Character-Level Chinese Dependency Parsing via Modeling Latent Intra-Word Structure

Anonymous ACL submission

Abstract

Revealing the syntactic structure of sentences in Chinese poses significant challenges for word-level parsers due to the absence of clear word boundaries. To facilitate a transition from word-level to character-level Chinese dependency parsing, this paper proposes modeling latent internal structures within words. In this way, each word-level dependency tree is interpreted as a forest of character-level trees. A constrained Eisner algorithm is implemented to ensure the compatibility of character-level trees, guaranteeing a single root for intra-word structures and establishing inter-word dependencies between these roots. Experiments on Chinese treebanks demonstrate the superiority of our method over both the pipeline framework and previous joint models. A detailed analysis reveals that a coarse-to-fine parsing strategy empowers the model to predict more linguistically plausible intra-word structures.

1 Introduction

In the field of natural language processing, dependency parsing plays a crucial role in revealing the syntactic structure of sentences, thereby forming the foundation for numerous downstream applications such as machine translation (Shen et al., 2008; Wu et al., 2017), information extraction (Culotta and Sorensen, 2004; Gamallo et al., 2012), and sentiment analysis (Nakagawa et al., 2010; Sun et al., 2019).

This task, although straightforward in space-delimited languages, encounters significant challenges in languages like Chinese, where explicit word boundaries are absent. Traditional Chinese parsing methods rely heavily on word-level treebanks, necessitating the segmentation of text into distinct words before parsing. This prerequisite not only adds an additional layer of complexity but also makes the parsing outcome vulnerable to inaccuracies in segmentation.

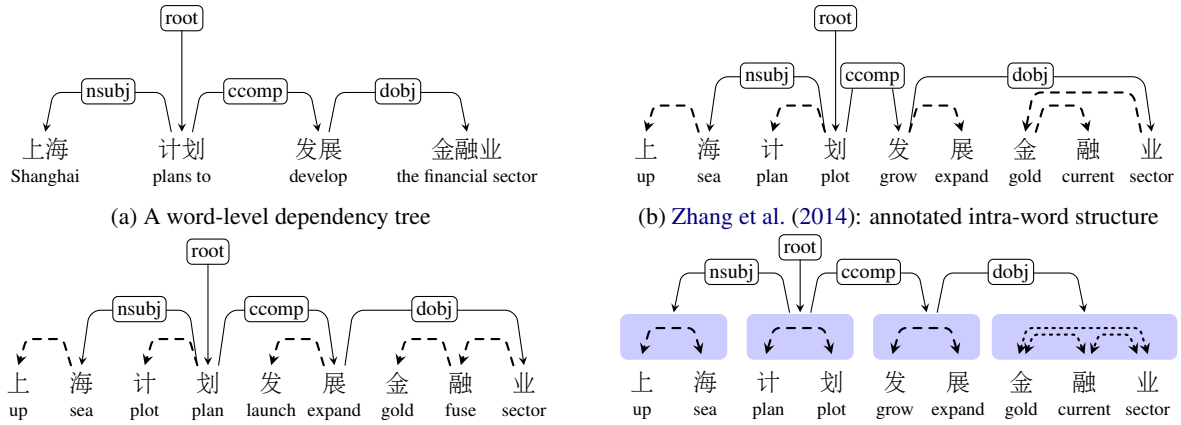
The need to address these issues has prompted a transition from word-level to character-level Chinese dependency parsing. However, the lack of character-level Chinese treebanks presents a challenge. As a workaround, researchers have endeavored to derive character-level dependency trees from word-level ones (Hatori et al., 2012; Zhang et al., 2014, 2015; Kurita et al., 2017; Li et al., 2018; Yan et al., 2020; Wu and Zhang, 2021).

Zhang et al. (2014) pioneered the integration of character- and word-level annotations. Figure 1 demonstrates a fully depicted character-level dependency tree (Figure 1b) by combining the word-level tree (Figure 1a) with annotated intra-word structures. Nonetheless, the application of this method is constrained by the non-trivial task of deriving linguistically coherent intra-word structures.

Some researchers have opted for a simpler approach by defining pseudo intra-word structures (Hatori et al., 2012; Yan et al., 2020). As illustrated in Figure 1c, these structures utilize a left-wavy pattern, with the rightmost character acting as the root and other characters headed by their right-adjacent characters. Although this method circumvents the labor-intensive annotation process, it may not accurately represent the syntactic roles of characters.

This paper proposes a new approach to character-level Chinese dependency parsing via modeling latent intra-word structure. As illustrated in Figure 1d, our approach allows for the implicit representation of all potential internal structures within words. For example, for the word "发展 (develop)", both "发 (grow)→展 (expand)" and "发 (grow)←展 (expand)" are acceptable structures. In this way, *each word-level dependency tree is interpreted as a forest of character-level trees.*

Central to our approach is a constrained Eisner algorithm (Eisner, 1996), crafted to maintain the compatibility of character-level trees it generates. This algorithm enforces two critical constraints: the *single-root subtree* constraint and the *root-as-*



(a) A word-level dependency tree

(b) Zhang et al. (2014): annotated intra-word structure

(c) Yan et al. (2020): pseudo leftward intra-word structure

(d) Ours: latent intra-word structure

Figure 1: A word-level dependency tree and corresponding character-level trees with three types of intra-word structure. Intra-word dependencies are represented by dashed arcs and their labels are omitted.

head constraint, which together guarantee that each word corresponds to a single-root subtree and that inter-word dependencies link to root characters of subtrees. Furthermore, we introduce a coarse-to-fine parsing strategy to refine the parsing process. Our primary contributions include:

- This work explores modeling latent intra-word structure for character-level Chinese dependency parsing.
- By implementing a novel, linguistically informed algorithm, the compatibility of character-level trees with their word-level counterparts is ensured.
- We devise a coarse-to-fine parsing strategy that improves parsing accuracy and generates more linguistically plausible intra-word structures.
- Experiment results on Chinese treebanks demonstrate that our approach outperforms both the pipeline model and previous joint models. Additionally, we provide insightful analyses of the predicted intra-word structures.

We will release our code on `Github`.

2 Parsing with Latent Structure

2.1 Word-level Tree to Char-level Forest

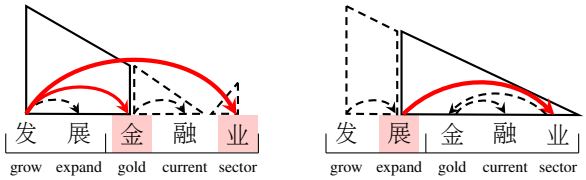
Latent Structure. To transform word-level trees into character-level trees, previous studies typically defined fixed internal structures for each word, either annotated by human experts (Zhang et al., 2014) or generated through rules (Yan et al., 2020). Our approach does not explicitly define intra-word structures. Instead, it allows for the representation of all possible internal structures within each word. This method acknowledges the multifaceted nature of language, where a single word may have multiple structures, especially for words with multiple parts of speech and coordinate characters (Gong et al., 2021). The implicit representation of intra-word structures empowers the model to identify the most plausible structure based on context.

Conversion. The latent nature of the intra-word structures facilitates a flexible construction of character-level dependencies, which are categorized into intra-word and inter-word for clarity. Within a given word, any two characters can form an intra-word dependency. Conversely, given a head-modifier pair, an inter-word dependency can originate from any characters in the head word to any characters in the modifier word. In this way, a word-level dependency tree can be interpreted as a forest comprising various potential character-level trees, as illustrated by the specific examples in Figures 1b and 1c for the forest in Figure 1d.

2.2 Compatibility: Two Constraints

The aforementioned conversion process is structurally sound, indicating there are no conflicts between character-level and word-level dependencies. However, ensuring the character-level trees faithfully represent both the internal structure of words and the syntactic relationships between them requires addressing compatibility issues. These issues, while not explicitly defined, adhere to certain linguistic principles. To this end, we introduce two constraints:

(1) The single-root subtree constraint. This constraint upholds the linguistic principle that each word corresponds to a single-root subtree within



(a) Two root characters: "金(gold)" and "业(sector)"

(b) Non-root character "展(expand)" as head

Figure 2: Two examples that violate compatibility constraints. The incorrect characters and arcs are highlighted in red. Triangles represent complete spans, while trapezoids represent incomplete spans. Dashed or solid lines are used to indicate intra-word or inter-word.

the character-level trees. It implies several aspects: (i) characters in the word form a subtree; (ii) there is a single, most important character representing the word, selected as the root of the subtree; (iii) all other characters are descendants of this root character; (iv) given the single-headed nature of dependency trees, the root character—and only the root character—can modify a character from another word, resulting in an inter-word dependency. An illustration showing a word erroneously assigned two root characters is provided in Figure 2a.

(2) The root-as-head constraint. While the single-root subtree constraint guarantees that only the root character can act as the modifier in an inter-word dependency, it is possible that a root character of one word modifies a non-root character in another word, as shown in Figure 2b. To accurately reflect the relation between intra-word structures, we require that only the root character of a word can serve as the head in an inter-word dependency.

The two constraints collectively assert that *a root character not only represents the central syntactic role of the word but also exclusively participates in forming inter-word dependencies*.

2.3 The Constrained Eisner Algorithm

This subsection elaborates on the implementation of two compatibility constraints using a modified version of the Eisner algorithm (Eisner, 1996).

Eisner algorithm. The Eisner Algorithm is a dynamic programming algorithm designed to find the highest-scoring dependency tree for a given sentence. It works by iteratively combining spans into larger spans and ultimately into a complete tree. The algorithm considers two types of spans: complete spans and incomplete spans. Complete spans comprise a head word and its descendants on one side. Incomplete spans encompass a dependency and the region between the head and modifier. Please refer to the examples in Figure 2.

Algorithm 1 Constrained Eisner Algorithm.

Input: arc scores $s(i,j)$, word-level tree T_w
▷ *arc scores conflicting with T_w are masked to $-\infty$* ◁
Define: $I, C \in \mathbf{R}^{n \times n}$
Initialize: $C_{i \to i} = 0, 1 \le i \le n$
for $w = 1, \ldots, n$ **do**
  for $i = 1, \ldots, n - w$ **do**
    $j = i + w$
    $I_{i \to j} = \max_{i \le k < j} (s(i,j) + C_{i \to k} + C_{k+1 \leftarrow j})$
    $I_{i \leftarrow j} = \max_{i \le k < j} (s(j,i) + C_{i \to k} + C_{k+1 \leftarrow j})$
    $C_{i \to j} = \max_{i < k \le j} (I_{i \to k} + C_{k \to j})$
    ▷ *j belongs to the right boundaries of the words* ◁
    ▷ *if (i,k) inside a word, (k,j) also inside this word* ◁
    $C_{i \leftarrow j} = \max_{i \le k < j} (C_{i \leftarrow k} + I_{k \leftarrow j})$
    ▷ *i belongs to the left boundaries of the words* ◁
    ▷ *if (k,j) inside a word, (i,k) also inside this word* ◁
return $C_{1 \to n}$

Given all scores of character-level dependencies, it is straightforward to obtain an optimal character-level tree using the Eisner algorithm. However, it is more complex to derive an optimal character-level tree that is *compatible* with a given word-level tree. To satisfy the compatibility constraints, we propose a constrained Eisner algorithm, presented in Algorithm 1.[1]

Constraint enforcement. To clarify the implementation of two constraints, we first differentiate spans into two types: intra-word and inter-word. Intra-word spans consist solely of intra-word dependencies, spanning either part or the entirety of a word. Inter-word spans contain at least one inter-word dependency, spanning multiple words. Please refer to the examples in Figure 2.

For the single-root subtree constraint, we observe that cases of multi-roots arise from inter-word complete spans including residual characters from a word (see Figure 2a for an example). Inspired by recent work (Zhang et al., 2021, 2022), we stipulate that inter-word complete spans must terminate at word boundaries.

For the root-as-head constraint, based on our observations, instances where non-characters become the heads of inter-word dependencies arise when combining an intra-word incomplete span with an inter-word complete span. An example is provided in Figure 2b. Therefore, we prohibit all such combination operations. To the best of our knowledge, *we are the first to address the root-as-head constraint*

[1] During training, a constrained Inside algorithm is used to enumerate all compatible character-level trees.

in graph-based dependency parsing.

The implementation of two constraint rules is straightforward by using auxiliary mask tensors. The additional time complexity is $O(n^3)$ but becomes negligible when accelerated by GPUs.

2.4 A Coarse-to-Fine Parsing Strategy

In the absence of the word-level trees, determining the intra-word and inter-word roles for a dependency in the character-level trees is not straightforward. As the Eisner algorithm conflates two distinct roles, determining these roles can only occur after the arc labeling step (described in Section 3), potentially resulting in the cases that an intra-word dependency arc overlays an inter-word dependency arc.[2] These illegal arcs hinder the recovery from character-level trees to word-level trees (see Appendix A.1 for details).

To ensure the validity of output trees, we propose a coarse-to-fine parsing strategy, explicitly assigning each arc two scores for intra-word and inter-word roles. The core idea is to first construct intra-word spans and then inter-word spans, thus ensuring that intra-word dependency arcs underlie the inter-word dependency arcs. The deduction rules are depicted in Figure 3. We refer interested readers to Algorithm 2 in the appendix for details.

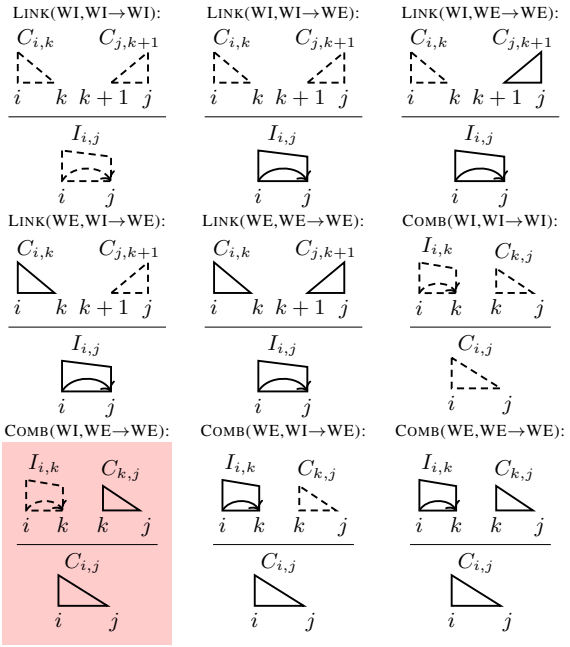


Figure 3: Deduction rules for coarse-to-fine parsing. Dashed or solid lines are used to indicate intra-word spans (WI) or inter-word spans (WE). The highlighted rule can be ignored to satisfy the *root-as-head* constraint. We present only R-rules, omitting the symmetric L-rules and initial conditions for brevity.

3 Model

Notations. Given a sentence $\boldsymbol{x} = c_0, c_1, \ldots, c_n$, where c_i represents the ith character of $\boldsymbol{x}$ and c_0 denotes an artificial ROOT token, a labeled dependency tree for $\boldsymbol{x}$ is denoted as $\boldsymbol{t}$. We view $\boldsymbol{t}$ as a set of labeled dependency arcs, using $(i, j, l) \in \boldsymbol{t}$ to indicate an arc from character c_i to c_j with a label $l \in \mathcal{L}$, where $\mathcal{L}$ is the set of dependency labels.[3] Additionally, an unlabeled dependency tree is denoted as $\boldsymbol{y}$ and an unlabeled dependency arc is denoted as (i, j).

3.1 Parsing Modeling

Adhering to Dozat and Manning (2017), we employ a two-stage parsing framework that first predicts unlabeled trees and then labels the arcs in these trees. The score of an unlabeled dependency tree is the cumulative sum of its unlabeled arc scores:

$$s(\boldsymbol{x}, \boldsymbol{y}) = \sum_{(i,j)\in\boldsymbol{y}} s(i, j) \tag{1}$$

The conditional probability of a unlabeled tree $\boldsymbol{y}$ is defined as:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{e^{s(\boldsymbol{x},\boldsymbol{y})}}{\boldsymbol{Z}(\boldsymbol{x}) \equiv \sum_{\boldsymbol{y'}\in\mathcal{Y}(x)} e^{s(\boldsymbol{x},\boldsymbol{y'})}} \tag{2}$$

where $\boldsymbol{Z}(\boldsymbol{x})$ is known as the partition term, and $\mathcal{Y}(x)$ denotes the set of all possible (projective) trees for $\boldsymbol{x}$.

Forest probability. The forest, denoted as $\mathcal{F}$, comprises dependency trees that meet compatibility constraints. The probability of $\mathcal{F}$ is the aggregate probability of each tree $\boldsymbol{y}$ within $\mathcal{F}$.

$$\begin{aligned} p(\mathcal{F}|\boldsymbol{x}) &= \sum_{\boldsymbol{y}\in\mathcal{F}} p(\boldsymbol{y}|\boldsymbol{x}) \\ &= \frac{\boldsymbol{Z}(\boldsymbol{x}, \boldsymbol{\mathcal{F}}) \equiv \sum_{\boldsymbol{y}\in\mathcal{F}} e^{s(\boldsymbol{x},\boldsymbol{y})}}{\boldsymbol{Z}(\boldsymbol{x})} \end{aligned} \tag{3}$$

where $\boldsymbol{Z}(\boldsymbol{x}, \boldsymbol{\mathcal{F}})$ can be computed via a constrained Inside algorithm, by substituting the max-product in Algorithm 1 with the sum-product.

3.2 Training

During training, the loss for a sentence $\boldsymbol{x}$ is composed of two parts: (unlabeled) tree loss and label

[2] Under ideal conditions, all intra-word arcs should underlie the inter-word arcs, as depicted in Figure 1d.

[3] An additional INTRA label is used to indicate the intra-word dependency arcs.

loss.

$$L(\boldsymbol{x}) = L^{\text{tree}}(\boldsymbol{x}) + L^{\text{label}}(\boldsymbol{x}) \quad (4)$$

Tree loss. Given a sentence $\boldsymbol{x}$, the tree loss is naturally defined as the negative log-probability of the forest $\mathcal{F}$:

$$L^{\text{tree}}(\boldsymbol{x}) = -\log p(\mathcal{F}|\boldsymbol{x}) \quad (5)$$

Label loss. The probability of assigning label l to an unlabeled arc (i, j) is defined as:

$$p(l|i,j) = \frac{e^{s(i,j,l)}}{\sum_{l' \in \mathcal{L}} e^{s(i,j,l')}} \quad (6)$$

The label loss is the sum of negative log probabilities of correctly labeling each arc in the forest $\mathcal{F}$.[4]

$$L^{\text{label}}(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathcal{F}} \sum_{(i,j) \in \boldsymbol{y}} -\log p(l|i,j) \quad (7)$$

3.3 Inference

To parse a sentence $\boldsymbol{x}$, the model first selects the highest-scoring unlabeled tree $\hat{\boldsymbol{y}}$ via (vanilla) Eisner algorithm.

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} s(\boldsymbol{x}, \boldsymbol{y}) \quad (8)$$

Subsequently, the optimal label for each arc $(i, j) \in \hat{\boldsymbol{y}}$ is determined.

$$\hat{l} = \arg\max_{l \in \mathcal{L}} s(i, j, l) \quad (9)$$

3.4 Network Architecture

Encoding. The sentence $\boldsymbol{x}$ is directly input into the pre-trained BERT model, and the output from the last layer is used as the representation of characters.

$$\ldots, \mathbf{h}_i, \cdots = \mathbf{BERT}(\ldots, c_i, \ldots) \quad (10)$$

Scoring. To score dependency arcs, we utilize the biaffine attention mechanism as outlined by Dozat and Manning (2017). In the coarse-to-fine parsing, intra- and inter-word arcs are scored separately through distinct biaffine attentions. More details are provided in Appendix A.3.

4 Experiments

Data. We conduct experiments on three versions of the Penn Chinese Treebank (CTB): CTB5, CTB6, and CTB7.[5] The split of train, development, and test sets follows established practices (Zhang and Clark, 2010; Yang and Xue, 2012; Wang et al., 2011). Table 5 in the appendix provides detailed statistics. The conversion from phrase structures to dependency structures is performed using two methods: (1) the Stanford parser v3.3.0[6] with Stanford Dependencies (SD) (de Marneffe et al., 2006); (2) the Penn2Malt tool[7] with the head-finding rules as described by Zhang and Clark (2008), henceforth referred to as Z&C. Only projective trees are retained during training. An intra-word structure dataset annotated by Gong et al. (2021) on CTB5 is utilized for experiments and analysis.[8]

Evaluation metrics. For Chinese word segmentation (CWS), we employ standard F1 measures (F1_{seg}). For dependency parsing, evaluation is conducted at the word level, using word-level F1 scores (UF_{dep} and LF_{dep}) as the evaluation metrics (Yan et al., 2020). A dependency arc is considered correct only if the head-modifier word pair is correctly segmented. Punctuation is excluded during the evaluation of dependency parsing.

Baseline and proposed models. The evaluation includes the following models:

- **TreeCRF**: A word-level biaffine parsing model with a CRF loss, detailed in Zhang et al. (2020).
- **Pipeline**: This framework first performs CWS by assigning 'BMES' tags to characters and then feeds the segmented results into **TreeCRF**.
- **Leftward**: A model uses pseudo leftward intra-word structures as described by Yan et al. (2020).
- **Latent**: The proposed model uses latent intra-word structures. The constrained Eisner algorithm is used to ensure compatibility.
- **Latent-c2f**: Enhancing **Latent** with a coarse-to-fine parsing strategy, as described in Section 2.4.

Results using pseudo rightward structures and annotated structures are provided in Appendix B.1.

Hyper-parameters. All models utilize the "bert-base-chinese"[9] as the encoder to obtain contextual representations. For word-level models, word representations are derived by averaging the corresponding character representations. The configuration of the scoring layer adheres to Zhang et al.

[4] Refer to Appendix A.2 for the enumeration of these arcs.

[5] https://catalog.ldc.upenn.edu/LDC2010T07

[6] https://nlp.stanford.edu/software/lex-parser.shtml

[7] https://cl.lingfil.uu.se/~nivre/research/Penn2Malt.html

[8] https://github.com/SUDA-LA/wist

[9] https://huggingface.co/bert-base-chinese

Model	CTB5			CTB6			CTB7		
	$F1_{seg}$	UF_{dep}	LF_{dep}	$F1_{seg}$	UF_{dep}	LF_{dep}	$F1_{seg}$	UF_{dep}	LF_{dep}
w/ head-finding rules of SD									
Yan et al. (2020)	98.46	89.59	85.94	–	–	–	97.06	85.06	80.71
Pipeline	98.72	90.93	88.39	97.23	87.09	83.86	97.16	85.77	82.00
Leftward	98.76	90.91	88.37	97.30	87.21	84.04	**97.22**	85.85	82.17
Latent (*Ours*)	98.76	**91.06**	**88.49**	97.28	87.22	84.03	97.17	85.74	82.04
Latent-c2f (*Ours*)	**98.79**	90.95	88.34	**97.33**	**87.30**	**84.12**	**97.22**	**85.90**	**82.23**
w/ head-finding rules of Z&C									
Hatori et al. (2012)†	97.75	81.56	–	95.45	74.88	–	95.42	73.58	–
Zhang et al. (2014)†	97.67	81.63	–	95.63	76.75	–	95.53	75.63	–
Zhang et al. (2015)†	98.04	82.01	–	–	–	–	–	–	–
Kurita et al. (2017)†	98.37	81.42	–	–	–	–	95.86	74.04	–
Wu and Zhang (2021)‡	98.57	–	90.38	97.32	–	86.49	97.25	–	84.68
Pipeline	98.72	92.00	91.04	97.23	87.71	86.77	97.16	86.39	85.19
Leftward	98.67	91.98	91.03	**97.39**	88.02	87.06	**97.26**	86.48	85.30
Latent (*Ours*)	98.74	**92.16**	**91.25**	97.37	87.99	87.06	97.22	86.50	85.33
Latent-c2f (*Ours*)	**98.77**	92.03	91.08	97.37	**88.10**	**87.14**	**97.26**	**86.55**	**85.37**

Table 1: Results on CTB5, CTB6, and CTB7 test sets. The best results are in bold. † indicates using additional POS tag information. ‡: Wu and Zhang (2021) consider a dependency arc correct even if the head word is wrongly segmented; thus, the reported results are not directly comparable to ours.

(2020). Refer to Appendix A.4 for detailed hyper-parameter settings and optimization procedures. All results are averaged over four runs with different random seeds.

4.1 Main Results

Comparison with the pipeline framework. As shown in Table 1, our latent models (Latent and Latent-c2f) consistently outperform the pipeline model across all metrics, except for LF_{dep} on CTB5 using SD, where Latent-c2f is lower by 0.05%. Latent-c2f achieves absolute improvements of 0.27% and 0.37% in LF_{dep} score on CTB6 across two dependency representations. Similar improvements are observed on CTB5 and CTB7. The results demonstrate the efficacy of our proposed latent parsing method in mitigating the error propagation problem.

Comparison with previous joint models. Table 1 also compares our method against previous joint models. The majority of prior models rely on traditional discrete features or static embeddings, resulting in performance lag compared to our latent models. The exception is Yan et al. (2020), which utilizes pre-trained BERT. Nevertheless, our latent models achieve substantial improvements, e.g., a 1.52% increase in LF_{dep} on CTB7.

Notably, Leftward can be considered a reimplementation of Yan et al. (2020), employing the same network architecture and hyper-parameter settings as our latent models. In comparison, Latent achieves comparable parsing performance and Latent-c2f achieves better parsing performance.

Parsing with gold-standard segmentation. To isolate the impact of word segmentation errors on parsing performance, we also conduct experiments using gold-standard segmentation, employing attachment score metrics (UAS and LAS).

As shown in Table 2, character-level models lag behind the word-level model (TreeCRF) by a significant margin, except for Latent on CTB5.[10] Among character-level models, Latent-c2f significantly enhances the performance of Latent on CTB7 and two latent models consistently outperform Leftward. This suggests that our latent models possess a superior ability in identifying head characters of words, and *enforcing the rightmost character as*

[10]This discrepancy may be attributed to the utilization of word-level information. Unlike word-level models that can directly utilize word representations, character-level models are merely aware of word boundaries.

Model	CTB5		CTB7	
	UAS	LAS	UAS	LAS
w/ head-finding rules of SD				
TreeCRF	92.83	90.14	**90.14**	**85.89**
Leftward	92.69	89.91	89.08	84.77
Latent (*Ours*)	**92.99**	**90.19**	89.29	84.99
Latent-c2f (*Ours*)	92.84	89.99	89.69	85.45
w/ head-finding rules of Z&C				
TreeCRF	93.95	92.90	**90.52**	**89.16**
Leftward	93.77	92.70	89.67	88.26
Latent (*Ours*)	**93.96**	**93.00**	89.86	88.46
Latent-c2f (*Ours*)	93.88	92.89	90.06	88.70

Table 2: Results using gold-standard segmentation on CTB5 and CTB7 test sets. Best results are in bold.

Model	CTB7			
	$F1_{seg}$	UF_{dep}	LF_{dep}	CM
w/ head-finding rules of SD				
Latent-c2f	**97.12**	**85.59**	**81.87**	**29.22**
- single-root	96.89	85.31	81.57	28.16
- root-as-head	97.07	85.52	81.79	28.59
- both constraints	96.80	85.21	81.48	27.26

Table 3: Ablation study on CTB7 dev set. "CM": Complete match of labeled dependency trees.

the word head may not be the best practice.

4.2 Analysis

Impact of proposed constraints. Ablation studies are conducted to investigate the individual and combined effects of the single-root subtree and root-as-head constraints. Complete match (CM) scores for the entire dependency tree are also provided. The removal of both constraints, as shown in Table 3, results in the lowest LF_{dep} score. The individual application of each constraint is less effective than using both constraints together. Notably, the absence of the single-root subtree constraint leads to a more significant decline in performance. This is justified by the fact that the single-root subtree constraint minimizes the segmentation of words into disjoint parts. The application of the root-as-head constraint alone offers a modest 0.08% improvement in LF_{dep} but leads to a substantial 0.63% increase in CM. The results indicate that *an accurate representation of intra-word structures and their syntactic relationships is beneficial for parsing performance and tree completeness.*

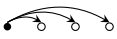
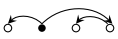
Structure	Latent		Latent-c2f		Annt.
	SD	Z&C	SD	Z&C	
	99.52	99.70	49.32	50.26	48.07
	0.48	0.30	50.68	49.74	51.93
	91.34	91.32	41.04	42.39	34.67
	0.02	0.02	4.67	1.64	34.20
	0.01	0.00	40.20	37.64	1.87
	54.06	55.34	10.07	9.28	7.24
	0.00	0.00	21.33	30.83	0.07
	2.77	2.66	1.78	0.98	15.45
	0.00	0.00	5.27	2.48	7.20

Table 4: Distribution of intra-word structures predicted by our latent models on CTB6 test set. "Annt." denotes annotated structures. Only high-frequency structures are provided. Filled dots represent root characters.

Distribution of predicted intra-word structures. A unique feature of our method is its capacity to infer complex intra-word structures. We assess the distribution of predicted structures, grouping them by word length to evaluate common patterns.[11] We focus on words of two, three, and four characters, as longer words are infrequent. A reference distribution of annotated structures by Gong et al. (2021) is also provided. High-frequency structures are shown in Figure 4. A comprehensive overview is available in Table 6 in the appendix.

For Latent, a prevalent left-wavy pattern emerges across words of varying lengths. Latent-c2f alleviates this leftward bias. For two-character words, the left-headed and right-headed structures in Latent-c2f are balanced, closely aligning with the annotated ones. For three- and four-character words, Latent-c2f can predict right-branched structures, which are seldom or never observed in Latent.

The leftward bias in Latent deserves further discussion. The Latent model, employing the Eisner algorithm, does not distinctly differentiate between intra- and inter-word dependencies. Consequently, this conflation unintentionally transfers the arc direction bias from the inter-word dependencies—derived from word-level trees—to the inherently latent intra-word dependencies. Given

[11]The complete match evaluation is presented in Appendix B.2.

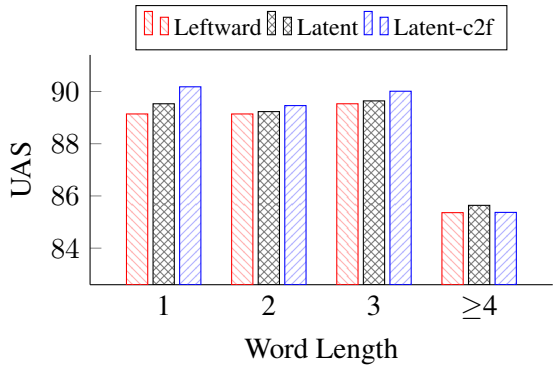


Figure 4: The unlabeled attachment score (UAS) for words of different lengths on CTB7 test set using SD.

that Chinese is a left-branching language, CTB exhibits a predominant occurrence of leftward arcs over rightward ones, with a distribution of 60% on SD and 70% on Z&C. The Latent-c2f model utilizes dual biaffine attention mechanisms for scoring dependencies, which serves to selectively filter arc direction information, thereby mitigating the inherent leftward bias observed in Latent.

Performance across word lengths. We further investigate the performance of character-level models across words of different lengths. The results in Figure 4 are obtained using gold-standard segmentation. Latent-c2f exhibits the best performance for words of lengths 1, 2, and 3. However, for words with a length greater than or equal to 4, Latent-c2f performs worse than Latent, suggesting that coarse-to-fine parsing may not be advantageous for longer words. Interestingly, the performance difference between Leftward and Latent is marginal for words of length 2 and 3. This is consistent with the information in Table 4, where intra-word structures of lengths 2 and 3 primarily exhibit a left-wavy pattern for Latent, nearly identical to Leftward.

5 Related Work

Intra-word structure. Zhao (2009) were the first to explore intra-word structures in Chinese through unlabeled dependency forms. Li (2011) and Zhang et al. (2013) extended this work by introducing constituency trees to depict these structures, which were further refined by Zhang et al. (2014) through their conversion of constituency trees into dependency trees. Gong et al. (2021) went on to investigate intra-word (labeled) dependencies, positioning the parsing of these structures as a distinct task.

Character-level dependency parsing. The area of character-level dependency parsing, especially within the context of Chinese, has undergone significant evolution. Hatori et al. (2012) led the initial efforts by introducing a transition-based parser that leveraged pseudo intra-word structures. This was followed by Zhang et al. (2014), who integrated annotated intra- and inter-word dependencies. Subsequent studies aimed to enhance the transition-based parsers with neural networks (Kurita et al., 2017; Li et al., 2018). Yan et al. (2020) were the first to adopt the graph-based parsing approach.

Span constraints. The dependency structure is closely related to spans (not limited to phrases and words). Spitkovsky et al. (2010) demonstrated how naturally annotated spans could be transformed into dependency structures by applying various parsing constraints. For transition-based parsers, Nivre et al. (2014) emphasized the necessity of a single-root subtree over the input spans. Similarly, Zhang et al. (2022) framed span-based semantic role labeling as dependency parsing, enforcing semantic arguments corresponding to single-root subtrees.

6 Conclusion

This paper explores modeling latent intra-word structures for character-level Chinese dependency parsing. Our approach, underpinned by the constrained Eisner algorithm, ensures the compatibility of constructed character-level trees. The incorporation of a coarse-to-fine parsing strategy further enhances the effectiveness and rationality of the parsing process. Our experiments and detailed analyses reveal the following findings:

- Our method outperforms not only the pipeline model but also previous joint models in character-level Chinese dependency parsing.
- Given gold-standard segmentation, our latent models, especially the coarse-to-fine one, demonstrate superior capability in identifying the head character of a word, suggesting that designating the rightmost character as the head of the word may not be optimal.
- The proposed compatibility constraints can improve both parsing accuracy and the completeness of tree structures.
- The intra-word structures predicted by the latent model tend to exhibit a left-wavy shape. The coarse-to-fine strategy alleviates the leftward bias and produces structures more aligned with manually annotated ones.

7 Limitations

Projectivity. Our method treats intra-word structures as latent, offering a flexible and rich representation of internal word structures. However, it operates within the confines of projective parsing due to the inherent nature of the Eisner algorithm. This constraint might limit the applicability of the model in accurately parsing non-projective trees.

Computational Efficiency. The introduction of constraints into the Eisner algorithm undoubtedly increases its complexity. Although auxiliary tensors and GPU utilization help mitigate the additional time burden, computational efficiency remains a concern, particularly as the necessity to calculate inside scores twice doubles the training duration. Moreover, the incorporation of a coarse-to-fine strategy, while beneficial for parsing accuracy, further compounds the computational demands.

8 Ethics Statement

We are committed to upholding high ethical standards throughout this paper. Our research focuses on Chinese dependency parsing, utilizing the Penn Chinese Treebank (LDC2010T07) for experimental purposes. We have obtained the necessary permissions and licenses for the acquisition of the data, and we strictly adhere to the terms of use associated with it. Researchers with access to the treebank can replicate our experiments using our provided code. Moreover, the annotated intra-word structures used for analysis are openly accessible and do not impose any acquisition or usage requirements. We believe that the utilization of these datasets will not compromise the confidentiality or integrity of individuals, nor will it contain offensive content. Additionally, given that our work primarily explores syntactic methodologies, we do not foresee any potential risks associated with our research.

References

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 423–429, Barcelona, Spain.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. 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.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Pablo Gamallo, Marcos Garcia, and Santiago Fernández-Lanza. 2012. Dependency-based open information extraction. In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP*, pages 10–18, Avignon, France. Association for Computational Linguistics.

Chen Gong, Saihao Huang, Houquan Zhou, Zhenghua Li, Min Zhang, Zhefeng Wang, Baoxing Huai, and Nicholas Jing Yuan. 2021. An in-depth study on internal structure of Chinese words. 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 5823–5833, Online. Association for Computational Linguistics.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2012. Incremental joint approach to word segmentation, POS tagging, and dependency parsing in Chinese. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1045–1053, Jeju Island, Korea. Association for Computational Linguistics.

Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305, Prague, Czech Republic. Association for Computational Linguistics.

Shuhei Kurita, Daisuke Kawahara, and Sadao Kurohashi. 2017. Neural joint model for transition-based Chinese syntactic analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1204–1214, Vancouver, Canada. Association for Computational Linguistics.

Haonan Li, Zhisong Zhang, Yuqi Ju, and Hai Zhao. 2018. Neural character-level dependency parsing for chinese. In *Proceedings of AAAI*, pages 5205–5212.

Zhongguo Li. 2011. Parsing the internal structure of words: A new paradigm for Chinese word segmentation. In *Proceedings of the 49th Annual Meeting of*

the Association for Computational Linguistics: Human Language Technologies, pages 1405–1414, Portland, Oregon, USA. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using CRFs with hidden variables. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 786–794, Los Angeles, California. Association for Computational Linguistics.

Joakim Nivre, Yoav Goldberg, and Ryan McDonald. 2014. Squibs: Constrained arc-eager dependency parsing. *Computational Linguistics*, 40(2):249–257.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, pages 577–585, Columbus, Ohio. Association for Computational Linguistics.

Valentin I. Spitkovsky, Daniel Jurafsky, and Hiyan Alshawi. 2010. Profiting from mark-up: Hyper-text annotations for guided parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1278–1287, Uppsala, Sweden. Association for Computational Linguistics.

Kai Sun, Richong Zhang, Samuel Mensah, Yongyi Mao, and Xudong Liu. 2019. Aspect-level sentiment analysis via convolution over dependency tree. 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 5679–5688, Hong Kong, China. Association for Computational Linguistics.

Ke M. Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. 2016. Unsupervised neural hidden Markov models. In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 63–71, Austin, TX. Association for Computational Linguistics.

Yiou Wang, Jun'ichi Kazama, Yoshimasa Tsuruoka, Wenliang Chen, Yujie Zhang, and Kentaro Torisawa. 2011. Improving Chinese word segmentation and POS tagging with semi-supervised methods using large auto-analyzed data. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 309–317, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Linzhi Wu and Meishan Zhang. 2021. Deep graph-based character-level chinese dependency parsing. *TASLP*, 29:1329–1339.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 698–707, Vancouver, Canada. Association for Computational Linguistics.

Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2020. A graph-based model for joint Chinese word segmentation and dependency parsing. *Transactions of the Association for Computational Linguistics*, 8:78–92.

Yaqin Yang and Nianwen Xue. 2012. Chinese comma disambiguation for discourse analysis. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 786–794, Jeju Island, Korea. Association for Computational Linguistics.

Liwen Zhang, Ge Wang, Wenjuan Han, and Kewei Tu. 2021. Adapting unsupervised syntactic parsing methodology for discourse dependency 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 5782–5794, Online. Association for Computational Linguistics.

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.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. Character-level Chinese dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1326–1336, Baltimore, Maryland. Association for Computational Linguistics.

Yu Zhang, Zhenghua Li, and Min Zhang. 2020. Efficient second-order TreeCRF for neural dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3295–3305, Online. Association for Computational Linguistics.

Yu Zhang, Qingrong Xia, Shilin Zhou, Yong Jiang, Guohong Fu, and Min Zhang. 2022. Semantic role labeling as dependency parsing: Exploring latent tree structures inside arguments. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4212–4227, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Yuan Zhang, Chengtao Li, Regina Barzilay, and Kareem Darwish. 2015. Randomized greedy inference for joint segmentation, POS tagging and dependency parsing. In *Proceedings of the 2015 Conference of*

the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 42–52, Denver, Colorado. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 562–571, Honolulu, Hawaii. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2010. A fast decoder for joint word segmentation and POS-tagging using a single discriminative model. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 843–852, Cambridge, MA. Association for Computational Linguistics.

Hai Zhao. 2009. Character-level dependencies in Chinese: Usefulness and learning. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 879–887, Athens, Greece. Association for Computational Linguistics.

A Implementation Details

A.1 Char-Tree to Word-Tree Recovery

After predicting an optimal character-level tree, a word-level tree can be recovered from it. The first step is to identify all subtrees corresponding to words, which must satisfy two conditions: (1) contain only intra-word dependency arcs (indicated by an INTRA label); (2) be linked by an inter-word dependency arc (indicated by common syntactic labels). Next, these subtrees are collapsed into words. Finally, the character-level inter-word arcs are revived into word-level arcs.

A.2 Loss Function

To calculate the label loss, we need to enumerate each arc in each tree in the forest, which is exponential in the worst case. Inspired by Zhang et al. (2022), we find this enumeration can be integrated into the computation of the tree loss.

First, we define the probability of assigning the labels to all arcs in the unlabeled tree $\boldsymbol{y}$ as:

$$p(\boldsymbol{r}|\boldsymbol{x},\boldsymbol{y}) = \prod_{(i,j)\in\boldsymbol{y}} p(l|i,j) \tag{11}$$

where $\boldsymbol{r}$ is the set of labels for all arcs in $\boldsymbol{y}$.

Then, we define the probability of the labeled tree $\boldsymbol{t}$ of a given sentence $\boldsymbol{x}$ as:

$$p(\boldsymbol{t}|\boldsymbol{x}) = p(\boldsymbol{y}|\boldsymbol{x}) \cdot p(\boldsymbol{r}|\boldsymbol{x},\boldsymbol{y}) \tag{12}$$

Dataset	Train	Dev.	Test
CTB5	18,104	352	348
CTB6	23,420	2,079	2,796
CTB7	31,112	10,043	10,292

Table 5: Data statistics. We present the number of sentences in the training, development, and test sets.

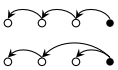
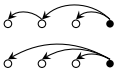
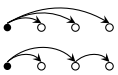
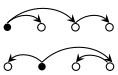
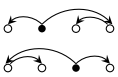
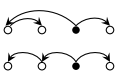
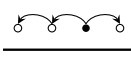
Structure	Latent		Latent-c2f		Annt.
	SD	Z&C	SD	Z&C	
	99.52	99.70	49.32	50.26	48.07
	0.48	0.30	50.68	49.74	51.93
	91.34	91.32	41.04	42.39	34.67
	0.02	0.02	4.67	1.64	34.20
	7.03	8.10	8.15	7.31	5.78
	0.01	0.00	40.20	37.64	1.87
	0.02	0.01	2.96	9.14	7.02
	0.96	0.19	2.37	1.60	15.30
	54.06	55.34	10.07	9.28	7.24
	12.37	22.33	9.28	12.08	9.39
	0.44	1.98	18.72	14.79	0.57
	0.00	0.00	21.33	30.83	0.07
	0.04	0.00	6.19	3.92	11.94
	2.77	2.66	1.78	0.98	15.45
	0.00	0.00	5.27	2.48	7.20
	0.32	0.20	0.20	0.00	7.13

Table 6: Distribution of intra-word structures predicted by our latent models on the CTB6 test set. "Annt." denotes annotated structures. Filled dots represent root characters.

Finally, the loss function is defined as the negative log-likelihood of the labeled forest $\mathcal{T}$:

$$\begin{aligned} L(\boldsymbol{x}) &= -\log p(\mathcal{T}|\boldsymbol{x}) \\ p(\mathcal{T}|\boldsymbol{x}) &= \sum_{\boldsymbol{t}\in\mathcal{T}} p(\boldsymbol{t}|\boldsymbol{x}) \\ &= \frac{\sum_{\boldsymbol{y}\in\mathcal{F}} e^{s(\boldsymbol{x},\boldsymbol{y})} \cdot p(\boldsymbol{r}|\boldsymbol{x},\boldsymbol{y})}{\boldsymbol{Z}(\boldsymbol{x})} \\ &= \frac{\sum_{\boldsymbol{y}\in\mathcal{F}} \prod_{(i,j)\in\boldsymbol{y}} e^{s(i,j)+\log p(l|i,j)}}{\boldsymbol{Z}(\boldsymbol{x})} \end{aligned} \tag{13}$$

By adding the log probability of labels to the arc scores, the label loss is naturally integrated into the tree loss via the constrained Inside algorithm.

Algorithm 2 Coarse-to-fine Eisner Algorithm.

Input: intra-word arc scores $\hat{s}(i,j)$ and inter-word arc scores $s(i,j)$
Define: $\hat{I}, I, \hat{C}, C \in \mathbf{R}^{n \times n}$ ▷ *The hat symbol denotes an intra-word span*
Initialize: $\hat{C}_{i \to i} = 0, C_{i \to i} = -\infty, 1 \leq i \leq n$
for $w = 1, \ldots, n$ **do**
  for $i = 1, \ldots, n - w$ **do**
    $j = i + w$
    $\hat{I}_{i \to j} = \max_{i \leq k < j} (\hat{s}(i,j) + \hat{C}_{i \to k} + \hat{C}_{k+1 \leftarrow j})$
    $I_{i \to j} = \max_{i \leq k < j} (s(i,j) + \hat{C}_{i \to k} + \hat{C}_{k+1 \leftarrow j}, s(i,j) + \hat{C}_{i \to k} + C_{k+1 \leftarrow j},$
      $s(i,j) + C_{i \to k} + \hat{C}_{k+1 \leftarrow j}, s(i,j) + C_{i \to k} + C_{k+1 \leftarrow j})$
    $\hat{I}_{i \leftarrow j} = \max_{i \leq k < j} (\hat{s}(j,i) + \hat{C}_{i \to k} + \hat{C}_{k+1 \leftarrow j})$
    $I_{i \leftarrow j} = \max_{i \leq k < j} (s(j,i) + \hat{C}_{i \to k} + \hat{C}_{k+1 \leftarrow j}, s(j,i) + \hat{C}_{i \to k} + C_{k+1 \leftarrow j},$
      $s(j,i) + C_{i \to k} + \hat{C}_{k+1 \leftarrow j}, s(j,i) + C_{i \to k} + C_{k+1 \leftarrow j})$
    $\hat{C}_{i \to j} = \max_{i < k \leq j} (\hat{I}_{i \to k} + \hat{C}_{k \to j})$
    $C_{i \to j} = \max_{i < k \leq j} (\hat{I}_{i \to k} + C_{k \to j}, I_{i \to k} + \hat{C}_{k \to j}, I_{i \to k} + C_{k \to j})$
    $\hat{C}_{i \leftarrow j} = \max_{i \leq k < j} (\hat{C}_{i \leftarrow k} + \hat{I}_{k \leftarrow j})$
    $C_{i \leftarrow j} = \max_{i \leq k < j} (C_{i \leftarrow k} + \hat{I}_{k \leftarrow j}, \hat{C}_{i \leftarrow k} + I_{k \leftarrow j}, C_{i \leftarrow k} + I_{k \leftarrow j})$
return $C_{1 \to n}$

A.3 Coarse-to-fine Scoring

To score a dependency arc $i \to j$, we first feed the output from encoder $\mathbf{h}$ into two MLPs to obtain the representations of character as head and modifier. Then, to distinguish the intra-word and inter-word roles, the arc is scored by two different biaffine layers.

$$\begin{aligned}
\mathbf{h}_i^{(arc-head)} &= \mathbf{MLP}^{(arc-head)}(\mathbf{h}_i) \\
\mathbf{h}_j^{(arc-mod)} &= \mathbf{MLP}^{(arc-mod)}(\mathbf{h}_j) \\
s^{(intra)}(i,j) &= \mathbf{h}_i^{(arc-head)} W^{(intra)} \mathbf{h}_j^{(arc-mod)} \\
s^{(inter)}(i,j) &= \mathbf{h}_i^{(arc-head)} W^{(inter)} \mathbf{h}_j^{(arc-mod)}
\end{aligned} \tag{14}$$

A.4 Hyper-parameter Details

We utilize the default parameter configurations for pre-trained BERT and directly fine-tune the entire model. The configuration of the scoring layer adheres to Zhang et al. (2020). We employ AdamW (Loshchilov and Hutter, 2019) for parameter optimization with $\beta_1 = 0.9$, $\beta_2 = 0.9$, $\epsilon = 1 \times 10^{-12}$, and weight decay of 0. The learning rate is set to 5×10^{-5} for the encoder and 1×10^{-3} for the scorer. The dropout rate is set to 0.1 for the encoder and 0.33 for the scorer. We train the model for 10 epochs with 1,000 tokens per batch.

B Supplementary Results

B.1 Additional Models

Two additional models are included in the comparison, employing different strategies to handle the internal structures of words:

- **Rightward:** A model uses pseudo intra-word structures in a right-wavy pattern, which is similar to the leftward pattern but in the opposite direction.
- **Annotated:** A model uses annotated intra-word structures by Gong et al. (2021). If no annotated structure is available for a word, the latent structure is employed.

The results are presented in Table 7. Rightward achieves performance similar to Leftward. Specifically, it performed slightly better on SD but slightly worse on Z&C. Surprisingly, Annotated only achieves comparable performance to Pipeline. Comparing Annotated and Latent, the use of annotated structures does not improve performance and even degrades it. This finding is consistent with Wu and Zhang (2021), who observed that using annotated structures by Zhang et al. (2014) is detrimental to neural dependency parsers. Two points

Model	CTB5			CTB6			CTB7		
	$F1_{seg}$	UF_{dep}	LF_{dep}	$F1_{seg}$	UF_{dep}	LF_{dep}	$F1_{seg}$	UF_{dep}	LF_{dep}
w/ head-finding rules of SD									
Pipeline	98.72	90.93	88.39	97.23	87.09	83.86	97.16	85.77	82.00
Annotated	98.68	90.74	88.05	97.30	87.10	83.87	97.17	85.70	81.95
Leftward	98.76	90.91	88.37	97.30	87.21	84.04	97.22	85.85	82.17
Rightward	98.76	90.83	88.24	**97.35**	**87.34**	**84.12**	**97.23**	85.89	**82.23**
Latent (*Ours*)	98.76	**91.06**	**88.49**	97.28	87.22	84.03	97.17	85.74	82.04
Latent-c2f (*Ours*)	**98.79**	90.95	88.34	97.33	87.30	**84.12**	97.22	**85.90**	**82.23**
w/ head-finding rules of Z&C									
Pipeline	98.72	92.00	91.04	97.23	87.71	86.77	97.16	86.39	85.19
Annotated	98.66	91.85	90.92	97.34	87.87	86.90	97.23	86.35	85.17
Leftward	98.67	91.98	91.03	**97.39**	88.02	87.06	**97.26**	86.48	85.30
Rightward	98.72	91.65	90.71	97.33	87.84	86.90	**97.26**	86.46	85.28
Latent (*Ours*)	98.74	**92.16**	**91.25**	97.37	87.99	87.06	97.22	86.50	85.33
Latent-c2f (*Ours*)	**98.77**	92.03	91.08	97.37	**88.10**	**87.14**	**97.26**	**86.55**	**85.37**

Table 7: Results on CTB5, CTB6, and CTB7 test sets. The best results are in bold.

Model	CM	$CM_{M\text{-}1}$
Latent (SD)	42.86	44.20
Latent (Z&C)	42.77	44.11
Latent-c2f (SD)	**44.26**	**85.00**
Latent-c2f (Z&C)	42.41	84.36

Table 8: Complete match (CM) of intra-word structures on CTB6 test set.

can be concluded from the results:

- Both leftward and rightward intra-word structures are effective for the joint CWS and dependency parsing task.
- The usefulness of annotated structures in the deep learning era is questionable and deserves further investigation.

B.2 Complete Match of Structures

In addition to investigating the distribution of intra-word structures, we utilize the complete match (CM) metric to evaluate the performance of our latent models in predicting intra-word structures. The complete match measures the percentage of words with correct whole structures. Here, we refer to the intra-word structures annotated by Gong et al. (2021) as the gold standard. We calculate the average of the results from four seed models. Additionally, since no gold-standard structures are employed during training, the evaluation can be regarded as unsupervised. Following studies on unsupervised POS tagging (Johnson, 2007; Tran et al., 2016), we employ a many-to-one (M-1) mapping to align the predicted structures with the gold standard. Specifically, if any predicted structure by a seed model matches the gold standard, it is considered a complete match. The results are shown in Table 8. Compared to Latent, Latent-c2f achieves a similar CM score but higher M-1 mapping results. This is because Latent-c2f favors leftward arcs in some seed models and rightward arcs in others. When employing a many-to-one mapping, more structures predicted by Latent-c2f align with their gold-standard counterparts.

B.3 Model Focus on Optimal Structure Prediction

During inference, our method enables investigation into the feasibility of all possible internal structures within each word. Considering the challenge of enumerating all structures, we assess the model focus on each arc using marginal probabilities. We analyze the sentence "金/杯子/的/白开水" (The clear water in a golden cup) using the Latent-c2f model as a case study. Figure 5 displays the probability matrix for this sentence. For the word "杯子 (cup)", the model confidently identifies a dependency from character "杯 (cup)" to character "子 (child)". However, the model struggles to determine the head character of "白 (white)" in the word "白开水 (clear water)". This case study demon-

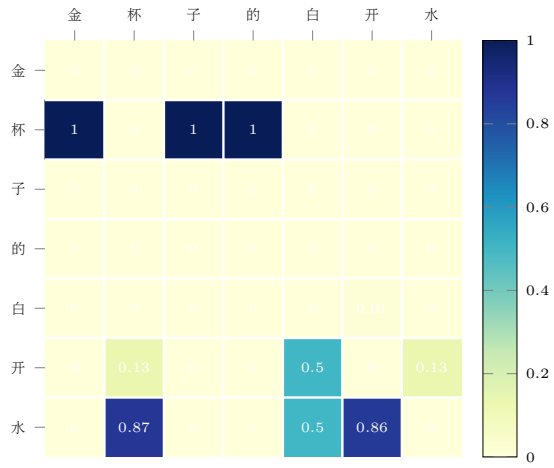


Figure 5: The probability matrix for the sentence "金/杯子/的/白开水" (The clear water in a golden cup). The cell (i, j) corresponds to an arc from i to j.

strates that the model is very certain about the optimal structure for some words but still leaves room for predictions on other structures.