Bottom-Up Constituency Parsing and Nested Named Entity Recognition with Pointer Networks

Songlin Yang, Kewei Tu*
School of Information Science and Technology, ShanghaiTech University
Shanghai Engineering Research Center of Intelligent Vision and Imaging
{yangsl,tukw}@shanghaitech.edu.cn

Abstract

Constituency parsing and nested named entity recognition (NER) are similar tasks since they both aim to predict a collection of nested and non-crossing spans. In this work, we cast nested NER to constituency parsing and propose a novel pointing mechanism for bottom-up parsing to tackle both tasks. The key idea is based on the observation that if we traverse a constituency tree in post-order, i.e., visiting a parent after its children, then two consecutively visited spans would share a boundary. Our model tracks the shared boundaries and predicts the next boundary at each step by leveraging a pointer network. As a result, it needs only linear steps to parse and thus is efficient. It also maintains a parsing configuration for structural consistency, i.e., always outputting valid trees. Experimentally, our model achieves the state-of-the-art performance on PTB among all BERT-based models (96.01 F1 score) and competitive performance on CTB7 in constituency parsing; and it also achieves strong performance on three benchmark datasets of nested NER: ACE2004, ACE2005, and GENIA [1].

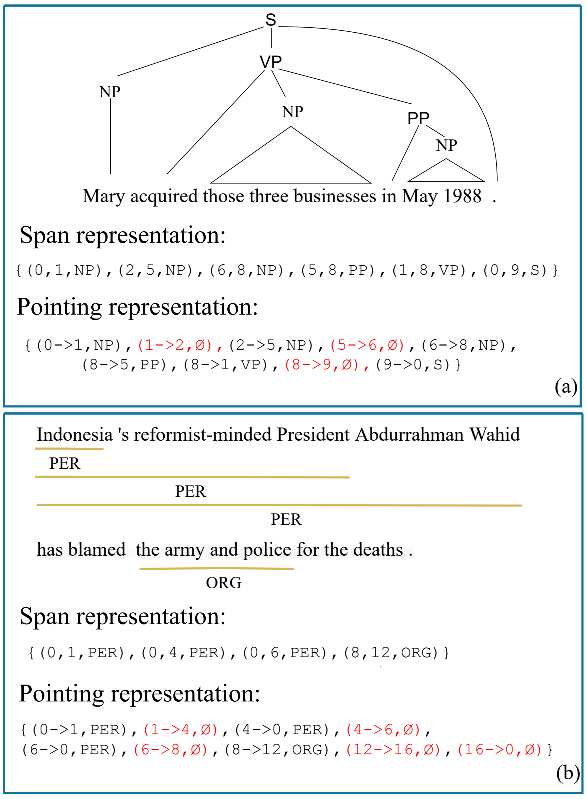

S
VP
NP
NP
PP
NP
Mary acquired those three businesses in May 1988 .
Span representation:
{(0,1,NP),(2,5,NP),(6,8,NP),(5,8,PP),(1,8,VP),(0,9,S)}
Pointing representation:
{(0->1,NP),(1->2,∅),(2->5,NP),(5->6,∅),(6->8,NP),
(8->5,PP),(8->1,VP),(8->9,∅),(9->0,S)}
(a)
Indonesia 's reformist-minded President Abdurrahman Wahid
PER
PER
PER
has blamed the army and police for the deaths .
ORG
Span representation:
{(0,1,PER),(0,4,PER),(0,6,PER),(8,12,ORG)}
Pointing representation:
{(0->1,PER),(1->4,∅),(4->0,PER),(4->6,∅),
(6->0,PER),(6->8,∅),(8->12,ORG),(12->16,∅),(16->0,∅)}
(b)

Figure 1: (a) an example non-binary constituency tree. (b) an example sentence with nested named entities. We show the span and pointing representations.

1 Introduction

Constituency parsing is an important task in natural language processing, having many applications in downstream tasks, such as semantic role labeling (Fei et al., 2021), opinion mining (Xia et al., 2021), among others. Named entity recognition (NER) is a fundamental task in information extraction and nested NER has been receiving increasing attention due to its broader applications (Byrne, 2007).

Constituency parsing and nested NER are similar tasks since they both aim to predict a collection of nested and non-crossing spans (i.e., if two spans overlap, one must be a subspan of the other). Fig.1 shows example span representations of both tasks. The difference between the two tasks is that the collection of spans form a connected tree in constituency parsing, whereas they form several tree fragments in nested NER. However, we can add a node that spans the whole sentence to connect all tree fragments in nested NER to form a tree. Because of the similarity, there are some previous studies adapting methods from the constituency parsing literature to tackle nested NER (Finkel and Manning, 2009; Wang et al., 2018; Fu et al., 2021). In this work, we focus on constituency parsing, but our proposed method tackles nested NER as well.

The two main paradigms in constituency pars-

*Corresponding Author

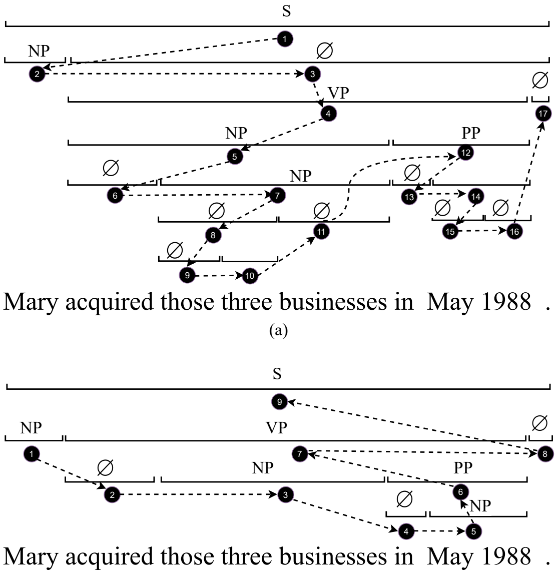
[1] Our code is publicly available at https://github.com/sustcsonglin/pointer-net-for-nested

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 2403 - 2416
May 22-27, 2022 ©2022 Association for Computational Linguistics

ing are span-based and transition-based methods. Span-based methods (Stern et al., 2017; Kitaev and Klein, 2018; Zhang et al., 2020; Xin et al., 2021, *inter alia)* decompose the score of a constituency tree into the scores of constituent spans and use chart-based algorithms for inference. Built upon powerful neural encoders, they have obtained state-of-the-art results. However, they suffer from the high inference time complexity of exact algorithms or error propagation of top-down approximate algorithms. In contrast, transition-based methods (Dyer et al., 2016; Cross and Huang, 2016; Liu and Zhang, 2017, *inter alia)* conduct a series of local actions (e.g., shift and reduce) to build the final parse in linear steps, so they enjoy lower parsing time complexities. However, they suffer from the error propagation and exposure bias problems.

Recently, Nguyen et al. (2021a) propose a sequence-to-sequence (seq2seq) model with pointer networks (Vinyals et al., 2015a). They cast constituency parsing to a top-down splitting problem. First, they use neural encoders to obtain span representations, similar to span-based methods. Then they feed input parent span representations into the neural decoder recursively following the order shown in Fig. 2(a)[2]—which amounts to pre-order traversal—to output a series of splitting points (i.e., boundaries) via pointer networks, so that each parent span is split into two child spans. Notably, Nguyen et al. (2020) propose a similar top-down pointing mechanism, but they design a chart-based parsing algorithm instead of adopting seq2seq modeling, and has been shown underperforming Nguyen et al. (2021a). Thanks to seq2seq modeling, Nguyen et al. (2021a)'s model achieves a competitive parsing performance with a lower parsing complexity compared with span-based methods.

However, their model has two main limitations. First, when generating each constituent, its subtree features cannot be exploited since its subspans have not been realized yet (Liu and Zhang, 2017). Thus it is difficult for the model to predict the splitting point of a long span due to a lack of its subtree information, which exacerbates the error propagation problem and undermines the parsing performance. Second, since each parent span can only be split into two, their parsing algorithm can only ouput binary trees, thus needing binarization.


S
NP
∅
VP
NP
PP
∅
NP
∅
∅
∅
∅
∅
∅
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
Mary acquired those three businesses in May 1988 .
S
NP
VP
∅
∅
NP
PP
∅
NP
1 2 3 4 5 6 7 8 9
Mary acquired those three businesses in May 1988 .

(a)

(b)

Figure 2: Illustration of pre-order and post-order traversal over the constituency tree shown in Figure 1(a). (a): pre-order traversal. (b): post-order traversal. We mark the generation order in the circles below spans and link two consecutively visited constituents by arrows. Note that in (a), binarization is assumed.

In this work, we devise a novel pointing mechanism for *bottom-up* parsing using (almost) the same seq2seq backbone as Nguyen et al. (2021a). Our model is able to overcome the two aforementioned limitations of Nguyen et al. (2021a). The main idea is based on the observation that if we traverse a constituency tree in post-order (i.e., visiting a parent after its children), two consecutively visited constituent spans would share a boundary. Fig. 2(b) shows an example: the right boundary of ❶ is also the left boundary of ❷ and the right boundary of ❺ is also the right boundary of ❻. Based on this observation, we propose to use a cursor to track the shared boundary boundaries and at each step, leverage a pointer network to predict the next boundary for generating the next constituent span and update the cursor to the right boundary of the new span. Our model generates one span at each step, thus needing only linear steps to parse a sentence, which is efficient. Besides, our model can leverage rich subtree features encoded in the neural decoder to generate parent constituent spans, which is especially helpful in predicting long spans. Finally, our model can output n-ary trees, enabling direct modeling of the original non-binary parse tree structures in treebanks and eliminating the need for binarization.

[2] Slightly different from the figure, they do not feed spans of length 1 into the decoder for obvious reasons.

We conduct experiments on the benchmarking PTB and CTB for constituency parsing. On PTB, we achieve the state-of-the-art performance (96.01 F1 score) among all BERT-based models. On CTB, we achieve competitive performance. We also apply our method to nested NER and conduct experiments on three benchmark datasets: ACE2004, ACE2005, and GENIA. Our method achieves comparable performance to many tailored methods of nested NER, beating previous parsing-based methods. Our contributions can be summarized as the following:

- We propose a novel pointing mechanism for bottom-up n-ary tree parsing in linear steps.

- Our model achieves the state-of-the-art result on PTB in constituency parsing. We further show its application in nested NER where it achieves competitive results.

2 Methods

2.1 Preprocessing

It is known that constituency parsing can be regarded as a top-down splitting problem where parent spans are recursively split into pairs of subspans (Stern et al., 2017; Shen et al., 2018; Nguyen et al., 2020, 2021a). However, this formulation can output binary trees. We make an extension to cast constituency parsing as top-down segmentation, i.e., parent spans are segmented into ≥ 2 subspans recursively, for the sake of outputting n-ary trees. To this end, we add some $\emptyset$ spans (we do not allow two adjacent $\emptyset$ spans to eliminate ambiguities) so that each span is either a bottommost span or can be segmented by its subspans. For instance, in Fig 2, ❸ is a bottom-most span, and ❼ can be segmented by ❷, ❸ and ❻. We always include the whole-sentence span in order to cast other tasks, e.g., nested NER, to constituency parsing. We also collapse unary chains to atomic labels in constituency parsing, e.g., `S->VP` → `S+VP`.

2.2 Parsing configuration

A problem of seq2seq constituency parsers is how to maintain structural consistency, i.e., outputting valid trees. To solve this problem, our pointing system maintains a *parsing configuration*, which is a quadruple (c, A, p, S) where:

- c: index of the cursor.

- A: set of indices of all candidate boundaries.

- p: the left boundary of the lastly created span, which is needed to maintain A.

- S: set of generated spans.

We can see from Fig. 3 that in the beginning, the cursor c lies at 0. At each step, c points to another boundary a from A to form a span $(\min(c, a), \max(c, a))$. There are two cases:

- $c < a$: a new bottom-most span is generated.

- $a < c$: several consecutive spans are merged into a larger span. It is worthy to note that we can merge $>= 2$ spans in a single step, which allows our model to perform n-ary tree parsing.

In the first case, the new bottom-most span can combine with the very previous span to form a larger span whose left boundary is p, so we push p back to A (except for the case that $p =$ `null`). In the later case, the very previous span is a subspan of the new span and thus p cannot be pushed back. In both cases, all indices $\min(c, a) \leq i < \max(c, a)$ are removed from A due to the post-order generation restriction; p is updated to $\min(c, a)$ and c is updated to $\max(c, a)$. The process stops when the whole-sentence span is generated. Table 1 formalises this process.

Oracle. The oracle pointing representations shown in Fig.1 can be generated by running a post-order traversal of the tree (e.g., Fig.2) and for each traversed span, pointing the cursor from its boundary shared with the previous span to its other boundary. If we do not allow two consecutive $\emptyset$ spans, the oracle is unique under our pointing system (we give a proof in Appendix A.1 by contradiction).

2.3 Model

Given a sentence $w = w_1, ..., x_n$, we add `<bos>` (beginning of sentence) as w_0 and `<eos>` (end of sentence) as w_{n+1}. The oracle is $\{q_i \rightarrow p_i, y_i\}_{i=1,...,m}$, where y_i is the span label and we use $l_i = \min(q_i, p_i)$ and $r_i = \max(q_i, p_i)$ to denote the left and right boundary of the i-th span, respectively.

Encoder. We feed the sentence into BERT (Devlin et al., 2019) and for each word w_i, we use the last subtoken emebedding of the last layer as its dense representations x_i. Then we feed $x_0, \ldots, x_{n+1}$ into a three-layer bidirectional

Initial configuration	$(c, A, p, s) = (0, \{1, 2, \ldots, n\}, \texttt{null}, \emptyset)$		
Goal	$(0, n) \in S$		
Pointing action	Input	Output	Precondition
LEFT-POINT-a	(c, A, p, S)	$\Rightarrow (c, A \setminus \{a, \ldots, c-1\}, a, S \cup \{(a, c)\})$	$0 \leq a < c$
RIGHT-POINT-a	(c, A, p, S)	$\Rightarrow (a, A \cup \{p\} \setminus \{c, \ldots, a-1\}, c, S \cup \{(c, a)\})$	$c < a \leq n,$

Table 1: Description of the parsing configuration.

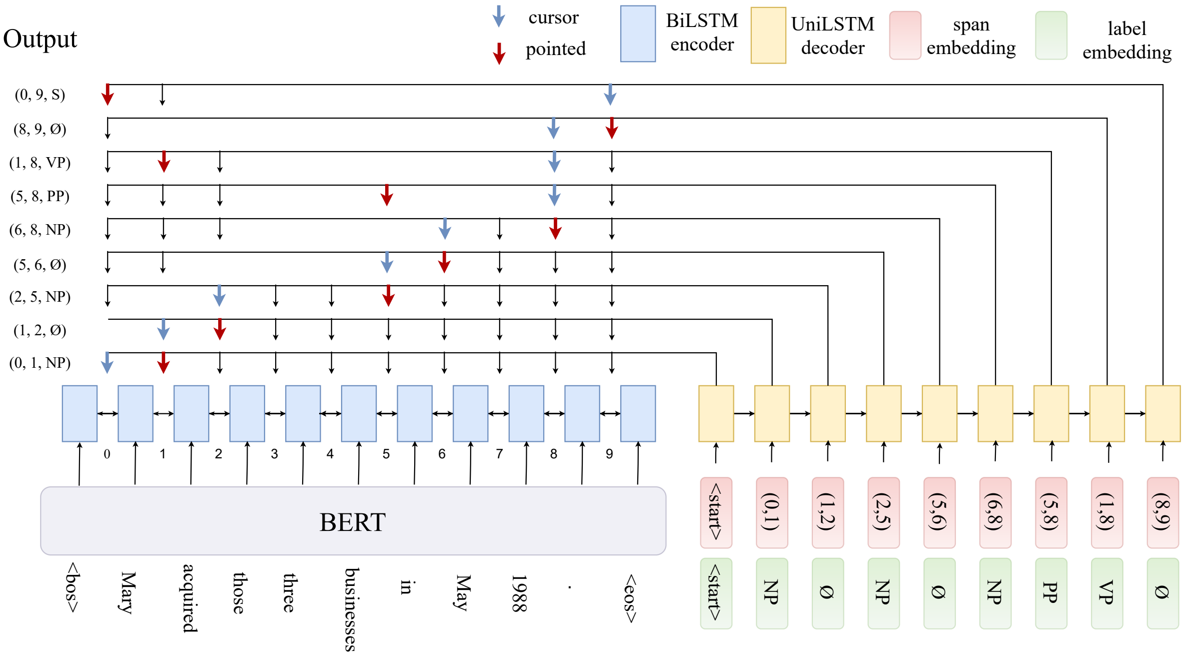


Figure 3: Demonstration of the generation process and the neural architecture. Black arrows point to candidate boundaries that are not selected in each step.

LSTM (Hochreiter and Schmidhuber, 1997) (BiLSTM) to obtain $c_0, \ldots, c_{n+1}$, where $c_i = [f_i; g_i]$ and f_i and g_i are the forward and backward hidden states of the last BiLSTM layer at position i respectively.

Boundary and span representation. We use fencepost representation (Cross and Huang, 2016; Stern et al., 2017) to encode the i-th boundary lying between x_i and x_{i+1}:

$$b_i = [f_i; g_{i+1}]$$

then we represent span (i, j) as:

$$h_{i,j} = \text{MLP}_{\text{span}}(b_j - b_i)$$

Decoder. We use a unidirectional one-layer LSTM network as the decoder:

$$d_t = \text{LSTM}(d_{t-1}, h_{l_{t-1}, r_{t-1}}; E_{y_{t-1}}), t \geq 2 \quad (1)$$

where d_t is the hidden state of the LSTM decoder at time step t, E is the label embedding matrix, ; is the concatenation operation. For the first step, we feed a randomly initialized trainable vector d_0 and a special <START> embedding into the decoder to obtain d_1.

Pointing score. We use a deep biaffine function (Dozat and Manning, 2017) to estimate the pointing score s_i^t of selecting the i-th boundary at time step t:

$$d'_t = \text{MLP}_{\text{cursor}}(d_t)$$
$$b'_i = \text{MLP}_{\text{point}}(b_i)$$
$$s_i^t = \left[b'_i; 1\right]^\top W_{\text{point}} d'_t$$

where $\text{MLP}_{\text{cursor}}$ and $\text{MLP}_{\text{point}}$ are multi-layer perceptrons (MLPs) that project decoder states and boundary representations into k-dimensional spaces, respectively; $W_{\text{point}} \in \mathcal{R}^{(k+1)\times(k)}$.

Label score. For a newly predicted span, we feed the concatenation of the span representation and the decoder state into another MLP to calculate the label score e^t:

$$H = \text{MLP}_{\text{label}}([d^t; b_{r_t} - b_{l_t}])$$
$$e^t = HE^T$$

Note that we reuse the label embedding matrix from Eq. 1 to facilitate parameter sharing.

Training objective. The training loss is decomposed into the pointing loss and the labeling loss:

$$L = L_{\text{pointing}} + L_{\text{labeling}}$$
$$L_{\text{pointing}} = -\sum_{t=1}^{m} \log \frac{\exp\{s^t_{p_t}\}}{\sum_{j=0}^{n} \exp\{s^t_j\}}$$
$$L_{\text{labeling}} = -\sum_{t=1}^{m} \log \frac{\exp\{e^t_{y_t}\}}{\sum_{j=1}^{|L|} \exp\{e^t_j\}}$$

where $|L|$ is the number of labels. Note that in the pointing loss we normalize over all boundaries instead of only accessible boundaries, because we find it performs better in our preliminary experiments.

Parsing. Our model follows the description in the previous subsection for parsing. For each time step t, it selects the highest-scoring accessible boundary to generate the span, then selects the highest-scoring label of the generated span, and updates the parsing configuration (Table 1).

3 Experiment setup

3.1 Data setup

Constituency parsing. We conduct experiments on Penn Treebank (PTB) 3.0 (Marcus et al., 1993) and Chinese Treebank (CTB) (Xue et al., 2005). Many previous researchers report that the results on CTB5.1 are unstable and of high variance (Zhang et al., 2020; Yang and Deng, 2020). So we follow the suggestion of Zhang et al. (2020) to conduct experiments on CTB7 instead of CTB5.1 for more robust evaluation as CTB7 has more test sentences and has a higher annotation quality. We use the standard data splits for both PTB and CTB.

Nested NER. We conduct experiments on three benchmark datasets: ACE2004 (Doddington et al., 2004), ACE2005 (Walker et al., 2006), and GENIA (Kim et al., 2003). We use the same data preprocessing as Shibuya and Hovy (2020) [3].

3.2 Evaluation

We report labeled recall/precision/F1 scores based on EVALB [4] for constituency parsing; span-level labeled recall/precision/F1 scores for nested NER. All reported results are averaged over three runs with different random seeds.

3.3 Implementation details

We use "bert-large-cased" (Devlin et al., 2019) for PTB, ACE2004 and ACE2005; "bert-chinese-based" for CTB; and "biobert-large-cased-v1.1" (Lee et al., 2020) for GENIA. We use no other external resources (e.g., predicted/gold POS tags, external static word embedding). The hidden size of LSTM is set to 1000 for both the encoder and the decoder. We add dropouts in LSTM/MLP layers. The dropout rate is set to 0.33. The hidden and output sizes of all MLPs are set to 500. The value of gradient clipping is set to 5. The number of training epochs is set to 10 for PTB, CTB, GENIA; 50 for ACE2004/2005. We use Adam (Kingma and Ba, 2015) as the optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.9$. The maximal learning rate is set to $5e-5$ for BERT and $2.5e-3$ for all other components. We use the first 10% epochs to linearly warmup the learning rates of each components to their maximum value and gradually decay them to zero for the rest of epochs. We batch sentences of similar lengths to make full use of GPUs and the number of tokens in a single batch is set to 3000.

4 Main result

On both PTB and CTB, we find incorporating $E_{y_{t-1}}$ in Eq. 1 leads to a slightly inferior performance (-0.02 F1 score on PTB and -0.05 F1 score on CTB), so we report results without this input feature.

Table 2 shows the results on PTB test set. Our method achieves 96.01 F1 score, outperforming the method of Nguyen et al. (2021a) by 0.31 F1 and having the same worst-case $O(n^2)$ parsing time complexity as theirs [5]. It also outperforms all span-

[3] https://github.com/yahshibu/nested-ner-tacl2020-transformers

[4] https://nlp.cs.nyu.edu/evalb

[5] In their paper, they claim an $O(n)$ time complexity, which treats the complexity of a single pointing operation as O(1). This calculation, however, assumes full GPU parallelization. Without parallelization, their method has a worst-case $O(n^2)$ time complexity as ours.

Model	P	R	F
Kitaev et al. (2019) [S]	95.46	95.73	95.59
Zhou and Zhao (2019) [S]	95.70	95.98	95.84
Zhang et al. (2020) [S]	95.85	95.53	95.69
Yang and Deng (2020) [T]	96.04	95.55	95.79
Nguyen et al. (2020) [S]	-	-	95.48
Wei et al. (2020) [S]	95.5	96.1	95.8
Tian et al. (2020) [S]	96.09	95.62	95.86
Xin et al. (2021) [S]	**96.29**	95.55	95.92
Nguyen et al. (2021a) [Q]	-	-	95.7
Cui et al. (2021) [S]	95.70	**96.14**	95.92
Ours [Q]	96.19	95.83	**96.01**

Table 2: Results on PTB. All models use BERT as encoders. S: span-based methods. T: transition-based methods. Q: seq2seq-based methods. P: labeled precision. R: labeled recall. F: labeled F1.

Model	P	R	F
Zhang et al. (2020) [S]	**91.73**	**91.38**	**91.55**
Ours [Q]	91.66	91.31	91.49

Table 3: Results on CTB7. All models use BERT as encoders.

based methods, obtaining the state-of-the-art performance among all BERT-based models while enjoying a lower parsing complexity.

Table 3 shows the results on CTB7. Our method obtains 91.49 F1 score, which is comparable to the method of Zhang et al. (2020) but has a lower complexity (worst-case $O(n^2)$ vs. $O(n^3)$).

Table 4 shows the results on three benchmark dataset on nested NER. We find that incorporating $E_{y_{t-1}}$ is important, leading to +0.67 F1 score and +0.52 F1 sore on ACE2004 and ACE2005, respectively. Although our method underperforms two recent state-of-the-art methods: Shen et al. (2021) and Tan et al. (2021), we find it has a competitive performance to other recent works (Wang et al., 2021; Yan et al., 2021; Fu et al., 2021). The most comparable one is the method of Fu et al. (2021), which belongs to parsing-based methods as ours. They adapt a span-based constituency parser to tackle nested NER using the CYK algorithm for training and inference. Our model outperforms theirs by 0.34 F1 and 0.13 F1 scores on ACE2004 and ACE2005 and has a similar performance to theirs on GENIA, meanwhile enjoying a lower inference complexity.

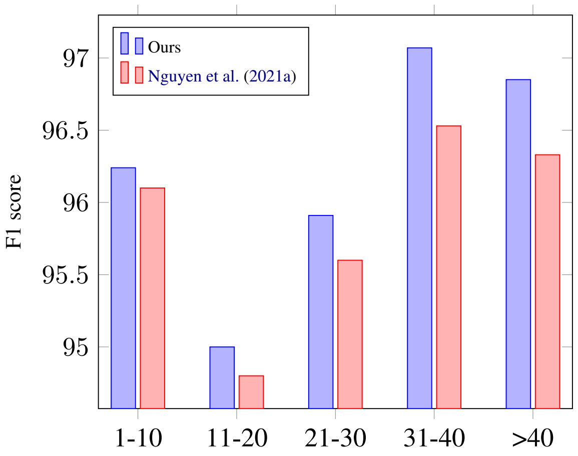


Figure 4: F1 scores against constituent span length on PTB test set.

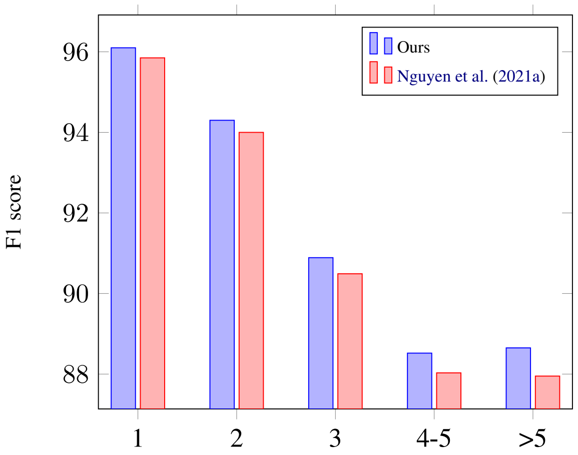


Figure 5: F1 scores on constituent nodes with different numbers of children on PTB test set.

5 Analysis

Error analysis. As we discussed previously, bottom-up parsing can make use of the subtree features when predicting parent spans, so it is expected to have higher F1 scores on longer spans. To verify this, we plot Fig. 4 to show the changes of F1 scores with different constituent span lengths on the PTB test set. We can see that our method consistently outperforms the method of (Nguyen et al., 2021a) on all span lengths, but our advantage is most prominent for spans of length >30, which verifies our conjecture. In Fig. 5, we can see that when a constituent has multiple children (>3), our method performs much better than that of (Nguyen et al., 2021a), which validates the benefit of *n-ary tree parsing*. An intuitive explanation of this benefit is that our method predicts n-ary branching structures in a single step, whereas theirs needs multiple steps, which is more error-prone.

Model	ACE2004			ACE2005			GENIA		
	P	R	F	P	R	F	P	R	F
Shibuya and Hovy (2020)	84.71	83.96	84.33	82.58	84.29	83.42	79.92	76.55	78.20
Wang et al. (2020)	86.08	86.48	86.26	83.95	85.39	84.66	79.45	78.94	79.19
Wang et al. (2021)	86.27	85.09	85.68	85.28	84.15	84.71	79.20	78.16	78.67
Fu et al. (2021)	86.7	86.5	86.6	84.5	86.4	85.4	78.2	78.2	78.2
Xu et al. (2021)	86.9	85.8	86.3	85.7	85.2	85.4	80.3	78.9	79.6
Yan et al. (2021)	87.27	86.41	86.84	83.16	86.38	84.74	78.57	79.3	78.93
Shen et al. (2021)	87.44	87.38	87.41	86.09	**87.27**	86.67	**80.19**	**80.89**	**80.54**
Tan et al. (2021)	88.46	86.10	87.26	**87.48**	86.64	**87.05**	82.31	78.66	80.44
Ours	86.60	87.28	86.94	84.61	86.43	85.53	78.08	78.26	78.16
w.o. $E_{y_{t-1}}$ in Eq.1	85.66	86.88	86.27	83.75	86.31	85.01	78.46	77.97	78.22

Table 4: Results on ACE2004, ACE2005 and GENIA. All models use BERT as encoders.

Effect of beam search. We also tried beam search but observed very slight improvement or even worse performance (e.g., +0.05 F1 score on PTB and -0.03 F1 score on CTB when we use a beam size 20). Hence we report all results using greedy decoding for simplicity. This suggests that greedy decoding can yield near-optimal solutions, indicating that our model is less prone to the error propagation problem.

Effect of training loss. As discussed in Sec. 2.3, we find that explicitly considering the structural consistency constraints when normalizing is harmful (-0.12 F1 score on PTB, -0.10 F1 score on CTB). We speculate that not enforcing the constraints during training can help the model to learn the constraints implicitly, which is helpful for the model to generalize better on the unseen test set. Notably, Nguyen et al. (2021a) also adopt this strategy, i.e., normalizing over all boundaries.

Speed. Similar to Nguyen et al. (2021a), the training process (i.e., teacher forcing) can be fully parallelized without resorting to structured inference, which could be compute-intensive or hard to parallelize. On PTB, it takes only 4.5 hours to train the model using BERT as the encoder with a single Titan V GPU. As for parsing, our method has the same parsing complexity as Nguyen et al. (2021a), i.e., worst-case $O(n^2)$. Table 5 shows the speed comparison on parsing the PTB test set (we report values based on a single Titan V GPU and not using BERT as encoder following Nguyen et al. (2021a)). We report the average number of pointing actions in Appendix A.2.

System	Speed (Sents/s)	Speedup
Petrov and Klein (2007) (Berkeley)	6	1.0x
Zhu et al. (2013)(ZPar)	90	15.0x
Stern et al. (2017)	76	12.7x
Shen et al. (2018)	111	18.5x
Nguyen et al. (2020)	130	21.7x
Zhou and Zhao (2019)	159	26.5x
Wei et al. (2020)	220	36.7x
Gómez-Rodríguez and Vilares (2018)	780	130x
Kitaev and Klein (2018) (GPU)	830	138.3x
Zhang et al. (2020)	924	154x
Nguyen et al. (2021a)	1127	187.3x
Ours	855	142.5x

Table 5: Speed comparison.

6 Related Work

Constituency parsing. There are many methods to tackle constituency parsing, such as transition-based methods (Dyer et al., 2016; Cross and Huang, 2016; Liu and Zhang, 2017; Yang and Deng, 2020), span-based methods (Stern et al., 2017; Kitaev and Klein, 2018; Kitaev et al., 2019; Zhang et al., 2020; Wei et al., 2020; Nguyen et al., 2020; Xin et al., 2021), sequence-to-sequence (seq2seq)-based methods (Vinyals et al., 2015b; Fernández-González and Gómez-Rodríguez, 2020), sequence-labeling-based methods (Gómez-Rodríguez and Vilares, 2018; Vilares et al., 2019; Kitaev and Klein, 2020), among others.

Our work belongs to the category of seq2seq-based methods. Previous seq2seq models linearize constituency trees into bracket sequences (Vinyals et al., 2015b) or shift-reduce action sequences (Ma et al., 2017; Fernández-González and Gómez-Rodríguez, 2020). However, they may produce invalid outputs and their performance lags behind span-based methods. Recently, seq2seq models linearize constituency trees into sequences of spans in pre-order (Nguyen et al., 2021a) or in in-order (Wei et al., 2021). Our method generates sequences of spans in post-order instead, which has the advantage of utilizing rich subtree features and performing direct n-ary tree parsing.

Binarization is *de facto* in constituency parsing, but there is a recent trend toward n-ary parsing. Previous span-based methods adopt either explicit binarization (Zhang et al., 2020) or implicit binarization (Stern et al., 2017; Kitaev and Klein, 2018). Although the implicit binarization strategy eliminates the need for binarization in training, it can only output binary trees during decoding. Xin et al. (2021) propose an n-ary-aware span-based method by defining semi-Markov processes on each parent span so that the transition scores of adjacent sibling child-spans are explicitly considered in parsing. Fernández-González and Gómez-Rodríguez (2019); Yang and Deng (2020) propose novel transition systems to model n-ary trees. Our method outputs n-ary trees without the need for binarization via a novel pointing mechanism.

Parsing with pointer networks. Pointer Networks (Vinyals et al., 2015a) are introduced to the parsing literature by Ma et al. (2018) and quickly become popular in various parsing subtasks because they are flexible to predict various trees/graphs and can achieve very competitive performance. Ma et al. (2018) linearize a dependency tree in a top-down depth-first and inside-out manner and use a pointer network to predict the linearized dependency tree, which is then extended by Lin et al. (2019) to discourse parsing. Liu et al. (2019) add shortcuts between the decoder states of the previously generated parents/siblings to the current decoder states in both dependency and discourse parsing. Fernández-González and Gómez-Rodríguez (2019) propose a left-to-right dependency parser that predicts the heads of each word autoregressively, and later, they propose right-to-left and outside-in variants (Fernández-González and Gómez-Rodríguez, 2021a). They also adapt the left-to-right dependency parser to semantic dependency parsing (which predicts acyclic graphs instead of trees) (Fernández-González and Gómez-Rodríguez, 2020), discontinuous constituency parsing (by treating discontinuous constituency trees as augmented dependency trees) (Fernández-González and Gómez-Rodríguez, 2020), and joint dependency and constituency parsing (Fernández-González and Gómez-Rodríguez, 2020). They use a pointer network to reorder the sentence to reduce discontinuous constituency parsing to continuous constituency parsing (Fernández-González and Gómez-Rodríguez, 2021b). Nguyen et al. (2021a,b) cast (discourse) constituency/RST parsing as conditional splitting and use pointer networks to select the splitting points. Zhou et al. (2021) propose an action-pointer network for AMR parsing.

Nested NER. There are also many methods to tackle nested NER, such as hypergraph-based methods (Lu and Roth, 2015; Katiyar and Cardie, 2018; Wang and Lu, 2018), sequence-labeling-based methods (Shibuya and Hovy, 2020; Wang et al., 2021), parsing-based methods (Finkel and Manning, 2009; Wang et al., 2018; Fu et al., 2021), layered methods (Fisher and Vlachos, 2019; Wang et al., 2020; Luo and Zhao, 2020), span-based methods (Yu et al., 2020; Li et al., 2021), object-detection-based methods (Shen et al., 2021; Tan et al., 2021) etc.

Our work belongs to the category of parsing-based methods. Finkel and Manning (2009) insert named entities into a constituency tree and use a discriminative parser (Finkel et al., 2008) for learning and prediction. Wang et al. (2018) adapt a shift-reduce transition-based parser to output a constituency forest instead of a constituency tree for nested NER. Fu et al. (2021) adapt a span-based neural TreeCRF parser, treat nested named entities as the observed parts of a partially-observed constituency tree and develop a masked inside algorithm to marginalize all unobserved parts for maximizing the probability of the observed named entities. Our method has a better performance as well as a lower time complexity than Fu et al. (2021). Recently, Lou et al. (2022) extend the work of Fu et al. (2021), casting nested NER to lexicalized constituency parsing for leveraging headword information. They achieve a higher performance at the cost of a higher parsing complexity, i.e., $O(n^4)$.

7 Discussion and future work

In the deep learning era, global optimization on trees becomes less important in both training and decoding. Teng and Zhang (2018) show that a span-based model trained with a local span classification loss performs well in conjunction with CYK decoding. Wei et al. (2020); Nguyen et al. (2020) show that top-down greedy decoding performs comparably. In this work we have shown that greedy decoding works well. Thus it would also be a fruitful direction to design more powerful neural decoders which can leverage more subtree information and can maintain structural consistency. Also, it is a fruitful direction to devise more powerful span representations.

8 Conclusion

In this work we have presented a novel pointing mechanism and model for bottom-up constituency parsing, which allows n-ary tree parsing in linear steps. Experiments on multiple datasets show the effectiveness of our methods in both constituency parsing and nested NER.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work was supported by the National Natural Science Foundation of China (61976139).

References

Kate Byrne. 2007. Nested named entity recognition in historical archive text. In *Proceedings of the First IEEE International Conference on Semantic Computing (ICSC 2007), September 17-19, 2007, Irvine, California, USA*, pages 589–596. IEEE Computer Society.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Austin, Texas. Association for Computational Linguistics.

Leyang Cui, Sen Yang, and Yue Zhang. 2021. Investigating non-local features for neural constituency parsing. *CoRR*, abs/2109.12814.

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).

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Hao Fei, Shengqiong Wu, Yafeng Ren, Fei Li, and Donghong Ji. 2021. Better combine them together! integrating syntactic constituency and dependency representations for semantic role labeling. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 549–559, Online. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2019. Faster shift-reduce constituent parsing with a non-binary, bottom-up strategy. *Artif. Intell.*, 275:559–574.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2019. Left-to-right dependency parsing with pointer networks. 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 710–716, Minneapolis, Minnesota. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2020. Discontinuous constituent parsing with pointer networks. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7724–7731. AAAI Press.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2020. Enriched in-order linearization for faster sequence-to-sequence constituent parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4092–4099, Online. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2020. Multitask pointer network for multi-representational parsing. *CoRR*, abs/2009.09730.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2020. Transition-based semantic dependency parsing with pointer networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7035–7046, Online. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2021a. Dependency parsing with bottom-up hierarchical pointer networks. *CoRR*, abs/2105.09611.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2021b. Reducing discontinuous to continuous parsing with pointer network reordering. *CoRR*, abs/2104.06239.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proceedings of ACL-08: HLT*, pages 959–967, Columbus, Ohio. Association for Computational Linguistics.

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.

Yao Fu, Chuanqi Tan, Mosha Chen, Songfang Huang, and Fei Huang. 2021. Nested named entity recognition with partially-observed treecrfs. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 12839–12847. AAAI Press.

Carlos Gómez-Rodríguez and David Vilares. 2018. Constituent parsing as sequence labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1314–1324, Brussels, Belgium. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

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.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. In *Proceedings of the Eleventh International Conference on Intelligent Systems for Molecular Biology, June 29 - July 3, 2003, Brisbane, Australia*, pages 180–182.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2020. Tetra-tagging: Word-synchronous parsing with linear-time inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6255–6261, Online. Association for Computational Linguistics.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinform.*, 36(4):1234–1240.

Fei Li, ZhiChao Lin, Meishan Zhang, and Donghong Ji. 2021. A span-based model for joint overlapped and discontinuous named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4814–4828, Online. Association for Computational Linguistics.

Xiang Lin, Shafiq Joty, Prathyusha Jwalapuram, and M Saiful Bari. 2019. A unified linear-time framework for sentence-level discourse parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4200, Florence, Italy. Association for Computational Linguistics.

Jiangming Liu and Yue Zhang. 2017. In-order transition-based constituent parsing. *Transactions of the Association for Computational Linguistics*, 5:413–424.

Linlin Liu, Xiang Lin, Shafiq Joty, Simeng Han, and Lidong Bing. 2019. Hierarchical pointer net parsing. 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 1007–1017, Hong Kong, China. Association for Computational Linguistics.

Chao Lou, Songlin Yang, and Kewei Tu. 2022. Nested named entity recognition as latent lexicalized constituency parsing. In *ACL*.

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.

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.

Chunpeng Ma, Lemao Liu, Akihiro Tamura, Tiejun Zhao, and Eiichiro Sumita. 2017. Deterministic attention for sequence-to-sequence constituent parsing. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 3237–3243. AAAI Press.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414, Melbourne, Australia. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Thanh-Tung Nguyen, Xuan-Phi Nguyen, Shafiq Joty, and Xiaoli Li. 2020. Efficient constituency parsing by pointing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3284–3294, Online. Association for Computational Linguistics.

Thanh-Tung Nguyen, Xuan-Phi Nguyen, Shafiq Joty, and Xiaoli Li. 2021a. A conditional splitting framework for efficient constituency parsing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5795–5807, Online. Association for Computational Linguistics.

Thanh-Tung Nguyen, Xuan-Phi Nguyen, Shafiq Joty, and Xiaoli Li. 2021b. RST parsing from scratch. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1613–1625, Online. Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York. Association for Computational Linguistics.

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron Courville, and Yoshua Bengio. 2018. Straight to the tree: Constituency parsing with neural syntactic distance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1180, Melbourne, Australia. Association for Computational Linguistics.

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

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.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Zeqi Tan, Yongliang Shen, Shuai Zhang, Weiming Lu, and Yueting Zhuang. 2021. A sequence-to-set network for nested named entity recognition. In *IJCAI*.

Zhiyang Teng and Yue Zhang. 2018. Two local models for neural constituent parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 119–132, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Yuanhe Tian, Yan Song, Fei Xia, and Tong Zhang. 2020. Improving constituency parsing with span attention. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1691–1703, Online. Association for Computational Linguistics.

David Vilares, Mostafa Abdou, and Anders Søgaard. 2019. Better, faster, stronger sequence tagging constituent parsers. 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 3372–3383, Minneapolis, Minnesota. Association for Computational Linguistics.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015a. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2692–2700.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2015b. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2773–2781.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus.

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.

Yiran Wang, Hiroyuki Shindo, Yuji Matsumoto, and Taro Watanabe. 2021. Nested named entity recognition via explicitly excluding the influence of the best path. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3547–3557, Online. Association for Computational Linguistics.

Yang Wei, Yuanbin Wu, and Man Lan. 2020. A span-based linearization for constituent trees. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3267–3277, Online. Association for Computational Linguistics.

Yang Wei, Yuanbin Wu, and Man Lan. 2021. In-order chart-based constituent parsing. *CoRR*, abs/2102.04065.

Qingrong Xia, Bo Zhang, Rui Wang, Zhenghua Li, Yue Zhang, Fei Huang, Luo Si, and Min Zhang. 2021. A unified span-based approach for opinion mining with syntactic constituents. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1795–1804, Online. Association for Computational Linguistics.

Xin Xin, Jinlong Li, and Zeqi Tan. 2021. N-ary constituent tree parsing with recursive semi-Markov model. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2631–2642, Online. Association for Computational Linguistics.

Yongxiu Xu, Heyan Huang, Chong Feng, and Yue Hu. 2021. A supervised multi-head self-attention network for nested named entity recognition. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 14185–14193. AAAI Press.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Nat. Lang. Eng.*, 11(2):207–238.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various NER subtasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5808–5822, Online. Association for Computational Linguistics.

Kaiyu Yang and Jia Deng. 2020. Strongly incremental constituency parsing with graph neural networks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics.

Yu Zhang, Houquan Zhou, and Zhenghua Li. 2020. Fast and accurate neural CRF constituency parsing. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 4046–4053. ijcai.org.

Jiawei Zhou, Tahira Naseem, Ramón Fernandez Astudillo, and Radu Florian. 2021. AMR parsing with action-pointer transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5585–5598, Online. Association for Computational Linguistics.

Junru Zhou and Hai Zhao. 2019. Head-Driven Phrase Structure Grammar parsing on Penn Treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence, Italy. Association for Computational Linguistics.

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

A Appendix

A.1 Uniqueness of oracle

If there are two oracles o_1 and o_2 outputting the same tree. Their parsing configuration is (c_1, A_1, p_1, S_1) and (C_2, A_2, p_2, S_2), respectively. Assume that the first kth pointing actions of o_1 and o_2 are the same (so they share the same cursor c) and the $k+1$th action is $(c \rightarrow a_1, y_1)$ and $(c \rightarrow a_2, y_2)$ respectively. We enumerate all possibilities:

- $a_2 < c < a_1$, then (a_2, c) exists in S_2. c_1 would be updated to a_1, so thereafter the endpoint of the generated span is $\geq a_1$, thus (a_2, c) cannot exist in S_1 since $c < a_1$.
- $a_1 < c < a_2$, then (a_1, c) exists in S_2. Similar to the previous case, we can conclude that (a_1, c) cannot exist in S_1.
- $a_1 < a_2 < c$, then (a_2, c) exists in S_2, but $a_2 \notin A_1$ for all remaining steps, thus (a_2, c) cannot exist in S_1.
- $a_2 < a_1 < c$. This is similar to the previous case.
- $c < a_1 < a_2$, then (a_1, c) exists in S_1, but $a_1 \notin A_2$ for the remaining steps, thus (a_1, c) cannot exist in S_2.
- $c < a_2 < a_1$. This is similar to the previous case.

Hence there is exact one oracle and we have proved it by contradiction.

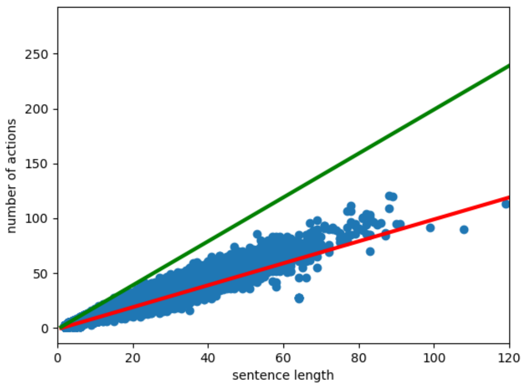


Figure 6: The number of actions with different sentence lengths in PTB

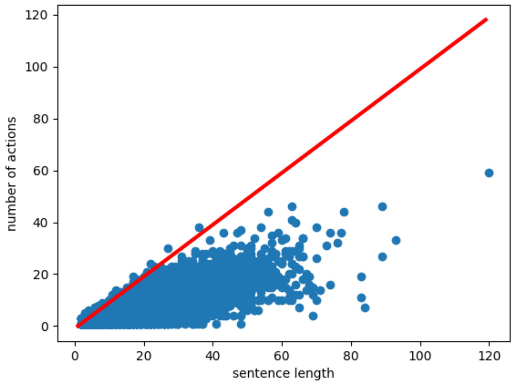


Figure 7: The number of actions with different sentence lengths in ACE2004

A.2 Number of actions

The system of Nguyen et al. (2021a) needs exact $n-1$ actions to parse a length-n sentence. While our model requires $2n-1$ actions in the worst case because we generate one span at each step and there are at most $2n-1$ spans if the corresponding constituency tree is a full binary tree. So there is a concern that our model needs twice time to parse. Empirically, since the constituency trees in the treebank are not full binary trees in most cases, we need less than $2n-1$ steps to parse. Fig. 6 shows the number of actions needed to parse with different sentence lengths in PTB training set. The red line is $y = x - 1$ and the green line is $y = 2x - 1$. In average, our method needs 1.13 actions per token, Nguyen et al. (2021a) needs 0.96 action per token. So, our method is around 20% slower than theirs. Fig. 7 shows the case in nested NER. We only need 0.40 action per token since the spans in nested NER is more *sparse* than that in constituency parsing. Our method is expectedly faster than other parsing-based methods in nested NER, such as the transition system of Wang and Lu (2018), which needs at least one action per word; and the span-based method of Fu et al. (2021), which needs cubic time for CYK parsing.