N-ary Constituent Tree Parsing with Recursive Semi-Markov Model

Xin Xin[*], **Jinlong Li**
School of Computer Science and Technology,
Beijing Institute of Technology, Beijing, China
BJ ER Center of HVLIPCCA, BIT, Beijing, China
{xxin,jllee}@bit.edu.cn

Zeqi Tan
Zhejiang University,
Hangzhou, China
zqtan@zju.edu.cn

Abstract

In this paper, we study the task of graph-based constituent parsing in the setting that binarization is not conducted as a pre-processing step, where a constituent tree may consist of nodes with more than two children. Previous graph-based methods on this setting typically generate hidden nodes with the dummy label inside the n-ary nodes, in order to transform the tree into a binary tree for prediction. The limitation is that the hidden nodes break the sibling relations of the n-ary node's children. Consequently, the dependencies of such sibling constituents might not be accurately modeled and is being ignored. To solve this limitation, we propose a novel graph-based framework, which is called "recursive semi-Markov model". The main idea is to utilize 1-order semi-Markov model to predict the immediate children sequence of a constituent candidate, which then recursively serves as a child candidate of its parent. In this manner, the dependencies of sibling constituents can be described by 1-order transition features, which solves the above limitation. Through experiments, the proposed framework obtains the F1 of 95.92% and 92.50% on the datasets of PTB and CTB 5.1 respectively. Specially, the recursive semi-Markov model shows advantages in modeling nodes with more than two children, whose average F1 can be improved by 0.3-1.1 points in PTB and 2.3-6.8 points in CTB 5.1.

1 Introduction

There are two settings for constituent parsing models, including binary tree parsing and n-ary tree parsing. In the former, the original constituent tree with n-ary nodes is converted into a binary tree by language-specific rules. The model first predicts the binary tree, and then converts it back. In the latter, the model directly predicts the n-ary tree without the intermediate step of binarization.

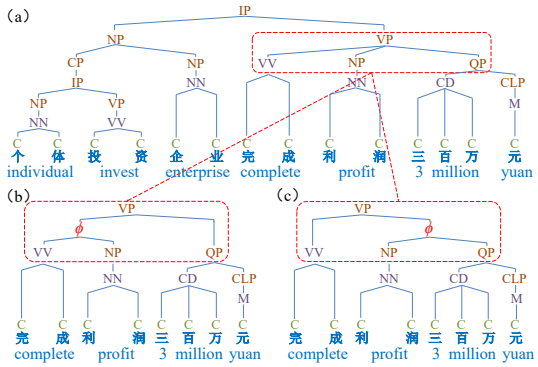


Figure 1: An n-ary node and the hidden nodes.

In the paper, we focus on the setting of n-ary tree parsing. Compared with binary tree parsing, which has the advantage of utilizing the lexical head information, n-ary tree parsing is more natural to fit the original tree structure, and is more adaptable to languages that do not have head rules for binarization. In addition, for languages with the word segmentation issue, such as Chinese, it is very convenient for n-ary tree parsing models to deal with the joint task of word segmentation, part-of-speech (POS) tagging and constituent parsing, by just enlarging the label set with the POS labels, as shown in Fig. 1 (a), which alleviates the error propagation from the pipeline.

Specifically, we target at improving graph-based models for n-ary tree parsing, which obtain better performances in recent work (Kitaev et al., 2019; Zhang et al., 2020; Wei et al., 2020) from the two streams of well-developed parsing methods, graph-based and transition-based. For n-ary tree parsing, the main idea of previous graph-based models is to generate hidden nodes with the dummy label ϕ inside the n-ary node, in order to expand the n-ary tree into a binary tree. In this way, n-ary tree parsing can be converted into binary tree parsing with hidden nodes, which are unobservable in the training process. Consider the n-ary node "VP→VV,

[*]Xin Xin is the corresponding author.

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 2631–2642
August 1–6, 2021.

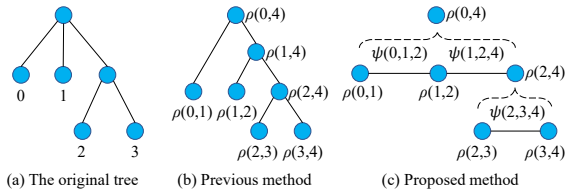


Figure 2: Comparisons of previous and our models. (i, j) denotes the span from i to $j-1$. $\rho(i, j)$ denotes the feature score of span (i, j); $\psi(i, j, k)$ denotes the feature score of the sibling span pair (i, j) and (j, k).

NP, QP" in Fig. 1 (a) as an example. The hidden nodes can be in two manners, as shown in Fig. 1 (b, c). Either of them can be seen as being correct in training. For convenience, the potential scores of such hidden nodes are manually set to zero, to ensure that the two manners are equivalent when calculating the likelihood (Kitaev and Klein, 2018).

The limitation of previous methods is that the generated hidden nodes break the sibling relations of the n-ary node's children. Consequently, such sibling dependency feature might not be accurately modeled and is being ignored. Consider the node "VP→VV, NP, QP" in the above example. If we model the 1-order dependency from the sibling node pair, dependency feature scores should be calculated from both pairs of (VV, NP) and (NP, QP). Without loss of generality, suppose the hidden node is as shown in Fig. 1 (b), and the case in Fig. 1 (c) is similar. As the hidden node ϕ is forced to be the sibling node of "QP", the dependency feature of (NP, QP) cannot be directly calculated. In implementation, only potential scores of each node are modeled, and the dependency potential scores of sibling node pairs are being ignored.

To solve this limitation, we propose a novel framework for n-ary tree parsing. Our main idea is to utilize 1-order semi-Markov model to directly predict the immediate children sequence of an n-ary node, without generating the hidden nodes for binarization, as shown in Fig. 2. Different from previous models that only have potential scores on nodes when evaluating a tree's likelihood, the potential scores of sibling node pairs are also calculated as 1-order transition features. Thus dependencies from sibling nodes can be naturally modeled, which solves the above limitation. When generating an n-ary tree, the semi-Markov model is recursively conducted on the node spans in a bottom-up manner, thus we call the proposed model "recursive semi-Markov model".

The main challenge of designing the recursive semi-Markov model is how to make the computational complexity being acceptable. In nowadays GPU era, to make full use of parallel computation is an important issue to enhance the processing speed. For example, in the previous CYK (Kasami, 1966) algorithm for binary trees, the absolute time complexity is $O(n^3)$, where n is the sentence length. But $O(n^2)$ out of it can be computed in parallel, by batchifying the spans with the same length and the divisions within a span. This means the hard time complexity of CYK, which cannot be computed in parallel, is $O(n)$. In the case of the proposed recursive semi-Markov model, the time complexity of the straight-forward dynamic programming algorithm is $O(n^5)$. But by careful design, we propose an algorithm, whose complexity is $O(n^4)$, with $O(n^3)$ out of it can be batchified. It means the increased $O(n)$ complexity compared with CYK can be calculated in parallel. In practice, the proposed framework can process 26 and 11 sentences per second in PTB and CTB 5.1 test sets respectively, by a single NVIDIA RTX GPU.

Our main contributions can be summarized as follows. (1) We propose a novel graph-based framework, recursive semi-Markov model, for n-ary constituent tree parsing, which can model the dependencies of sibling nodes. (2) We design a dynamic programming algorithm for the proposed framework, whose complexity is $O(n^4)$, with $O(n^3)$ inside can be batchified. (3) Experimental verifications demonstrate that the proposed framework outperforms previous methods. The F1 of the proposed framework is 95.92% and 92.50% in PTB and CTB 5.1 respectively. In the joint task with segmentation and POS tagging in CTB 5.1, the F1 is 91.84%. In addition, the proposed framework can effectively predict nodes with more than two children, improving the F1 by 0.3-1.1 points in PTB and 2.3-6.8 points in CTB 5.1.

Our code is released at *https://github.com/NP-NET-research/Recursive-Semi-Markov-Model*, which is developed on the base of the open-source Berkekey parser (Kitaev and Klein, 2018; Kitaev et al., 2019).

2 Related Work

2.1 Early Models for N-ary Tree Parsing

A representative of classical methods for n-ary tree parsing is the Earley algorithm (Earley, 1970). It can find legal trees of sentence fitting the grammar

rules with the complexity of $O(Cn^3)$ by dynamic programming, where n is the sentence length and C is dependent on the complexity of grammar rules. The dependency with the size of grammar rules in the Earley algorithm increases the computational complexity substantially in practice. Therefore, recent studies have paid more attention to utilizing "less grammar" (Hall et al., 2014), which is implemented in CYK/shift-reduce algorithms (Durrett and Klein, 2015; Liu and Zhang, 2017b; Stern et al., 2017; Teng and Zhang, 2018) instead of the Earley algorithm. It demonstrates it can reduce the complexity and also obtain better performances.

Our proposed framework is in line with the recent studies, whose complexity is independent with the size of grammar rules.

2.2 Graph-Based N-ary Tree Parsing

Graph-based parsing models utilize the CYK algorithm to find the tree with the largest feature score as the prediction. The main advantage is the large search space and the globally optimal inference. A representative of graph-based n-ary tree parsing model is the Berkeley parser (Stern et al., 2017; Kitaev and Klein, 2018; Kitaev et al., 2019), which employs hidden nodes to deal with n-ary nodes.

The proposed framework belongs to graph-based n-ary tree parsing models. Compared with previous work, the novelty lies in that semi-Markov model is utilized to directly model the children sequence of an n-ary node, instead of generating a binary tree with hidden nodes. Consequently, it can avoid breaking the sibling relation of nodes in the sequence. The proposed framework then makes use of such dependencies to improve the parsing performance.

2.3 Transition-Based N-ary Trees Parsing

Transition-based models make predictions sequentially, with advantages of the low computational cost and the utilization of high-order features. The models can be divided into post-order (Cross and Huang, 2016; Fernández-González and Gómez-Rodríguez, 2019), pre-order (Dyer et al., 2016), and in-order (Liu and Zhang, 2017a), according to the traversal manner of the action sequence. Post-order models require to deciding the number of reduced nodes for n-ary nodes (Fernández-González and Gómez-Rodríguez, 2019), or to introducing hidden nodes with dummy label (Cross and Huang, 2016). Pre-order models and in-order models are born to have convenience in dealing with n-ary nodes, as the number of reduced nodes is fixed.

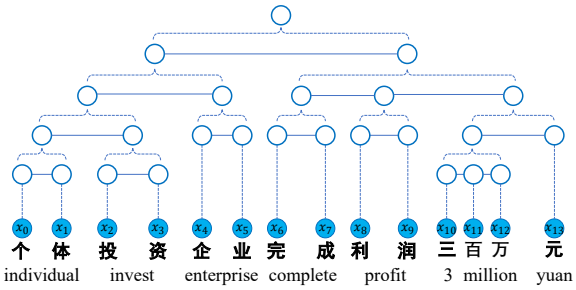


Figure 3: Probabilistic graph of recursive semi-Markov model.

Both the proposed framework and some of the above methods directly model the sequence within an n-ary node. The novelty of the proposed framework is that it models the sequence as a graph-based model rather than a transition-based model. Transition-based models suffer from the limitation of local optimization in the inference process, but graph-based models can guarantee the globally optimal inference. In recent studies, graph-based models have been demonstrated to perform better than transition-based models (Kitaev et al., 2019; Zhang et al., 2020; Wei et al., 2020).

3 The Recursive Semi-Markov Model

3.1 Preliminaries

A sentence is denoted by $x = \{x_i\}$, with x_i being the i^{th} word. The sentence length is denoted by n. Let $\mathcal{Y}$ be the set of the alphabet constituent labels. Following previous work (Kitaev and Klein, 2018; Zhang et al., 2020), the nodes with unary grammars are collapsed, and its label is replaced by the joint label of the collapsed nodes. For example, in Fig. 1 (a), "CP→IP" will be replaced by "CP+IP", where "CP+IP" is an atomic label. Given x, the task is to build an n-ary tree on top of it, and assign a label to each internal node. When conducting the joint parsing task with word segmentation and POS tagging in Chinese, $\mathcal{Y}$ is enriched with the POS labels and a "C" label (denoting characters), and x_i denotes the i^{th} character. For example, in Fig. 1 (a), "NN" is a POS label, and "NP+NN" is treated as an atomic label for the corresponding node in the joint parsing task.

3.2 The Framework Structure

In the proposed recursive semi-Markov model, the probabilistic graph of a constituent tree is shown

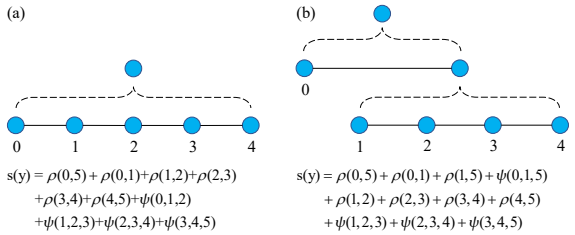


Figure 4: Examples of potential scores for a whole tree. For the convenience of presentation, we omit the labels. The full presentation for $\rho(i, j)$ is $\rho(i, j, l)$ and the full presentation for $\psi(i, j, k)$ is $\psi(i, j, k, l_1, l_2)$.

in Fig. 3. This graph corresponds to the tree in Fig. 1 (a). Full circles refer to the input x. Blank circles refer to the internal nodes, which can be seen as variables in the probabilistic graph. The full line, which connects two nodes, means that the two nodes are dependent with each other. The dotted line pointing to an internal node refers to the sequence of the node's immediate children. There are two kinds of cliques in the graph, the one with a single node, and the one with two sibling nodes. The former corresponds to 0-order cliques, and the latter corresponds to 1-order cliques. The whole framework is a 1-order semi-Markov model.

Potential scores, which are assigned to the above two kinds of cliques, are denoted by $\rho(i, j, l|x, \theta)$, and $\psi(i, j, k, l_1, l_2|x, \theta)$, respectively. θ is the model parameters, including neural network weights and word embeddings. In the following, we omit the symbol x and θ in equations for presentation simplicity. $\rho(i, j, l)$ defines the emission feature score of a span, describing how likely the span is a constituent. (i, j) denotes a span which starts at i and ends at $j - 1$, $0 \leq i < j \leq n$. $l \in \mathcal{Y}$ denotes the span's label. $\psi(i, j, k, l_1, l_2)$ defines the transition feature score of two sibling spans, describing how likely the two spans are sibling neighbors within an n-ary node. (i, j, k) denotes the two sibling spans (i, j) and (j, k). l_1 is the label of the left span, and l_2 is the label of the right span.

Let y denote a predicted tree given x. The conditional probability $p(y|x)$ can be defined on the probabilistic graph, under the framework of conditional random fields (CRF) (Lafferty et al., 2001), as shown in the following equations. $C_1(y)$ denotes the set of emission scores, and $C_2(y)$ denotes the set of transition scores. $\mathcal{T}(x)$ denotes all legal n-ary trees that can be built on top of the input sentence x. $s(y)$ is the sum of clique potential scores defined in a whole tree, with two examples shown

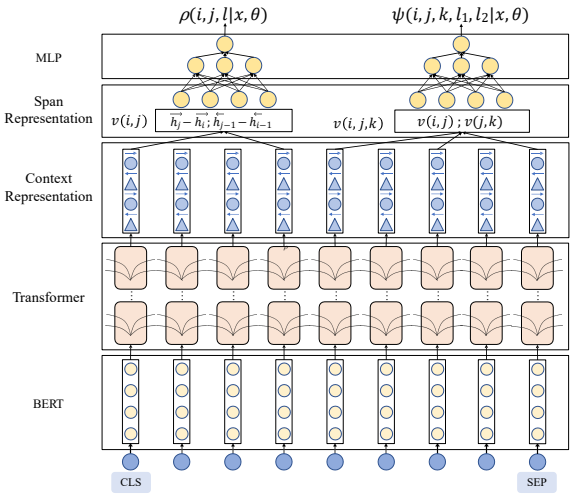


Figure 5: The neural architecture for feature learning.

in Fig. 4. Given the parameters θ, the inference process is to find a tree with the largest probability.

$$s(y) = \sum_{C_1(y)} \rho(i, j, l) + \sum_{C_2(y)} \psi(i, j, k, l_1, l_2) \quad (1)$$

$$p(y|x) = \frac{\exp(s(y)))}{\sum_{y' \in \mathcal{T}(x)} \exp(s(y'))}$$

3.3 Potential Score Calculations

Given an input sentence x, we follow the neural network architecture of the Berkeley parser (Kitaev and Klein, 2018) with some minor revisions, to calculate the two kinds of potential scores, $\rho(i, j, l)$, and $\psi(i, j, k, l_1, l_2)$, as shown in Fig. 5.

In the embedding layer, the BERT (Devlin et al., 2019; Wolf et al., 2020) is selected to generate pre-trained vectors, denoted by e_i, $0 \leq i < n$. For the Chinese language, e_i refers to the i^{th} character, and the embedding vector of last character within the word is chosen to represent the word.

$$e_i = \mathbf{BERT}(x_i|x)$$

In the encoding layer, the Transformer (Vaswani et al., 2017) is selected for extracting the context features, denoted by h_i, with odd dimensions $\overrightarrow{h}_i$ and the even dimensions $\overleftarrow{h}_i$.

$$h_i = \mathbf{Transformer}(e_i|x)$$

The representation of a single span (i, j) is formed by $v(i, j) = [\overrightarrow{h}_j - \overrightarrow{h}_i; \overleftarrow{h}_{j-1} - \overleftarrow{h}_{i-1}]$, and the representation of a sibling span pair (i, j) and (j, k) is formed by $v(i, j, k) = [v(i, j); v(j, k)]$. $[;]$

is the concatenate operation. By passing $v(i,j)$ and $v(i,j,k)$ through multi-layer perceptrons (MLP), the emission potential score is finally defined as

$$\rho(i,j,l) = \mathbf{MLP}_l^{\text{emission}}(v(i,j)),$$

and the transition potential score is defined as

$$\psi(i,j,k,l_1,l_2) = \mathbf{MLP}_{l_1,l_2}^{\text{transition}}(v(i,j,k)).$$

There are totally $|\mathcal{Y}|$ MLPs for ρ. $|\mathcal{Y}|$ is the size of the label set $\mathcal{Y}$. Parameters in the hidden layers are shared among them, and only the parameters of the output layers are different to distinguish different labels. Similarly, there are $|\mathcal{Y}|^2$ MLPs for ψ, whose parameters in hidden layers are also shared.

3.4 The Max-Margin Loss

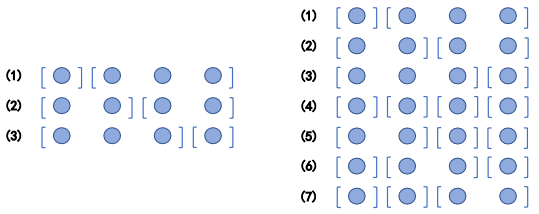
When designing the loss function, theoretically, we can follow the CRF framework to optimize the log-likelihood of the training data. But in practice, if we do this, the gradients of all potential scores, which is $O(n^4)$ (n is the sentence length), should be stored in the GPU memory. This is impossible to be implemented in a general GPU device. Therefore, we employ the max-margin loss as the training objective to learn the parameters of the proposed framework, following the Berkeley parser (Kitaev and Klein, 2018). By max-margin, only the gradients of the predicted tree structure and the gold structure need to be stored, which is $O(n)$. Consequently, it saves a lot of memory in implementation.

Let $s(y)$ in Eq. 1 denote the total potential score of a tree y. Suppose the gold tree is y^g, with the potential score $s(y^g)$. The key idea of the max-margin loss is to let the maximum potential score of the other trees, denoted as $s(y^*)$, be less than $s(y^g)$ by an acceptable margin. In the probability space, it is equivalent that the probability of the gold tree is larger than the maximum probability of the other trees by a margin. The formal definition of the objective is to minimize the following hinge loss, where $\Delta(y, y^g)$ refers to the number of spans in y^g not matched in y.

$$L = \max\left(0, \max_{y \in \mathcal{T}(x)} [s(y) + \Delta(y, y^g)] - s(y^g)\right)$$

3.5 Explanations of the Proposed Model

The Semi-Markov Property. The semi-Markov property of the proposed model refers to the one mentioned in Sarawagi and Cohen's work (Sarawagi and Cohen, 2004). When finding the immediate children of a constituent span, the linear-chain Markov structures are assumed over the sequence of candidate immediate constituents. In the implementation, we treat it as a segmentation problem, where each immediate child span can be seen as a segment, which has the similar setting with the previous work (Sarawagi and Cohen, 2004). Compared with the traditional "B-I-O" tagging schema in segmentation, which assigns a label to each token, the emission feature ρ is defined on the whole segment of several tokens in the proposed model, which is non-Markovian. Markov property exists in adjacent segments from the transition feature ψ. This shows the semi-Markov property.



(a) CYK (b) Semi-Markov Model

Figure 6: Comparisons between the CYK algorithm and the recursive semi-Markov model.

Connections with CRF. Traditional CRF models define a conditional probability over a probabilistic graph, and utilize the maximum likelihood estimation as the optimization objective. The proposed model shares the same conditional probability definition from the explanation view, but utilizes a margin-based loss in order to save the computational memory.

4 Algorithms

4.1 The Challenge

The core for the optimization is to find the tree with the maximum potential score. The previous CYK algorithm utilizes dynamic programming to find the maximum score, in a bottom-up manner. In order to calculate the maximum score of a given span, all the divisions should be enumerated. As shown in Fig. 6 (left), in the binary tree case, the number of the divisions is equal to $L-1$, where L is the span length. Besides, the span length should be enumerated from 1 to n, and for each span length L there is $L-n+1$ spans. Therefore the total time complexity of previous CYK is $O(n^3)$. In our case, a span can have more than two immediate

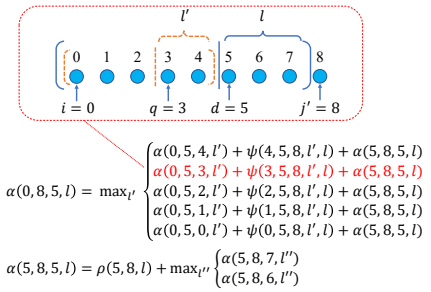


Figure 7: An example of the dynamic programming.

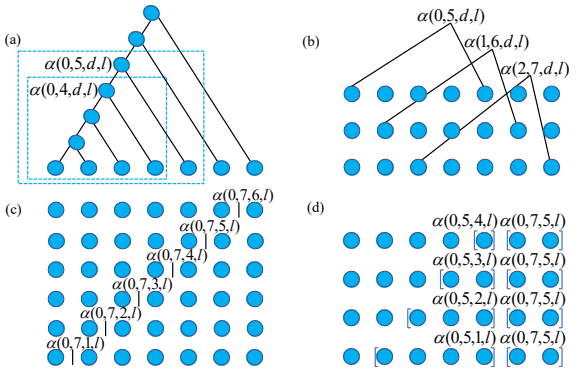


Figure 8: The main steps of the proposed algorithm.

children. Therefore, all the segmentation sequences should be enumerated, which obviously enlarges the search space. In Fig. 6 (right), for a span with the length equal to 4, the number of sequences to be considered increases from 3 to 7. This difference is the key issue to be solved in this section.

4.2 Straight-Forward Algorithm ($O(n^5)$)

Let (i, j) be a representative span $(i < j)$. We need to find its immediate children sequence with the maximum potential score. Dynamic programming is employed to accumulate the maximum potential score from the left to the right. Let $\alpha(i, j', d, l)$ be an accumulated variable in the dynamic programming, which accumulates potential scores from $j' = i + 1$ to $j' = j$. j' denotes the current accumulated position. d $(i < d < j')$ means that the last immediate child for span (i, j') is the span (d, j'). l refers to the label of (d, j'). The meaning of $\alpha(i, j', d, l)$ is the maximum accumulated score chosen from all the immediate children sequences of span (i, j') whose last immediate child is (d, j') with the label l. We also include the case of $d = i$, which refers to the maximum accumulated score of the span (i, j')'s children and the span (i, j') itself with l as its label.

$$\alpha(i, i+1, i, l) = \rho(i, i+1, l)$$

$$\alpha(i, j', d, l) = \max_{i \le q < d, l' \in \mathcal{Y}} [\alpha(i, d, q, l') + \psi(q, d, j', l', l) + \alpha(d, j', d, l)]$$

$$\alpha(i, j', i, l) = \rho(i, j', l) + \max_{i < k < j', l' \in \mathcal{Y}} \alpha(i, j', k, l')$$

In semi-Markov model, the above iterative calculation equations hold for the dynamic programming. The first equation is the initial state when $j' = i + 1$, and the second and third equations are the iterative functions when $(i < d < j', j' > i+1)$ and $(d = i, j' > i + 1)$, respectively. An example of the dynamic programming is shown in Fig. 7.

In the iterative calculation of the above dynamic programming, we need to enumerate q, d, j', i, j, each of which has the complexity of $O(n)$. The total time complexity of the straight-forward method is $O(n^5) * O(|\mathcal{Y}|^2)$. To simplify the complexity of $|\mathcal{Y}|^2$, in calculating $\psi(q, d, j', l', l)$, we manually group the labels in $\mathcal{Y}$ into clusters, according to the meaning of the constituent label, which reduces the complexity of $|\mathcal{Y}|^2$. Consequently, the main complexity comes from the $O(n^5)$ part.

4.3 The Proposed Algorithm ($O(n) * O^p(n^3)$)

In this section, we introduce how to reduce the above complexity of $O(n^5)$ to $O(n) * O^p(n^3)$. $O^p(n^3)$ means all the $O(n^3)$ calculations can be batchfied. The hard complexity, which cannot be computed in parallel, is $O(n)$.

The overall procedure for designing the algorithm is shown in Fig. 8. It includes four steps for reducing or batchifying the time complexity. In the first step, the complexity is reduced from $O(n^5)$ to $O(n^4)$ by sharing the α values in a set of spans. As shown in Fig. 8 (a), in the span of (0, 5), we need to calculate $\alpha(0, j, d, l)$ by enumerating j from 1 to 5. But the value $\alpha(0, 4, d, l)$ has been calculated in the span of (0, 4). Iteratively, all the values of $\alpha(0, j, d, l)(0 < j < 5)$ have been calculated in previous spans starting from 0. This means a set of spans that have the same start position can share the α values. If we enumerate the span length in the ascending order, in span (i, j), only the j^{th} position's value $\alpha(i, j, d, l)$ needs to be calculated, instead of enumerating the position j' from $i + 1$ to j, which reduces $O(n)$ of the time complexity. In the second step, the complexity is batchified

Algorithm 1 Algorithm for recursive semi-Markov model.

Input: sentence x (length N), model parameters θ.
Outputs: the constituent tree y^* with the maximum potential score $s(y^*|x;\theta)$.

for all spans (i, j) **do**
  calculate $\rho(i, j, l|x;\theta)$.
end for
for all sibling span pairs (i, j) and (j, k) **do**
  calculate $\psi(i, j, k, l_1, l_2|x;\theta)$.
end for
for span length T from 1 to N **do**
  calculate $\alpha(i, i+T, d, l)$.
  $0 \leq i \leq N-T, i \leq d < i+T$
end for
$s(y^*|x;\theta) = \max_{d,l} \alpha(0, N, d, l)$.
trace back the tree y^*.

from $O(n^4)$ to $O(n^3) * O^p(n)$, by computing the spans of the same length in parallel, as shown in Fig. 8 (b). In the third step, the complexity is batchified from $O(n^3) * O^p(n)$ to $O(n^2) * O^p(n^2)$, by computing different ds in $\alpha(i, j, d, l), i < d < j$ in parallel, as shown in Fig. 8 (c). In the fourth step, the complexity is batchified from $O(n^2) * O^p(n^2)$ to $O(n) * O^p(n^3)$, by computing $\alpha(i, j, d, l)$ when enumerating the second last immediate child with $i < q < d$ in parallel (To calculate the dynamic programming state at a new position given the last child, we need to enumerate previous states with different second last children, in order to calculate ψ), as shown in Fig. 8 (d).

The details of the proposed algorithm are shown in Alg. 1. The calculation of $\rho(i, j, l|x;\theta)$ and $\psi(i, j, k, l_1, l_2|x;\theta)$ can be easily computed in parallel, with the complexity $O^p(n^2)$ and $O^p(n^3)$, respectively. The complexity of calculating α is $O(n) * O^p(n^3)$. Therefore, the total time complexity of the proposed algorithm is $O(n) * O^p(n^3)$.

5 Experiments

5.1 Experimental Setup

We evaluate the proposed framework in both English and Chinese, on the datasets of PTB (WSJ sections (Marcus et al., 1993)) and CTB 5.1 (Xue et al., 2005), respectively. For Chinese, we evaluate both the single task of constituent parsing and the joint task with word segmentation and POS tagging. We follow the standard split of the datasets (Kitaev et al., 2019). In the single task for Chinese, some previous work utilize the Stanford tagger (Toutanova et al., 2003) to generate the POS tags as input, which leads to a fixed error propagation. In this paper, POS tags are removed and not used as input features in both training and testing in CTB 5.1, following the previous work in (Zhang et al., 2020). Standard precision, recall and F1-measure are employed as evaluation metrics, where the EVALB[1] tool is employed in the single task. The hyper-parameters in the implementation are shown in Table. 1. Most of them are set following the Berkeley parser (Kitaev and Klein, 2018). When choosing the pre-train models (Wolf et al., 2020), "bert-large-cased" is utilized for English with a single RTX 3090, "bert-base-chinese" is utilized for Chinese with a single RTX 1080TI.

Parameter	Value	Parameter	Value
batch size	32	learning rate	10^{-5}
decay factor	0.5	decay patience	5
max decay	3	dropout	0.2
MLP layer	1	MLP hidden	250
Trans. hidden	1024	Trans. layer	2
head number	8	label hidden	250

Table 1: Hyper-parameters.

Model	P	R	F1
Dyer et al. (2016)	–	–	93.3
Choe and Charniak (2016)	–	–	93.8
Liu and Zhang (2017a)	–	–	94.2
Fried et al. (2017)	–	–	94.66
Stern et al. (2017)	92.98	90.63	91.79
Liu et al. (2018)	–	–	92.3
Shen et al. (2018)	92.0	91.7	91.8
Gómez-Rodríguez and Vilares (2018)	–	–	90.7
Gaddy et al. (2018)	92.41	91.76	92.08
Teng and Zhang (2018)	92.5	92.2	92.4
Hong and Huang (2018)	91.5	92.5	92.0
Joshi et al. (2018)	93.8	94.8	94.3
Vilares et al. (2019)	–	–	90.60
Kitaev and Klein (2018)	94.85	95.40	95.13
Kitaev et al. (2019)	95.46	95.73	95.59
Zhou and Zhao (2019)	95.70	95.98	95.84
Zhang et al. (2020)	95.85	95.53	95.69
Wei et al. (2020)	95.5	**96.1**	95.8
Ours	**96.29**	95.55	**95.92**

Table 2: Single-task performances on test set of PTB.

5.2 Performances

The overall performances of the proposed framework in the single task of constituent parsing on

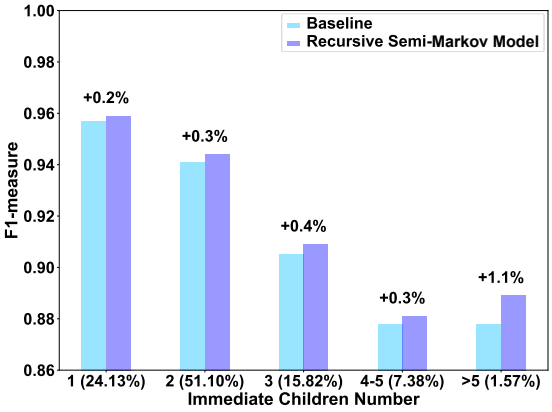
[1] https://nlp.cs.nyu.edu/evalb

Model	P	R	F1
Watanabe and Sumita (2015)	–	–	84.33
Gómez-Rodríguez and Vilares (2018)	–	–	84.40
Dyer et al. (2016)	–	–	84.60
Liu and Zhang (2017b)	85.90	85.20	85.50
Vilares et al. (2019)	–	–	85.61
Liu and Zhang (2017a)	–	–	86.10
Shen et al. (2018)	86.60	86.40	86.50
Wang et al. (2015)	–	–	86.60
Fernández-González and Gómez-Rodríguez (2019)	–	–	86.80
Fried and Klein (2018)	–	–	87.00
Teng and Zhang (2018)	87.50	87.10	87.30
Kitaev et al. (2019)	91.96	91.55	91.75
Zhou and Zhao (2019)	92.03	92.33	92.18
Zhang et al. (2020)	92.51	92.04	92.27
Wei et al. (2020)	92.2	**92.7**	92.4
Ours	**92.94**	92.06	**92.50**

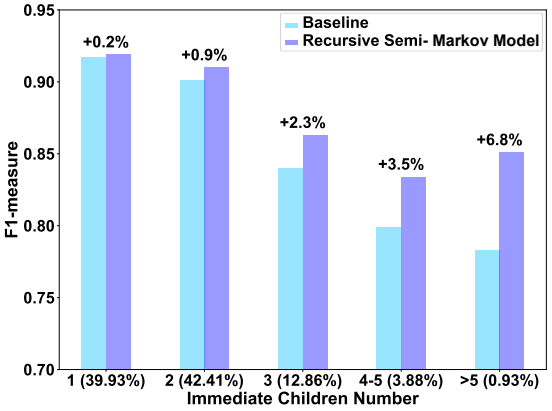
Table 3: Single-task performances on test set of CTB.

Model	Seg-F1	Pos-F1	Par-F1
Wang et al. (2006)	76.20	78.00	77.10
Jiang et al. (2009)	–	–	81.07
Qian and Liu (2012)	97.96	93.81	82.85
Wang et al. (2013)	97.86	94.40	83.42
Zhang et al. (2013)	97.84	94.80	84.43
Zheng et al. (2015)	–	–	84.22
Baseline	98.35	96.32	91.38
Ours	**98.92**	**96.70**	**91.84**

Table 4: Joint-task performances on test set of CTB. The "baseline" row shows our running results using a revision of the Berkeley parser (Kitaev et al., 2019).



(a) PTB



(b) CTB

Figure 9: F1-measure values on constituent nodes with different numbers of children. The "baseline" refers to our running results using the Berkeley parser (Kitaev et al., 2019). The percentage values at bottom refers to the distribution of different nodes.

the test set are shown in Table. 2 and Table. 3. The baselines in the first block are mainly based on basic word embeddings, and the baselines in the second block are based on BERT (Wolf et al., 2020). It can be observed that the F1-measures of proposed framework are 95.92% in PTB and 92.50% in CTB 5.1, which outperform the previous state-of-the-art methods. Our implementation for the proposed framework is based on the Berkeley parser (Kitaev et al., 2019). Therefore, many settings are similar with it for fair comparisons, such as learning schedule and feature normalization. Our method outperforms it by 0.33 points in PTB and 0.5 points in CTB 5.1 (0.25% of the 0.75% improvement is due to not utilizing automatically predicted POS tags in CTB 5.1), which demonstrates the advantage of modeling the sibling dependency features.

The overall performances in the joint task on the test set of CTB 5.1 are shown in Table. 4. As there are rare reports of performances with the BERT embedding, we have implemented a minor revision to the previous Berkeley parser (Kitaev et al., 2019) to make it adaptable to the joint task, which serves as the baseline method in the second block. It can observed that the F1-measures of proposed recursive semi-Markov model outperforms the competitive baseline by 0.46 points in F1, and consistently outperforms previous method in all tasks of word segmentation, POS tagging, and parsing.

The main improvement of the proposed framework comes from modeling the sibling dependencies of an n-ary node's children sequence. It has special advantage for predicting nodes with more children. We have divided all the constituent nodes into bins by how many children they have. Figure 9 shows the comparisons. The improvement is more obvious when the number of children becomes larger. For nodes with more than 2 immediate children, our framework outperforms the baseline by 0.3 to 1.1 points in PTB and 2.3 to 6.8 points in CTB 5.1.

Model	Sent./Sec.
Zhu et al. (2013)	90
Stern et al. (2017)	76
Shen et al. (2018)	111
Gómez-Rodríguez and Vilares (2018)	780
Zhou and Zhao (2019)	159
Wei et al. (2020)	220
Zhang et al. (2020)	1092
Ours	26

Table 5: Speed comparisons of different methods. The results of other methods are referred from the previous papers, and the hardware equipments are different.

5.3 Speed Analysis

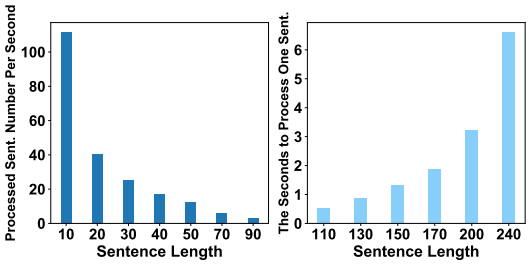


Figure 10: Speed analysis.

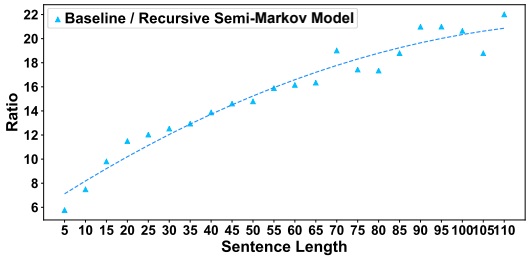


Figure 11: Speed comparisons with the baseline. Ratio=Speed(Baseline)/Speed(Ours).

The average processing speed in PTB test set is 26 sentences per second with a single RTX 3090, and the one in CTB 5.1 test set is 11 sentences per second with a single RTX 1080TI (or 20 sentences per second with single RTX 3090). Table 5 shows the speed comparisons of the proposed model with previous methods in the PTB dataset. Figure 10 shows the detailed processing speed of the proposed model in CTB 5.1 dataset. Figure 10 (left) shows the processing speeds with different sentence lengths; and Fig. 10 (right) shows the processing time of some special long sentences. For the longest sentence in the CTB 5.1, which contains 240 words, it takes around 6 seconds. Figure 11 shows the processing speed ratio between the Berkeley parser (Kitaev et al., 2019) and our model. It demonstrates that ratio does not grow linearly, by making full use of parallel computations. We know that the speed is still slower than some previous methods. On one hand, our proposed algorithm has already reduced the complexity by parallel computations. On the other hand, by considering its advantage in modeling nodes with multiple children, which especially happens a lot in the joint parsing task with segmentation and POS tagging in Chinese, the processing speed is still acceptable in many offline cases.

5.4 A Further Comparison on Fine-Grained Noun Phrase Structures

Within the nodes having more than two children, some of them are noun phrases, whose internal hierarchical structures have been annotated in the PTB dataset by previous work (Vadas and Curran, 2007, 2011). We have also conducted experiments with the Berkeley parser (Kitaev et al., 2019) on this refined PTB data. In the test process, we convert the generated fine-grained trees back to the original trees for comparisons. The F1 in the refined PTB test dataset by the Berkeley parser (Kitaev et al., 2019) is 95.62%, which is also outperformed by the proposed method in Table 2.

6 Conclusion

In this paper, a recursive semi-Markov model is proposed for n-ary constituent tree parsing, with the advantage of modeling the sibling relations within n-ary node. Experimental verifications on PTB and CTB 5.1 demonstrate that the proposed framework outperforms previous work in the single parsing task of both datasets and the joint task in CTB 5.1. For constituent nodes with more than 2 children, the F1 can be improved by $0.3 - 1.1$ points in PTB and $2.3 - 6.8$ points in CTB 5.1.

Acknowledgments

This work is supported by the grants from the National Natural Science Foundation of China (No. 61672100), and Beijing Natural Science Foundation (No. 4202069). It is partly supported by National Key R&D Program of China (No. 2018YFC0830705). We thank the anonymous reviewers for their comments and constructive suggestions to improve this paper.

References

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas. Association for Computational Linguistics.

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.

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.

Greg Durrett and Dan Klein. 2015. Neural CRF parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 302–312, Beijing, China. Association for Computational Linguistics.

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.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94–102.

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

Daniel Fried and Dan Klein. 2018. Policy gradient as a proxy for dynamic oracles in constituency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 469–476, Melbourne, Australia. Association for Computational Linguistics.

Daniel Fried, Mitchell Stern, and Dan Klein. 2017. Improving neural parsing by disentangling model combination and reranking effects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 161–166, Vancouver, Canada. Association for Computational Linguistics.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? an analysis. 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 999–1010, New Orleans, Louisiana. Association for Computational Linguistics.

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.

David Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–237, Baltimore, Maryland. Association for Computational Linguistics.

Juneki Hong and Liang Huang. 2018. Linear-time constituency parsing with RNNs and dynamic programming. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 477–483, Melbourne, Australia. Association for Computational Linguistics.

Wenbin Jiang, Liang Huang, and Qun Liu. 2009. Automatic adaptation of annotation standards: Chinese word segmentation and POS tagging – a case study. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 522–530, Suntec, Singapore. Association for Computational Linguistics.

Vidur Joshi, Matthew Peters, and Mark Hopkins. 2018. Extending a parser to distant domains using a few dozen partially annotated examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1190–1199, Melbourne, Australia. Association for Computational Linguistics.

Tadao Kasami. 1966. An efficient recognition and syntax-analysis algorithm for context-free languages. *Coordinated Science Laboratory Report no. R-257*.

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.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

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

Jiangming Liu and Yue Zhang. 2017b. Shift-reduce constituent parsing with neural lookahead features. *Transactions of the Association for Computational Linguistics*, 5:45–58.

Lemao Liu, Muhua Zhu, and Shuming Shi. 2018. Improving sequence-to-sequence constituency parsing. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 4873–4880. AAAI Press.

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.

Xian Qian and Yang Liu. 2012. Joint Chinese word segmentation, POS tagging and parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 501–511, Jeju Island, Korea. Association for Computational Linguistics.

Sunita Sarawagi and W. William Cohen. 2004. Semi-markov conditional random fields for information extraction. In *Neural Information Processing Systems*, pages 1185–1192.

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.

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.

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.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 252–259.

David Vadas and James Curran. 2007. Adding noun phrase structure to the Penn Treebank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 240–247, Prague, Czech Republic. Association for Computational Linguistics.

David Vadas and James R. Curran. 2011. Parsing noun phrases in the Penn Treebank. *Computational Linguistics*, 37(4):753–809.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

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.

Mengqiu Wang, Kenji Sagae, and Teruko Mitamura. 2006. A fast, accurate deterministic parser for Chinese. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 425–432, Sydney, Australia. Association for Computational Linguistics.

Zhiguo Wang, Haitao Mi, and Nianwen Xue. 2015. Feature optimization for constituent parsing via neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1138–1147, Beijing, China. Association for Computational Linguistics.

Zhiguo Wang, Chengqing Zong, and Nianwen Xue. 2013. A lattice-based framework for joint Chinese word segmentation, POS tagging and parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 623–627, Sofia, Bulgaria. Association for Computational Linguistics.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based neural constituent parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1169–1179, Beijing, China. 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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Nat. Lang. Eng.*, 11(2):207238.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2013. Chinese parsing exploiting characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 125–134, Sofia, Bulgaria. 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-20*, pages 4046–4053. International Joint Conferences on Artificial Intelligence Organization. Main track.

Xiaoqing Zheng, Haoyuan Peng, Yi Chen, Pengjing Zhang, and Wenqiang Zhang. 2015. Character-based parsing with convolutional neural network. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1054–1060. AAAI Press.

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.