Nested Named Entity Recognition via Explicitly Excluding the Influence of the Best Path

Yiran Wang[1]*, **Hiroyuki Shindo**[2], **Yuji Matsumoto**[3], **Taro Watanabe**[2]
[1]National Institute of Information and Communications Technology (NICT), Kyoto, Japan
[2]Nara Institute of Science and Technology (NAIST), Nara, Japan
[3]RIKEN Center for Advanced Intelligence Project (AIP), Tokyo, Japan
yiran.wang@nict.go.jp, shindo@is.naist.jp,
yuji.matsumoto@riken.jp, taro@is.naist.jp

Abstract

This paper presents a novel method for nested named entity recognition. As a layered method, our method extends the prior second-best path recognition method by explicitly excluding the influence of the best path. Our method maintains a set of hidden states at each time step and selectively leverages them to build a different potential function for recognition at each level. In addition, we demonstrate that recognizing innermost entities first results in better performance than the conventional outermost entities first scheme. We provide extensive experimental results on ACE2004, ACE2005, and GENIA datasets to show the effectiveness and efficiency of our proposed method.

1 Introduction

Named entity recognition (NER), as a key technique in natural language processing, aims at detecting entities and assigning semantic category labels to them. Early research (Huang et al., 2015; Ma and Hovy, 2016; Lample et al., 2016) proposed to employ deep learning methods and obtained significant performance improvements. However, most of them assume that the entities are not nested within other entities, so-called flat NER. Inherently, these methods do not work satisfactorily when nested entities exist. Figure 1 displays an example of the nested NER task.

Recently, a large number of papers proposed novel methods (Fisher and Vlachos, 2019; Wang et al., 2020) for the nested NER task. Among them, layered methods solve this task through multi-level sequential labeling, in which entities are divided into several levels, where the term *level* indicates the depth of entity nesting, and sequential labeling is performed repeatedly. As a special case of layered method, Shibuya and Hovy (2020) force the next level entities to locate on the second-best path of the current level search space. Hence, their algorithm can repeatedly detect inner entities through applying a conventional conditional random field (CRF) (Lafferty et al., 2001) and then exclude the obtained best paths from the search space. To accelerate computation, they also designed an algorithm to efficiently compute the partition function with the best path excluded. Moreover, because they search the outermost entities first, performing the second-best path search only on the spans of extracted entities is sufficient, since inner entities can only exist within outer entities.

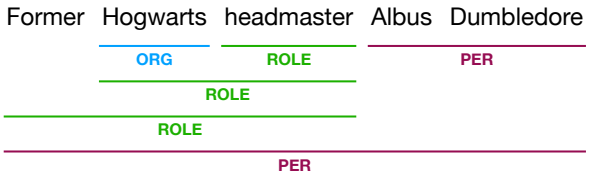

Figure 1: An example of nested NER.

However, we claim that the target path at the next level is neither necessary nor likely to be the second-best path at the current level. Instead, those paths sharing many overlapping labels with the current best path are likely to be the second-best path. Besides, Shibuya and Hovy (2020) reuse the same potential function at all higher levels. Thus, even though they exclude the best path, the influence of the best path is still preserved, since the emission scores of labels on the best path are used in the next level recognition. Moreover, these best path labels are treated as the target labels at the current level. However, if they are not on the best path of the next level, they will be treated as non-target labels at the next level, hence these adversarial optimization goals eventually hurt performance.

In this paper, we use **a different potential function at each level** to solve this issue. We propose to achieve this by introducing an encoder that pro-

*This work was done when the first author was at NAIST.

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 3547–3557
August 1–6, 2021. ©2021 Association for Computational Linguistics

duces a set of hidden states at each time step. At each level, we select some hidden states for entity recognition, then, remove these hidden states which have interaction with the best path labels before moving to the next level. In this way, the emission scores of these best path labels are completely different, so we can explicitly exclude the influence of the best path. Furthermore, we also propose three different selection strategies for fully leveraging information among hidden states.

Besides, Shibuya and Hovy (2020) proposed to recognize entities from outermost to inner. We empirically demonstrate that extracting **the innermost entities first** results in better performance. This may due to the fact that some long entities do not contain any inner entity, so using outermost-first encoding mixes these entities with other short entities at the same levels, therefore leading encoder representations to be dislocated. In this paper, we convert entities to the IOBES encoding scheme (Ramshaw and Marcus, 1995), and solve nested NER through applying CRF level by level.

Our contributions are considered as fourfold, (a) we design a novel nested NER algorithm to explicitly exclude the influence of the best path through using a different potential function at each level, (b) we propose three different selection strategies for fully utilizing information among hidden states, (c) we empirically demonstrate that recognizing entities from innermost to outer results in better performance, (d) and we provide extensive experimental results to demonstrate the effectiveness and efficiency of our proposed method on the ACE2004, ACE2005, and GENIA datasets.

2 Proposed Method

Named entities recognition task aims to recognize entities in a given sequence $\{x_t\}_{t=1}^{n}$. For nested NER some shorter entities may be nested within longer entities, while for flat NER there is no such case. Existing algorithms solve flat NER by applying a sequential labeling method, which assigns each token a label $y_t \in \mathcal{Y}$ to determine the span and category of each entity and non-entity simultaneously. To solve nested NER, we follow the previous layered method and extend this sequential labeling method with a multi-level encoding scheme. In this encoding scheme, entities are divided into several levels according to their depths, we apply the sequential labeling method level by level to recognize all entities.

2.1 Encoding Schemes

Shibuya and Hovy (2020) proposed to recognize the outermost entities first and recursively detect the nested inner entities. However, we find that detecting from the innermost entities results in better performance. We take the sentence in Figure 1 as an example to illustrate the details of these two encoding schemes. The results of the outermost-first encoding scheme look as follows.

(level 1)	B-PER	I-PER	I-PER	I-PER	E-PER
(level 2)	B-ROLE	I-ROLE	E-ROLE	B-PER	E-PER
(level 3)	O	B-ROLE	E-ROLE	O	O
(level 4)	O	S-ORG	S-ROLE	O	O
(level 5)	O	O	O	O	O
(level 6)	O	O	O	O	O

Labels B-, I-, E- indicate the current word is the beginning, the intermediate, and the end of an entity, respectively. Label S- means this is a single word entity, and label O stands for non-entity word. For example, the outermost entity "Former Hogwarts headmaster Albus Dumbledore" appears at the first level, while innermost entities "Hogwarts" and "headmaster" appear at the fourth level. Since there exists no deeper nested entity, the remaining levels contain only label O.

In contrast, the innermost-first encoding scheme converts the same example to the following label sequences.

(level 1)	O	S-ORG	S-ROLE	B-PER	E-PER
(level 2)	O	B-ROLE	E-ROLE	O	O
(level 3)	B-ROLE	I-ROLE	E-ROLE	O	O
(level 4)	B-PER	I-PER	I-PER	I-PER	E-PER
(level 5)	O	O	O	O	O
(level 6)	O	O	O	O	O

In this encoding scheme, innermost entities "Hogwarts", "headmaster", and "Albus Dumbledore" appear at the first level. Note that the innermost-first encoding scheme is not the simple reverse of the outermost-first encoding scheme. For example, the entity "Former Hogwarts headmaster" and the entity "Albus Dumbledore" appear at the same level in the outermost-first scheme but they appear at different levels in the innermost-first scheme.

2.2 Influence of the Best Path

Although the second-best path searching algorithm is proposed as the main contribution of Shibuya and Hovy (2020), we claim that forcing the target path at the next level to be the second-best path at

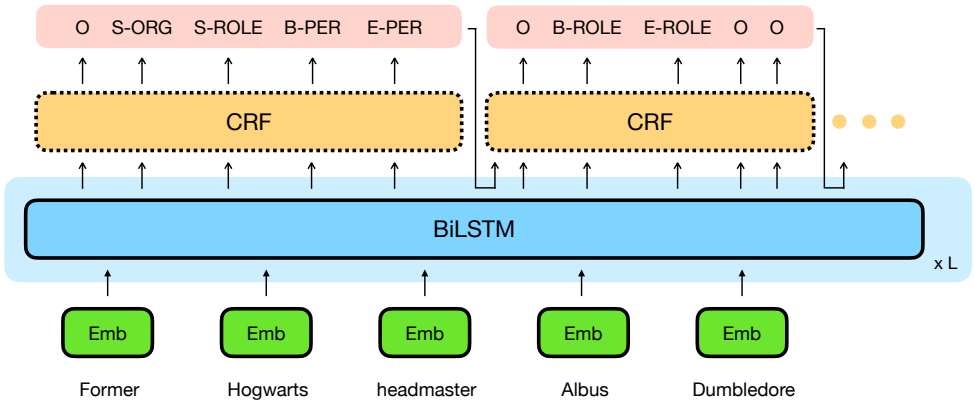

Figure 2: The architecture of our model. The dotted lines mean these components are shared across levels.

the current level is not optimal. As the innermost-first encoding example above, the best path at level 3 is `B-ROLE,I-ROLE,E-ROLE,O,O`. Therefore the second-best path is more likely to be one of those paths that share as many as possible labels with the best path, e.g., `B-ROLE,I-ROLE,E-ROLE,O,S-ORG`, rather than the actual target label sequence at level 4, i.e., `B-PER,I-PER,I-PER,I-PER,E-PER`, which does not overlap with the best path at all. In addition, Shibuya and Hovy (2020) reuse the same potential function at all higher levels. This indicates that, for instance, at level 3 and time step 1, their model encourages the dot product of the hidden state and the label embedding $\boldsymbol{h}_1^\top \boldsymbol{v}_{\texttt{B-ROLE}}$ to be larger than $\boldsymbol{h}_1^\top \boldsymbol{v}_{\texttt{B-PER}}$, while at level 4, the remaining influence of the best path reversely forces $\boldsymbol{h}_1^\top \boldsymbol{v}_{\texttt{B-PER}}$ to be larger than $\boldsymbol{h}_1^\top \boldsymbol{v}_{\texttt{B-ROLE}}$. These adversarial optimization goals eventually hurt performance and result in sub-optimal performance.

Therefore, the crux of the matter is to introduce different emission scores for different levels. For example, encouraging $\boldsymbol{h}_1^{3\top} \boldsymbol{v}_{\texttt{B-ROLE}} > \boldsymbol{h}_1^{3\top} \boldsymbol{v}_{\texttt{B-PER}}$ at level 3 and encouraging $\boldsymbol{h}_1^{4\top} \boldsymbol{v}_{\texttt{B-PER}} > \boldsymbol{h}_1^{4\top} \boldsymbol{v}_{\texttt{B-ROLE}}$ at level 4 will not lead to adversarial optimization directions anymore, where $\boldsymbol{h}_1^3$ and $\boldsymbol{h}_1^4$ are two distinctive hidden states to be used at levels 3 and 4, respectively.

To achieve this goal, we introduce a novel encoder which outputs m hidden states $\{\boldsymbol{h}_t^l\}_{l=1}^m$, where m is the number of levels, as an alternative to the conventional encoder which can only output a single hidden state $\boldsymbol{h}_t \in \mathbb{R}^{d_h}$ at each time step. To make a distinction between our m hidden states and the conventional single hidden state, we use the term *chunk* from now on to refer to these hidden states $\boldsymbol{h}_t^l \in \mathbb{R}^{d_h/m}$. We restrict chunk dimension to be d_h/m, so the total number of parameters remain unchanged.

2.3 Chunk Selection

As we mentioned above, our algorithm maintains a chunk set for each time step, through selecting and removing chunks, to exclude the influence of the best path. Naturally, how to select chunk becomes the next detail to be finalized.

For clarity, we use notation $\mathcal{H}_t^l$ to denote the chunk set at level l, and use $\mathcal{H}^l$ to refer to all of these chunk sets at level m across time steps, i.e., $\{\mathcal{H}_t^l\}_{t=1}^n$. Because we remove one and only one chunk at each time step, $|\mathcal{H}_t^l| + l = m + 1$ always holds.

An intuitive idea is to follow the original chunk order and simply to select the l-th chunk for level l. At level l, no matter to which label, the emission score is calculated by using $\boldsymbol{h}_t^l$. In this way, this *naive* potential function can be defined as follow,

$$\phi\left(y_{t-1}^l, y_t^l, \mathcal{H}_t^l\right) = A_{y_{t-1}^l, y_t^l} + \boldsymbol{h}_t^{l\top} \boldsymbol{v}_{y_t^l} \quad (1)$$

where $\mathbf{A} \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{Y}|}$ is the transition matrix, $\mathcal{Y}$ is the label set, $A_{y_{t-1}^l, y_t^l}$ indicates the transition score from label y_{t-1}^l to label y_t^l, and $\boldsymbol{v}_{y_t^l} \in \mathbb{R}^{d_h/m}$ is the embedding of label y_t^l. In this case, the l-th chunk $\boldsymbol{h}_t^l \in \mathcal{H}_t^l$ is just the chunk which have an interaction with target label, thus should be removed from $\mathcal{H}_t^l$.

$$\mathcal{H}_t^{l+1} = \mathcal{H}_t^l \setminus \{\boldsymbol{h}_t^l\} \quad (2)$$

One concern of the *naive* potential function is that it implicitly assumes the outputs of the encoder are automatically arranged in the level order instead of other particular syntactic or semantic order, e.g., the encoder may encodes all `LOC` related information at the first h_d/m dimensions while remaining

Algorithm 1: Training

input : first level chunk sets $\mathcal{H}^1$
input : target label sequences $\boldsymbol{y}^1, \cdots, \boldsymbol{y}^m$
output : negative log-likelihood $\mathcal{L}$
$\mathcal{L} \leftarrow 0$
for $l = 1$ **to** m **do**
  $\mathcal{L} \leftarrow \mathcal{L} - \log p\left(\boldsymbol{y}^l \mid \mathcal{H}^l\right)$
  for $t = 1$ **to** n **do**
    $\mathcal{H}_t^{l+1} \leftarrow \mathcal{H}_t^l \setminus \{\arg\max_{\boldsymbol{h} \in \mathcal{H}_t^l} \boldsymbol{h}^\top \boldsymbol{v}_{y_t^l}\}$
  end
end

Algorithm 2: Decoding

input : first level chunk sets $\mathcal{H}^1$
output : recognized entity set $\mathcal{E}$
$\mathcal{E} \leftarrow \varnothing$
for $l = 1$ **to** m **do**
  $\hat{\boldsymbol{y}}^l \leftarrow \arg\max_{\boldsymbol{y}' \in \mathcal{Y}^n} p\left(\boldsymbol{y}' \mid \mathcal{H}^l\right)$
  for $t = 1$ **to** n **do**
    $\mathcal{H}_t^{l+1} \leftarrow \mathcal{H}_t^l \setminus \{\arg\max_{\boldsymbol{h} \in \mathcal{H}_t^l} \boldsymbol{h}^\top \boldsymbol{v}_{\hat{y}_t^l}\}$
  end
  $\mathcal{E} \leftarrow \mathcal{E} \bigcup \texttt{label-to-entity}\left(\hat{\boldsymbol{y}}^l\right)$
end

ORG relevant information to the final h_d/m dimension. For instance, at level 3 time step 1, *naive* potential function forces $\boldsymbol{h}_1^{3\top} \boldsymbol{v}_{\texttt{B-ROLE}} > \boldsymbol{h}_1^{3\top} \boldsymbol{v}_{\texttt{B-PER}}$. But if there exists another chunk, say $\boldsymbol{h}_1^5$, which is more similar to $\boldsymbol{v}_{\texttt{B-PER}}$, then directly selecting $\boldsymbol{h}_1^5$ and forcing $\boldsymbol{h}_1^{3\top} \boldsymbol{v}_{\texttt{B-ROLE}} > \boldsymbol{h}_1^{5\top} \boldsymbol{v}_{\texttt{B-PER}}$ is more reasonable. Because it makes training harder than the former one, due to $\boldsymbol{h}_1^{5\top} \boldsymbol{v}_{\texttt{B-PER}} > \boldsymbol{h}_1^{3\top} \boldsymbol{v}_{\texttt{B-PER}}$. In other words, this selection strategy leads to $\boldsymbol{h}_t^{\sigma_1\top} \boldsymbol{v}_{y_t^1} > \boldsymbol{h}_t^{\sigma_2\top} \boldsymbol{v}_{y_t^2} > \ldots > \boldsymbol{h}_t^{\sigma_m\top} \boldsymbol{v}_{y_t^m}$, where σ_l is the index of selected chunk at level l, but for *naive* potential function, the inequation above does not always hold. From this aspect, our method can also be considered as selecting the best path in the second-best search space.

Therefore, instead of following the original chunk orders, we propose to let each label y_j select the most similar chunk to it to obtain an emission score. We denote this definition as *max* potential function,

$$\phi\left(y_{t-1}^l, y_t^l, \mathcal{H}_t^l\right) = A_{y_{t-1}^l, y_t^l} + \max_{\boldsymbol{h} \in \mathcal{H}_t^l} \boldsymbol{h}^\top \boldsymbol{v}_{y_t^l} \quad (3)$$

In this case, we update chunk sets by removing these chunks which are selected by the target labels.

$$\mathcal{H}_t^{l+1} = \mathcal{H}_t^l \setminus \{\arg\max_{\boldsymbol{h} \in \mathcal{H}_t^l} \boldsymbol{h}^\top \boldsymbol{v}_{y_t^l}\} \quad (4)$$

Furthermore, since the log-sum-exp operation is a well known differentiable approximation of the max operation, we also introduce it as the third potential function,

$$\phi\left(y_{t-1}^l, y_t^l, \mathcal{H}_t^l\right) = A_{y_{t-1}^l, y_t^l} + \log \sum_{\boldsymbol{h} \in \mathcal{H}_t^l} \exp \boldsymbol{h}^\top \boldsymbol{v}_{y_t^l} \quad (5)$$

The chunk set is updated in the same way as Equation 4. We refer to this potential function definition as *logsumexp* in the rest of this paper.

2.4 Embedding Layer

Following previous work (Shibuya and Hovy, 2020), we convert words to word embeddings $\boldsymbol{w}_t \in \mathbb{R}^{d_w}$ and employ a character-level bidirectional LSTM to obtain character-based word embeddings $\boldsymbol{c}_t \in \mathbb{R}^{d_c}$. The concatenation of them is fed into the encoding layer as the token representation $\boldsymbol{x}_t = [\boldsymbol{w}_t, \boldsymbol{c}_t] \in \mathbb{R}^{d_x}$.

2.5 Encoding Layer

We employ a three-layered bidirectional LSTM to encode sentences and leverage contextual information,

$$\{\boldsymbol{h}_t\}_{t=1}^n = \mathrm{LSTM}\left(\{\boldsymbol{x}_t\}_{t=1}^n\right) \quad (6)$$

where $\boldsymbol{h}_t \in \mathbb{R}^{d_h}$ is the hidden state. In contrast to the encoders of previous work, which can only output single hidden states at each time step, we split $\boldsymbol{h}_t$ into m chunks,

$$[\boldsymbol{h}_t^1, \ldots, \boldsymbol{h}_t^m] = \boldsymbol{h}_t \quad (7)$$

where $\boldsymbol{h}_t^j \in \mathbb{R}^{d_h/m}$, and use them as the first level chunk set, i.e., $\mathcal{H}_t^1 = \{\boldsymbol{h}_t^j\}_{j=1}^m$, to start recognition.

2.6 Decoding Layer

At each level, we run a shared conventional CRF with its corresponding potential function $\phi\left(y_{t-1}^l, y_t^l, \mathcal{H}_t^l\right)$ and update the chunk sets until finishing all m levels. On the training stage, we remove chunks according to the selections of the target labels, while on the decoding stage, it depends on the selections of the predicted labels.

2.7 Training and Decoding

Following the definition of CRF, the conditional probabilistic function of a given label sequence at l-th level, i.e., $\boldsymbol{y}^l = \{y_t^l\}_{t=1}^n$, can be defined as,

$$p\left(\boldsymbol{y}^l \mid \mathcal{H}^l\right) = \frac{1}{Z(\mathcal{H}^l)} \exp \sum_{t=1}^{n} \phi\left(y_{t-1}^l, y_t^l, \mathcal{H}_t^l\right) \tag{8}$$

$$Z(\mathcal{H}^l) = \sum_{\boldsymbol{y}' \in \mathcal{Y}^n} \exp \sum_{t=1}^{n} \phi\left(y_{t-1}'^l, y_t'^l, \mathcal{H}_t^l\right) \tag{9}$$

where $Z(\mathcal{H}^l)$ is the sum of all paths' scores and is commonly known as the partition function.

We optimize our model by minimizing the sum of the negative log-likelihoods of all levels.

$$\mathcal{L} = -\sum_{l=1}^{m} \log p\left(\boldsymbol{y}^l \mid \mathcal{H}^l\right) \tag{10}$$

On the decoding stage, we iteratively apply the Viterbi algorithm (Forney, 1973) at each level to search the most probable label sequences.

$$\hat{\boldsymbol{y}}^l = \arg\max_{\boldsymbol{y}' \in \mathcal{Y}^n} p\left(\boldsymbol{y}' \mid \mathcal{H}^l\right) \tag{11}$$

The pseudocodes of the training and the decoding algorithms with *max* or *logsumexp* potential function can be found in Algorithms 1 and 2, respectively.

3 Experiments

3.1 Datasets

We conduct experiments on three nested named entity recognition datasets in English, i.e., ACE2004 (Doddington et al., 2004), ACE2005 (Walker et al., 2006) and GENIA (Kim et al., 2003). We divide all these datasets into tran/dev/test split by following Shibuya and Hovy (2020) and Wang et al. (2020). The dataset statistics can be found in Table 1.

Dataset	Sentences	Mentions	$\lvert\mathcal{Y}\rvert$	m
ACE2004	6,198 / 742 / 809	22,195 / 2,514 / 3,034	29	6
ACE2005	7,285 / 968 / 1,058	24,700 / 3,218 / 3,029	29	6
GENIA	15,022 / 1,669 / 1,855	47,006 / 4,461 / 5,596	21	4

Table 1: Sizes of the dataset shown in the train/dev/test split. $\lvert\mathcal{Y}\rvert$ is the size of the label set, m is the maximal depth of entity nesting.

3.2 Hyper-parameters Settings

For word embeddings initialization, we utilize 100-dimensional pre-trained GloVe (Pennington et al., 2014) for the ACE2004 and the ACE2005 datasets, and use 200-dimensional biomedical domain word embeddings[1] (Chiu et al., 2016) for the GENIA dataset. Moreover, we randomly initialize 30-dimensional vectors for character embeddings. The hidden state dimension of character-level LSTM d_c is 100, i.e., 50 in each direction, thus the dimension of token representation d_x is 200. We apply dropout (Srivastava et al., 2014) on token representations before feeding it into the encoder.

The hidden state dimension of the three-layered LSTM is 600 for ACE2004 and ACE2005, i.e., 300 in each direction, and 400 for GENIA. Choosing a different dimension is because the maximal depth of entity nesting m is different. We apply layer normalization (Ba et al., 2016) and dropout with 0.5 ratio after each bidirectional LSTM layer.

Different from Shibuya and Hovy (2020), we use only one CRF instead of employing different CRFs for different entity types. Besides, our CRF is also shared across levels, which means we learn and decode entities at all levels with the same CRF.

Our model is optimized by using stochastic gradient descent (SGD), with a decaying learning rate $\eta_\tau = \eta_0/(1+\gamma\cdot\tau)$, where τ is the index of the current epoch. For ACE2004, ACE2005, and GENIA, the initial learning rates η_0 are 0.2, 0.2, and 0.1, and the decay rates γ are 0.01, 0.02, and 0.02 respectively. We set the weight decay rate, the momentum, the batch size, and the number of epochs to be 10^{-8}, 0.5, 32, and 100 respectively, especially we use batch size 64 on the GENIA dataset. We clip the gradient exceeding 5.

Besides, we also conduct experiments to evaluate the performance of our model with contextual word representations. BERT (Devlin et al., 2019) and Flair (Akbik et al., 2018) are the most commonly used contextual word representations in previous work, and have also been proved that they can substantially improve the model performance. In these settings, contextual word representations are concatenated with word and character representations to form the token representations, i.e., $\boldsymbol{x}_t = [\boldsymbol{w}_t, \boldsymbol{c}_t, \boldsymbol{e}_t]$, where $\boldsymbol{e}_t$ is the contextual word representation and it is not fine-tuned in any of our experiments.

[1] https://github.com/cambridgeltl/BioNLP-2016

Methods	ACE2004			ACE2005			GENIA		
	P	R	F_1	P	R	F_1	P	R	F_1
Ju et al. (2018)				74.2	70.3	72.2	78.5	71.3	74.7
Wang et al. (2018)	74.9	71.8	73.3	74.5	71.5	73.0	78.0	70.2	73.9
Wang and Lu (2018)	78.0	72.4	75.1	76.8	72.3	74.5	77.0	73.3	75.1
Luo and Zhao (2020)				75.0	75.2	75.1	77.4	74.6	76.0
Lin et al. (2019)				76.2	73.6	74.9	75.8	73.9	74.8
Straková et al. (2019)	78.92	75.33	77.08	76.35	74.39	75.36	79.60	73.53	76.44
Shibuya and Hovy (2020)	79.93	75.10	77.44	78.27	75.44	76.83	78.70	75.74	77.19
Wang et al. (2020)	80.83	78.86	79.83	79.27	79.37	**79.32**	77.91	77.20	**77.55**
Our Method (naive)	81.12	77.71	79.38 (0.31)	79.45	77.22	78.32 (0.26)	78.83	75.32	77.03 (0.13)
Our Method (max)	81.90	78.05	79.92 (0.10)	80.68	77.03	78.81 (0.04)	78.80	75.71	77.22 (0.10)
Our Method (logsumexp)	81.24	78.96	**80.08** (0.22)	79.49	77.65	78.55 (0.12)	78.58	76.21	77.37 (0.15)
Straková et al. (2019) [B]	84.71	83.96	84.33	82.58	84.29	83.42	79.92	76.55	78.20
Shibuya and Hovy (2020) [B]	85.23	84.72	84.97	83.30	84.69	83.99	77.46	76.65	77.05
Wang et al. (2020) [B]	86.08	86.48	**86.28**	83.95	85.39	84.66	79.45	78.94	**79.19**
Our Method (naive) [B]	86.19	85.28	85.73 (0.24)	84.23	84.17	84.20 (0.30)	78.83	78.07	78.45 (0.32)
Our Method (max) [B]	86.27	85.09	85.68 (0.09)	85.28	84.15	**84.71** (0.09)	79.20	78.16	78.67 (0.18)
Our Method (logsumexp) [B]	86.42	85.71	86.06 (0.10)	83.95	84.67	84.30 (0.13)	78.83	78.27	78.54 (0.02)
Straková et al. (2019) [B+F]	84.51	84.29	84.40	83.48	85.21	84.33	80.11	76.60	78.31
Shibuya and Hovy (2020) [B+F]	85.94	85.69	85.82	83.83	84.87	84.34	77.81	76.94	77.36
Wang et al. (2020) [B+F]	87.01	86.55	**86.78**	84.90	86.08	**85.49**	79.98	78.51	**79.24**
Our Method (naive) [B+F]	86.56	85.65	86.11 (0.24)	84.17	84.88	84.52 (0.21)	79.28	78.31	78.79 (0.17)
Our Method (max) [B+F]	86.96	85.45	86.19 (0.17)	84.70	84.76	84.73 (0.21)	79.51	78.25	78.87 (0.04)
Our Method (logsumexp) [B+F]	86.74	86.11	86.42 (0.31)	84.81	85.06	84.93 (0.24)	79.20	78.67	78.93 (0.26)

Table 2: Experimental results on the ACE2004, ACE2005 and GENIA datasets. Labels [B] and [F] stand for BERT and Flair contextual word representations respectively. **Bold** and underlined numbers indicates the best and the second-best results respectively. *naive*, *max*, and *logsumexp* refer to the three potential function definitions, i.e., Equations 1, 3, and 5, respectively. These numbers in parentheses are standard deviations.

BERT is a transformer-based (Vaswani et al., 2017) pre-trained contextual word representation. In our experiments, for the ACE2004 and ACE2005 datasets we use the general domain checkpoint `bert-large-uncased`, and for the GENIA dataset we use the biomedical domain checkpoint `BioBERT large v1.1` [2] (Lee et al., 2019). We average all BERT subword embeddings in the last four layers to build 1024-dimensional vectors.

Flair is a character-level BiLSTM-based pre-trained contextual word representation. We concatenate these vectors obtained from the `news-forward` and `news-backward` checkpoints for ACE2004 and ACE2005, and use the `pubmed-forward` and `pubmed-backward` checkpoints for GENIA, to build 4096-dimensional vectors.

3.3 Evaluation

Experiments are all evaluated by precision, recall, and F_1. All of our experiments were run 4 times with different random seeds and averaged scores are reported in the following tables.

Our model [3] is implemented with `PyTorch` (Paszke et al., 2019) and we run experiments on `GeForce GTX 1080Ti` with 11 GB memory.

3.4 Experimental Results

Table 2 shows the performance of previous work and our model on the ACE2004, ACE2005, and GENIA datasets. Our model substantially outperforms most of the previous work, especially when comparing with our baseline Shibuya and Hovy (2020). When using only word embeddings and character-based word embeddings our method exceeds theirs by 2.64 F_1 score, and also achieves comparable results with the recent competitive method (Wang et al., 2020). In the case of utilizing BERT and further employing Flair, our method consistently outperforms Shibuya and Hovy (2020) by 1.09 and 0.60 by F_1 scores, respectively.

On the ACE2005 dataset, our method improves the F_1 scores by 1.98, 0.72, and 0.59 respectively, comparing with Shibuya and Hovy (2020). Although our model performance is inferior to Wang

[2] https://github.com/naver/biobert-pretrained

[3] https://github.com/speedcell4/nersted

et al. (2020) at general, our *max* potential function method is slightly superior to them by 0.05 in F_1 score when employing BERT.

Furthermore, on the biomedical domain dataset GENIA, our method constantly outperforms Shibuya and Hovy (2020) by 0.18, 1.62, and 1.57 in F_1 score, respectively. Although the low scores of Shibuya and Hovy (2020) are due to their usage of the general domain checkpoint `bert-large-uncased`, instead of our biomedical domain checkpoint, our model is still superior to Straková et al. (2019) by 0.47 and 0.62 in F_1 scores, who used the same checkpoint as us.

As for these three potential functions, we notice the *max* and *logsumexp* potential functions generally works better than the *naive* potential function. These results demonstrate that the chunk selection strategy of the *max* and *logsumexp* can leverage information from all remaining chunks and constrains hidden states of LSTM to be more semantically ordered. When we use BERT and Flair, the advantage of the *max* and the *logsumexp* potential function is less obvious compared with the case when we only use word embeddings and character-based word embeddings, especially on the GENIA dataset. We hypothesize that BERT and Flair can provide rich contextual information, then selecting chunks in the original order is sufficient, thus our dynamic selecting mechanism can only slightly improve the model performance.

3.5 Influence of the Encoding Scheme

We also conduct experiments on the ACE2004 dataset to measure the influence of the outermost-first and innermost-first encoding schemes. As shown in Table 3, the innermost-first encoding scheme consistently works better than the outermost-first encoding scheme with all potential functions. We hypothesize that outermost entities do not necessarily contain inner entities especially for longer ones, and that putting those diversely nested outermost entities at the same level would dislocate the encoding representation. Furthermore, even if we use the outermost-first encoding scheme, our method is superior to Shibuya and Hovy (2020), which further demonstrates the effectiveness of excluding the influence of the best path.

Encoding Scheme	ϕ	P	R	F_1
Outermost First	naive	79.08	76.57	77.80 (0.26)
	max	79.07	75.11	77.04 (0.20)
	logsumexp	79.05	76.39	77.70 (0.32)
Innermost First	naive	81.12	77.71	79.38 (0.31)
	max	81.90	78.05	79.92 (0.10)
	logsumexp	81.24	78.96	**80.08** (0.22)

Table 3: Influence of the two encoding schemes and the three potential functions.

3.6 Time Complexity and Speed

The time complexity of encoder is $\mathcal{O}(n)$, and because we employ the same tree reduction acceleration trick[4] as Rush (2020), the time complexity of CRF is reduced to $\mathcal{O}(\log n)$, therefore the overall time complexity is $\mathcal{O}(n + m \cdot \log n)$.

Even our model outperforms slightly worse than Wang et al. (2020), the training and inference speed of our model is much faster than them, as shown in Table 4, since we do not need to stack the decoding component to 16 layers. Especially, when we increase the batch size to 64, the decoding speed is more than two times faster than their model.

Method	Batch Size	Training	Decoding
Wang et al. (2020)	16	1,937.16	3,626.53
	32	3,632.64	4,652.05
	64	6,298.85	5,113.85
Our Method	16	4,106.03	3,761.03
	32	7,219.57	6,893.03
	64	10,584.80	11,652.92

Table 4: Speed comparison on the ACE2005 dataset. Numbers indicate how many words can be processed per second on average.

3.7 Level-wise Performance

We display the performance on the dataset ACE2005 at each level, as in Table 5. The *max* potential function at the first three levels achieves constantly higher precision scores than the *naive* and *logsumexp* potential functions, while at the same time obtains the lowest recall scores. The *logsumexp* potential function on the contrary achieves the highest recall scores but fails to obtain satisfactory precision scores. Because most entities are located at the first two levels, the *max* and *logsumexp* achieves the best overall precision and recall scores, respectively.

3.8 Chunk Distribution

We analyze the chunk distribution on the test split of the dataset ACE2005 by plotting the heat maps

[4] https://github.com/speedcell4/torchlatent

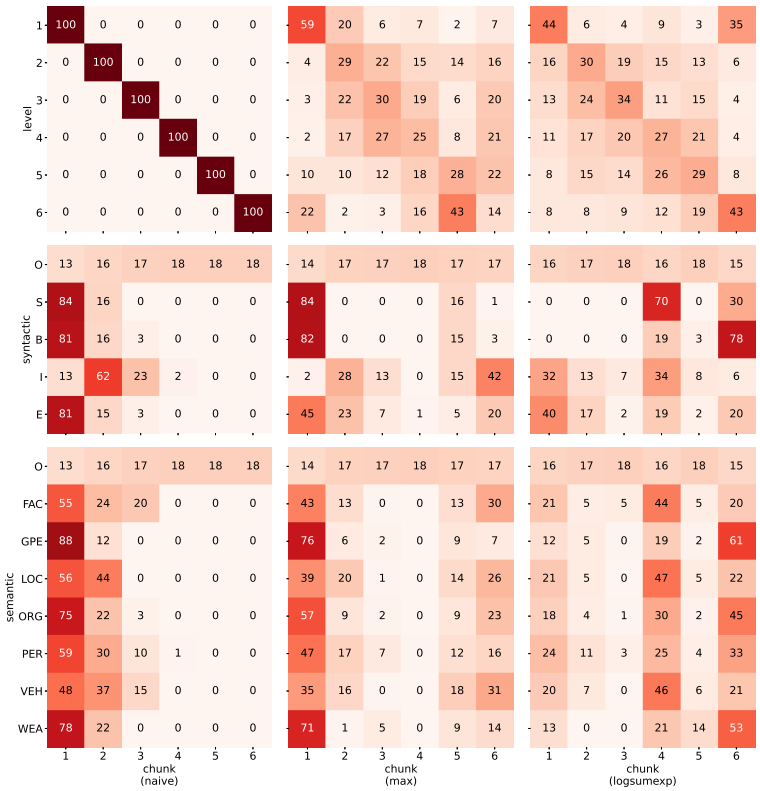

Figure 3: Chunk distributions of the *naive*, *max*, and *logsumexp* potential functions, respectively. Each row displays the chunk selection preferences with respect to levels, syntactic and semantic labels, respectively.

Level	Naive		Max		LogSumExp	
	P	R	P	R	P	R
1	80.83	80.12	**82.14**	79.51	80.98	80.12
2	73.91	68.67	**74.76**	70.76	73.85	70.76
3	60.09	48.80	**65.26**	49.10	60.17	53.01
4	**100.00**	16.67	37.50	10.42	66.67	14.58
5	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.00
Overall	79.45	77.22	**80.68**	77.03	79.49	77.65

Table 5: Precision and recall scores at each level with each potential functions.

in Figure 3, in which these numbers indicate the percentages of each chunk being selected by a particular level or label. For example, the 35 at the upper-right corner means when using *logsumexp* potential function, 35% of predictions at the first level are made by choosing the sixth chunk, while the 78 at the lower-left corner shows 78% of `WEA` are related to the first chunk with *naive*. To make it easier to compare with the *naive*, we arranged the chunk orders of *max* and *logsumexp*, without losing generality, to make the level-chunk distribution mainly concentrate on the diagonal.

The *naive* potential function simply selects the l-th chunk at l-th level, therefore the heat map is just diagonal. At the first level, the *logsumexp* potential function also prefers to select the sixth and the fourth chunks rather than the first chunk, we hypothesis this is due to most of `B-` and `S-` labels are located on the first level, and this can be confirmed according to the syntactic-chunk heat map of *logsumexp* where 78% `B-` and 70% `S-` labels go to the sixth and fourth chunks. Similarly, *max* also has a high probability to select the second chunk.

Generally, the chunk distribution of *logsumexp* is more smooth than *max*. Besides, we find label `O` almost uniformly select chunks, in both the syntactic and semantic heat maps, while other meaningful labels have their distinguished preferences.

Syntactic labels `S-` and `B-` mainly represent the beginning of an entity, while `I-` and `E-` stands for the continuation and ending of an entity. In the syntactic-chunk heat map of *naive*, they are indiscriminately distributed to the first chunk, because most of the entities are located on the first level. However, *max* and *logsumexp* utilize different chunks to represents these different syntactic categories.

Likewise, the semantic label `GPE`, when using *logsumexp*, also has a 61% probability to select the sixth chunks other than concentrating on the first

chunk as *naive*. These observations further demonstrate our dynamic chunk selection strategies are capable of learning more meaningful representations.

4 Related Work

Existing NER algorithms commonly employ various neural networks to leverage more morphological and contextual information to improve performance. For example, to handle the out-of-vocabulary issue through introducing morphological features, Huang et al. (2015) proposed to employ manual spelling feature, while Ma and Hovy (2016) and Lample et al. (2016) suggested introducing CNN and LSTM to build word representations from character-level. Zhang et al. (2018) and Chen et al. (2019) introduced global representation to enhance encoder capability of encoding contextual information.

Layered Model As a layered model, Ju et al. (2018) dynamically update span-level representations for next layer recognition according to recognized inner entities. Fisher and Vlachos (2019) proposed a merge and label method to enhance this idea further. Recently, Shibuya and Hovy (2020) designed a novel algorithm to efficiently learn and decode the second-best path on the span of detected entities. Luo and Zhao (2020) build two different graphs, one is the original token sequence, and the other is the tokens in recognized entities, to model the interaction among them. Wang et al. (2020) proposed to learn the l-gram representations at layer l through applying a decoder component to reduce a sentence layer by layer and to directly classify these l-gram spans.

Region-based Model Lin et al. (2019) proposed an anchor-region network to recognize nested entities through detecting anchor words and entity boundaries first, and then classify each detected span. Exhaustive models simply enumerate all possible spans and utilize a maximum entropy tagger (Byrne, 2007) and neural networks (Xu et al., 2017; Sohrab and Miwa, 2018; Zheng et al., 2019) for classification. Luan et al. (2019) additionally aims to consider the relationship among entities and proposed a novel method to jointly learn both entities and relations.

Hypergraph-based Model Lu and Roth (2015) proposed a hyper-graph structure, in which edges are connected to multiple nodes to represents nested entities. Muis and Lu (2017) and Wang and Lu (2018) resolved spurious structures and ambiguous issue of hyper-graph structure. And Katiyar and Cardie (2018) proposed another kind of hyper-graph structure.

Parsing-based Model Finkel and Manning (2009) indicated all these nested entities are located in some non-terminal nodes of the constituency parses of the original sentences, thus they proposed to use a CRF-based constituency parser to obtain them. However, the cubic time complexity limits its applicability. Wang et al. (2018) instead proposed to use a transition-based constituency parser to incrementally build constituency forest, its linear time complexity ensures it can handle longer sentences.

5 Conclusion

In this paper, we proposed a simple and effective method for nested named entity recognition by explicitly excluding the influence of the best path through selecting and removing chunks at each level to build different potential functions. We also proposed three different selection strategies to leverage information from all remaining chunks. Besides, we found the innermost-first encoding scheme works better than the conventional outermost-first encoding scheme. Extensive experimental results demonstrate the effectiveness and efficiency of our method. However, one of the demerits of our method is the number of chunks, i.e., the maximal depth of entity nesting, must be chosen in advance as a hyper-parameter. We will extend it to arbitrary depths as future work.

Acknowledgements

This work was partly supported by JST CREST Grant Number JPMJCR1513. The authors would like to thank the anonymous reviewers for their instructive comments.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

K. Byrne. 2007. Nested named entity recognition in historical archive text. In *International Conference on Semantic Computing (ICSC 2007)*, pages 589–596.

Hui Chen, Zijia Lin, Guiguang Ding, Jianguang Lou, Yusen Zhang, and Borje Karlsson. 2019. GRN: Gated relation network to enhance convolutional neural network for named entity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6236–6243.

Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. 2016. How to train good word embeddings for biomedical NLP. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, pages 166–174, Berlin, Germany. Association for Computational Linguistics.

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.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Jenny Rose Finkel and Christopher D. Manning. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 141–150, Singapore. Association for Computational Linguistics.

Joseph Fisher and Andreas Vlachos. 2019. Merge and label: A novel neural network architecture for nested NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5840–5850, Florence, Italy. Association for Computational Linguistics.

G David Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

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.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. 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 861–871, New Orleans, Louisiana. Association for Computational Linguistics.

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

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 Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *CoRR*, abs/1901.08746.

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.

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.

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.

Ying Luo and Hai Zhao. 2020. Bipartite flat-graph network for nested named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6408–6418, Online. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Aldrian Obaja Muis and Wei Lu. 2017. Labeling gaps between words: Recognizing overlapping mentions with mention separators. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2608–2618, Copenhagen, Denmark. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Alexander Rush. 2020. Torch-struct: Deep structured prediction library. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 335–342, Online. Association for Computational Linguistics.

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

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. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57:45.

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.

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.

Jue Wang, Lidan Shou, Ke Chen, and Gang Chen. 2020. Pyramid: A layered model for nested named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5918–5928, Online. Association for Computational Linguistics.

Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. 2017. A local detection approach for named entity recognition and mention detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1237–1247, Vancouver, Canada. Association for Computational Linguistics.

Yue Zhang, Qi Liu, and Linfeng Song. 2018. Sentence-state LSTM for text representation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 317–327, Melbourne, Australia. Association for Computational Linguistics.

Changmeng Zheng, Yi Cai, Jingyun Xu, Ho-fung Leung, and Guandong Xu. 2019. 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. Association for Computational Linguistics.