of filtering candidates as not viable. The improved precision of MentionTagger is further evidenced in Table 6.

In the NNE corpus, *BoningKnife* achieves 94.24 F1-score; an improvement of 1.05 points over the previous SOTA. We hypothesize that as NNE datasets has deeper nesting levels, [Tan et al., 2020]'s approach leads to error transmission in their recursive encoding process.

5 Analysis

5.1 Ablation & Flat/Nested Performance

To validate the contributions and effectiveness of different components in the proposed model, we introduce the following model variants to perform an ablation study:

---

[5]Unfortunately Tan et al. [2020] does not report results over NNE and did not release their code for further experiments

Method	P(%)	R(%)	F1(%)
BoningKnife	85.98	86.86	86.41
- w/o EntityDetection subtask	84.74	85.95	85.35 **(-1.06)**
- w/o Start/End sub-task	85.71	85.18	85.44 **(-0.97)**
- w/o Neighbor-level attention	85.44	86.56	86.00 **(-0.41)**
- w/o Two-level attention	84.75	86.43	85.58 **(-0.83)**
- w/o Mention-focus attention	85.64	86.63	86.13 **(-0.28)**
- w/o MentionTagger Stage[6]	87.25	83.86	85.52 **(-0.89)**

Table 4: Ablation study of the proposed *BoningKnife* over the ACE 2004 test set, where numbers in parenthesis denote performance change.

Table 4 highlights the performance contributions of each component in our proposed model, and removing any of them will generally lead to substantial performance drops. It can be seen that quality decreases significantly when either removing MentionTagger or its sub-tasks (entity token and start/end detection) sub-task, which indicates the proposed model makes effective usage of boundary knowledge (for example, to better handle long length entity spans). Without the proposed two-level attention in TypeClassifier, it becomes harder for the model to separate nested information and assign the proper type for nested entities; even more so than removing only the neighbor-level component of the two-level attention . This further demonstrates the benefits of the two-level structure and its ability to combine clear boundary and local context information. Lastly, while the effects of removing mention-focus mask attention are less prominent, it's still noticeable and removing this component leads to slower overall model convergence. Furthermore, Table 5 reports the Flat/Nested performance across datasets. It can be seen that *BoningKnife* excels in nested entities while remaining competitive on flat results; which further evidences the overall effectiveness of the model in leveraging boundary knowledge.

Method	ACE2004		ACE2005		NNE	
	Flat	Nested	Flat	Nested	Flat	Nested
BoningKnife	84.32	**87.10**	84.54	**86.23**	84.45	**94.73**
-w/o MentionTagger	83.56	86.27	84.16	85.45	83.89	93.65
Shibuya and Hovy [2020]	**84.45**	85.14	**84.86**	83.13	84.26	93.67

Table 5: **Flat/Nested** F1-scores over the ACE2004, ACE2005, and NNE test sets.

5.2   Time Complexity

Method	ACE2004		ACE2005		NNE	
	P	Time (s)	P	Time (s)	P	Time (s)
BoningKnife	89.46	479	87.22	649	95.48	7841
-w/o MentionTagger	31.54	839	32.58	1190	30.71	32758
TimeRatio		1.75X		1.83X		4.18X

Table 6: Comparison of mention precision and training time cost per epoch, between E2E system using MentionTagger and mention strategy from Tan et al. [2020].

Similarly to Tan et al. [2020], our method substantially improves the time complexity over typical span-based methods by generating high-quality candidates, which greatly reduce complexity and training time. Span-based models, which require traversing all sub-sequences, have $O(m \cdot n^2)$ time complexity, where m is the count of tags. Efficiently reducing the number of candidates is key in a two-step system like *BoningKnife*, as span classifying time complexity is determined by the number of candidates in its input. To measure the speedup from our approach due to its improved candidate generation and provide a comparison with Tan et al. [2020], we run two experiments: i) the complete *BoningKnife*

---

[6]Replacing MentionTagger with using entity start/end to generate mention candidates, similarly to Tan et al. [2020].

Sentence: The **Coventry University researchers who report the findings in the British journal of sports medicine** say anxiety and depression are common among **those so injured**, possibly as a result of pain and impaired mobility.

	SPAN	p_{start}	p_{end}	$p_{mention}$	p_s	p_{type}	y_{type}	y_{golden}
(a)	The Coventry University researchers who report the findings in the British journal of sports medicine	1.000	0.999	1.000	0.942	1.000	PER	PER
(b)	Coventry University	0.595	1.000	0.945	0.502	0.998	ORG	ORG
(c)	who	1.000	0.999	1.000	0.997	1.000	PER	PER
(d)	those so injured	1.000	0.483	0.953	0.045	1.000	PER	PER
(e)	those so injured, possibly as a result of pain and impaired mobility	1.000	0.000	0.000	0.573	0.000	Non-entity	Non-entity
(f)	.	0.000	0.000	0.000	0.501	0.000	Non-entity	Non-entity

Table 7: An example where *BoningKnife* leverages prior boundary knowledge to better predict nested entity type. p_s from the ablation experiment "w/o ED subtask".

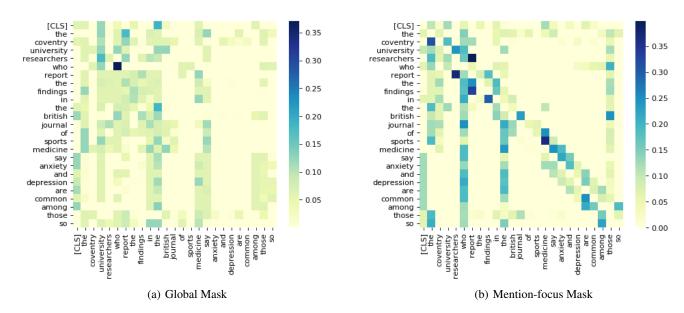

(a) Global Mask   (b) Mention-focus Mask

Figure 4: Visualization of the Dual-info Attention weights for the case study sentence (Table 7).

system and ii) *BoningKnife* - MentionTagger + mention strategy in Tan et al. [2020]. The experiments were run on a Ubuntu 16.04.6 server with Intel Xeon CPU E5-2690v3 @ 2.60GHz and one P100 GPU.

Table 6 reports the comparison between both experiments. We can see that our method provides significant speedup over the simpler modeling of boundary knowledge approach, especially with deeper nesting levels. *BoningKnife* is 1.75x, 1.83x, and 4.18x faster in ACE2004, ACE2005, and NNE, respectively, while still achieving higher quality.

5.3 Case Study and Attention Weight Visualization

Table 7 shows an example of *BoningKnife* prediction in ACE 2004. Span (d), *"those so injured"*, is a correct mention, but the probability of end token *"injured"* is small. For S/E based methods like Tan et al. [2020], Zheng et al. [2019], this span would likely be discarded, but in our method it is correctly identified. Compared with p_s only, the entity token detection knowledge reduces the number of high probability mentions (like (e) and (f)) inconsistent with the prior information, while not discarding very long entities, like mention (a).

Fig 4 shows the Dual-info attention weights for the sentence in Table 7. The global attention weights all focus on common keywords like *"university"*, *"the"*. While the mention-focus attention, focus on specific token neighbors, like *"report"* focusing on *"researchers"* and *"findings"*, which improve their semantic information. Also, additional tokens focus on the relevant entity tokens, like *"who"* focusing on the same entity type word *"those"* instead of on itself in global attention.

6 Conclusion

In this paper we propose a novel joint entity mention detection and typing model via prior boundary knowledge for the nested NER task. The proposed method effectively incorporates prior boundary knowledge information to generate high quality mention candidates, which greatly improves efficiency of the whole system. By introducing a Dual-info attention layer at the mention classification stage, it facilitates mention decoupling and more accurate mention classification at different levels. Experiments show that our system, *BoningKnife*, achieves state-of-the-art results on three standard benchmark datasets; and an ablation study further demonstrates the effectiveness of its components.

References

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Lrec*, volume 2, page 1. Lisbon, 2004. URL `http://www.lrec-conf.org/proceedings/lrec2004/pdf/5.pdf`.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57, 2006.

Nicky Ringland, Xiang Dai, Ben Hachey, Sarvnaz Karimi, Cécile Paris, and James R. Curran. NNE: A dataset for nested named entity recognition in english newswire. *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, 2019. URL `https://www.aclweb.org/anthology/P19-1510`.

Beatrice Alex, Barry Haddow, and Claire Grover. Recognising nested named entities in biomedical text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, pages 65–72. Association for Computational Linguistics, 2007. URL `https://www.aclweb.org/anthology/W07-1009`.

Wei Lu and Dan Roth. Joint mention extraction and classification with mention hypergraphs. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, 2015. URL `https://www.aclweb.org/anthology/D15-1102`.

Arzoo Katiyar and Claire Cardie. Nested named entity recognition revisited. In *Proc. Conf. North American Assoc. for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 861–871, 2018. URL `https://www.aclweb.org/anthology/N18-1079`.

Bailin Wang and Wei Lu. Neural segmental hypergraphs for overlapping mention recognition. *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, 2018. URL `https://www.aclweb.org/anthology/D18-1019`.

Zita Marinho, Alfonso Mendes, Sebastiao Miranda, and David Nogueira. Hierarchical nested named entity recognition. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 28–34, 2019. URL `https://www.aclweb.org/anthology/W19-1904`.

Mohammad Golam Sohrab and Makoto Miwa. Deep exhaustive model for nested named entity recognition. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 2843–2849, 2018. URL `https://www.aclweb.org/anthology/D18-1309`.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. A general framework for information extraction using dynamic span graphs. *Proc. Conf. North American Assoc. for Computational Linguistics (NAACL)*, 2019. URL `https://www.aclweb.org/anthology/N19-1308`.

Congying Xia, Chenwei Zhang, Tao Yang, Yaliang Li, Nan Du, Xian Wu, Wei Fan, Fenglong Ma, and Philip Yu. Multi-grained named entity recognition. *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, 2019. URL `https://www.aclweb.org/anthology/P19-1138`.

Joseph Fisher and Andreas Vlachos. Merge and label: A novel neural network architecture for nested ner. *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, 2019. URL `https://www.aclweb.org/anthology/P19-1585`.

Changmeng Zheng, Yi Cai, Jingyun Xu, Ho-fung Leung, and Guandong Xu. A boundary-aware neural model for nested named entity recognition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 357–366, Hong Kong, China, November 2019. Association for Computational Linguistics. doi:10.18653/v1/D19-1034. URL `https://www.aclweb.org/anthology/D19-1034`.

Chuanqi Tan, Wei Qiu, Mosha Chen, Rui Wang, and Fei Huang. Boundary enhanced neural span classification for nested named entity recognition. *Proc. Int. Conf. Proc. Conference on Alien Intelligence(AAAI)*, 2020.

Dan Shen, Jie Zhang, Guodong Zhou, Jian Su, and Chew-Lim Tan. Effective adaptation of a hidden markov model-based named entity recognizer for biomedical domain. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 49–56. Association for Computational Linguistics, 2003. URL `https://www.aclweb.org/anthology/W03-1307`.

Jie Zhang, Dan Shen, Guodong Zhou, Jian Su, and Chew-Lim Tan. Enhancing hmm-based biomedical named entity recognition by studying special phenomena. *Journal of biomedical informatics*, 37(6):411–422, 2004. URL `https://www.ncbi.nlm.nih.gov/pubmed/15542015`.

GD Zhou. Recognizing names in biomedical texts using mutual information independence model and svm plus sigmoid. *International Journal of Medical Informatics*, 75(6):456–467, 2006. URL `http://nlp.suda.edu.cn/~gdzhou/publication/zhougd2006_IJMI_BiomedicalNamedEntityRecognition.pdf`.

Aldrian Obaja Muis and Wei Lu. Labeling gaps between words: Recognizing overlapping mentions with mention separators. *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, 2018. URL `https://www.aclweb.org/anthology/D17-1276`.

Bailin Wang and Wei Lu. Combining spans into entities: A neural two-stage approach for recognizing discontiguous entities. *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, 2019. URL `https://www.aclweb.org/anthology/D19-1644`.

Bailin Wang, Wei Lu, Yu Wang, and Hongxia Jin. A neural transition-based model for nested mention recognition. *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, 2018. URL `https://www.aclweb.org/anthology/D18-1124`.

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. A neural layered model for nested named entity recognition. In *Proc. Conf. North American Assoc. for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1446–1459, 2018. URL `https://www.aclweb.org/anthology/N18-1131`.

Jana Strakova, Milan Straka, and Jan Hajic. Neural architectures for nested ner through linearization. *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, 2019. URL `https://www.aclweb.org/anthology/P19-1527`.

Takashi Shibuya and Eduard Hovy. Nested named entity recognition via second-best sequence learning and decoding. *Transactions of the Association for Computational Linguistics*, 8:605–620, 2020. doi:10.1162/tacl_a_00334. URL `https://aclanthology.org/2020.tacl-1.39`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proc. Conf. North American Assoc. for Computational Linguistics (NAACL)*, 2018. URL `https://www.aclweb.org/anthology/N19-1423`.

Timothy Dozat and Christopher D. Manning. Deep biaffine attention for neural dependency parsing. *Proc. Int. Conf. Learning Representations (ICLR)*, 2017.

Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017. URL `http://arxiv.org/abs/1708.02002`.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. Sequence-to-nuggets: Nested entity mention detection via anchor-region networks. *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, 2019. URL `https://www.aclweb.org/anthology/P19-1511`.

Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *Proc. Int. Conf. Learning Representations (ICLR)*, abs/1711.05101, 2019. URL `http://arxiv.org/abs/1711.05101`.