Semantic Role Labeling as Dependency Parsing: Exploring Latent Tree Structures Inside Arguments

Yu Zhang[☆], **Qingrong Xia**[☆☾], **Shilin Zhou**[☆], **Yong Jiang**[☼], **Guohong Fu**[☆*], **Min Zhang**[☆]
[☆]Institute of Artificial Intelligence, School of Computer Science and Technology,
Soochow University, Suzhou, China
[☾]Huawei Cloud, China
[☼]DAMO Academy, Alibaba Group, China
{yzhang.cs,slzhou.cs}@outlook.com; xiaqingrong@huawei.com
yongjiang.jy@alibaba-inc.com; {ghfu,minzhang}@suda.edu.cn

Abstract

Semantic role labeling (SRL) is a fundamental yet challenging task in the NLP community. Recent works of SRL mainly fall into two lines: 1) BIO-based; 2) span-based. Despite ubiquity, they share some intrinsic drawbacks of not considering internal argument structures, potentially hindering the model's expressiveness. The key challenge is arguments are flat structures, and there are no determined subtree realizations for words inside arguments. To remedy this, in this paper, we propose to regard flat argument spans as latent subtrees, accordingly reducing SRL to a tree parsing task. In particular, we equip our formulation with a novel span-constrained TreeCRF to make tree structures span-aware and further extend it to the second-order case. We conduct extensive experiments on CoNLL05 and CoNLL12 benchmarks. Results reveal that our methods perform favorably better than all previous syntax-agnostic works, achieving new state-of-the-art under both *end-to-end* and *w/ gold predicates* settings.

1 Introduction

Semantic role labeling (SRL) is a fundamental yet challenging task in the NLP community, involving predicate and argument identification, as well as semantic role classification. As SRL can provide informative linguistic representations, it has been widely adopted in downstream tasks like question answering (Berant et al., 2013; Yih et al., 2016), information extraction (Christensen et al., 2010; Lin et al., 2017), and machine translation (Liu and Gildea, 2010; Bazrafshan and Gildea, 2013), etc.

Recent works of SRL mainly fall into two lines: 1) BIO-based; 2) span-based. The former views SRL as a sequence labeling task (Zhou and Xu, 2015; Strubell et al., 2018; Shi and Lin, 2019). For each predicate, each token is tagged with a label starting with BIO prefixes indicating if it is at the **B**eginning, **I**nside, or **O**utside of an argument.

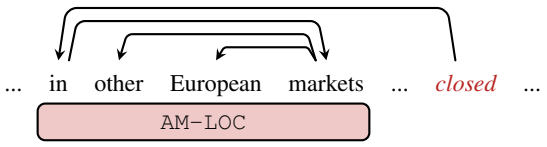


Figure 1: An argument example (below) and its related subtree structure (above) for the predicate "*closed*".

The latter (He et al., 2018a; Ouchi et al., 2018; Li et al., 2019), in contrast, opts to jointly predict all predicate and argument span pairs using a span-graph formulation.

Despite ubiquity, there are some drawbacks that limit the expressiveness of the two methods. First, framing predicate-argument structures as a BIO-tagging scheme is less effective as it lacks explicit modeling of span-level representations, so that long adjacencies of argument phrases can be ignored (Cohn and Blunsom, 2005; Jie and Lu, 2019; Zhou et al., 2020d; Xu et al., 2021). Second, span-based method seeks to pick very few (typically $<10\%$) positive examples from $O(n^3)$ candidate predicate-argument pairs, thus suffering from severe class imbalance problem (Li et al., 2021). To alleviate this issue, span-based method relies on heavy pruning (He et al., 2018a) to reduce the searching space, potentially impairing the performance.

Meanwhile, both formulations share some common flaws in terms of lacking explicit modeling of internal argument structures, which appear to be beneficial to SRL. Taking Fig. 1 as an example, internal dependencies of words ("in other European markets") inside the span provide strong clues for recognizing it as a locative modifier ("AM-LOC") of the predicate "*closed*". Besides, the predicate-argument relation can be naturally reflected by the dependency from the predicate to the span head-word ("*closed* $\xrightarrow{\text{AM-LOC}}$ in"), and we can properly recognize the argument span boundaries by retrieving all descendants of the subtree. Such observa-

*Corresponding author

Proceedings of the 29th International Conference on Computational Linguistics, pages 4212–4227
October 12–17, 2022.

tions have motivated many attempts on utilizing relations inside arguments (Gildea and Hockenmaier, 2003; Johansson and Nugues, 2008a,c; Xia et al., 2019; Li et al., 2019, *inter alia*). However, stuck on the fact that span-style SRL has no determined internal structure realizations, existing works have to resort to making use of external human-annotated syntax knowledge to bridge the gap (Shi et al., 2020; Li et al., 2021).

Our main goal in this work is to explicitly take internal argument structures into account meanwhile keeping our framework *end-to-end*. To this end, we propose to model flat arguments as latent subtrees, thus paving the way for reducing SRL to dependency parsing seamlessly: *we view predicate-argument structures as partially-observed trees where exact subtrees for each argument are not realized yet.* In this way, we reframe span-style SRL as parsing word-to-word relations by encoding all predicate-argument relations into a unified dependency graph. Unlike span-based methods (He et al., 2018a), a dependency graph contains no more than $O(n^2)$ possible dependencies, so that the class imbalance issue can be side-stepped effortlessly. Specifically, we make use of TreeCRF (Eisner, 2000; Zhang et al., 2020), which provides a viable way for probabilistic modeling of tree structures, to learn the partially-observed trees and marginalize the latent structures out during training. Unlike canonical TreeCRF, which enumerates all possible trees, in our setting, we have to impose many span constraints to reflect the argument boundaries on subtrees correctly. To accommodate this, we further design a novel span-constrained TreeCRF to adapt it to our learning procedure, which explicitly prohibits invalid edges across different arguments as well as multi-head subtrees (Nivre et al., 2014; Zhang et al., 2021a).

There are further advantages to our reduction. Conversion to tree structures enables us to easily conduct global optimization (Eisner, 1996; McDonald et al., 2005) in polynomial time, which has already been shown to often lead to improved results and more meaningful predictions (Toutanova et al., 2008; Täckström et al., 2015; FitzGerald et al., 2015; Li et al., 2020) compared to local unconstrained methods. On the other hand, by drawing on the experience in the parsing literature, we can further extend our method to some well-studied high-order methods (McDonald and Pereira, 2006) without any obstacle. We experiment with sibling factors in this work and find significant gains, in line with many parsing works (Zhang et al., 2020; Fonseca and Martins, 2020). Our contributions can be summarized as follows:[1]

- Aware of the benefits of internal argument structures, we propose to model flat argument spans as latent subtrees, thereby reducing SRL to dependency parsing seamlessly.
- We propose a novel span-constrained TreeCRF to learn the converted trees and further extend it to the second-order case.
- Experiments on CoNLL05 and CoNLL12 benchmarks reveal that our proposed methods outperform existing works significantly, achieving new state-of-the-art results under the syntax-agnostic setting.

2 Overview

In span-style SRL, an argument of a predicate corresponds to one word or multiple continuous words. In the latter case, each word in the argument span is treated as equal, and the internal structure of a multi-word argument, i.e., the relationship between words inside the argument, is usually overlooked due to the lack of corresponding annotations.

In this work, we propose to explicitly model internal structures of multi-word arguments and treat arguments as latent subtrees. Our approach deals with each predicate separately, and assumes each corresponds to a single-root tree. Consequently, each argument subtree is attached to the predicate. During the training process, all possible structures are enumerated and accumulated to compose the argument representation. While decoding, we seek to find a 1-best tree and recover arguments from the subtrees belonging to the resulting structure. We highlight four key points.

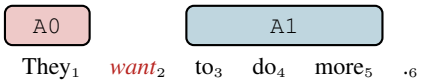
i Our approach is syntax-agnostic. The tree structures are modeled and predicted solely to serve the SRL task without referring to any linguistic syntax knowledge.

ii The predicate identification subtask is handled as a simple classification procedure.

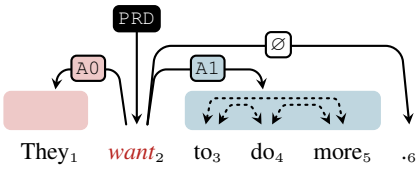
iii For argument identification, argument boundaries are decided by subtrees attached to the predicate, and edge labels are used for role disambiguation.

iv We adopt a consistent scoring architecture for the two subtasks and train them jointly.

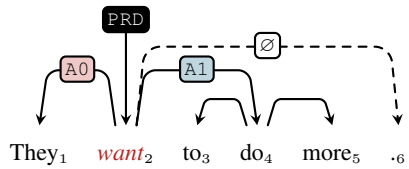
[1]Our code is publicly available at https://github.com/yzhangcs/crfsrl.



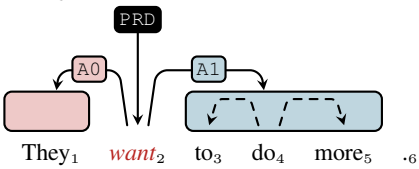
(a) Original structure: arguments of the predicate are located in the upper half-plane of the sentence and do not overlap with each other.



(b) Training: convert the predicate-argument structure to a dependency tree with (dotted) latent annotations; non-argument spans are assigned "∅" for distinction.



(c) Decoding: realize a tree rooted at the predicate with the arc labeled as "PRD"; (dashed) arcs labeled as "∅" are discarded.



(d) Recovery: collapse all (dashed) subtrees governed by the predicate into flat argument spans.

Figure 2: Illustration of our SRL→Tree conversion (Fig. 2a and Fig. 2b), and its inverse Tree→SRL process (Fig. 2c and Fig. 2d). We emphasize the predicate "*want*" in the figures for clarity. The two arguments with roles "A0" and "A1" are framed by red and blue rectangles, respectively.

2.1 SRL → Tree Conversion

Formally, given an input sentence $\boldsymbol{x} = x_1, \ldots, x_n$, we first seek to obtain tree structures for each predicate $p \in \boldsymbol{x}$, which are taken as materials of training a parser. We define a directed acyclic dependency tree $\boldsymbol{t}$ by assigning a head $h \in \{x_0, x_1, \ldots, x_n\}$ together with a relation label $r \in \mathcal{R}$ to each modifier $m \in \boldsymbol{x}$, where a dummy word x_0 is attached before $\boldsymbol{x}$ as the pseudo root node.[2]

For predicate p, the first step is to link x_0 to p. To facilitate predicate identification, we assign a special label PRD (resp. ∅ for non-predicate) to the dependency $x_0 \rightarrow p$. Then, we make all corresponding latent argument subtrees descendants of p. As we showcase in Fig. 2a, this takes advantage of the non-overlapping constraint for arguments belonging to the same predicate (Punyakanok et al., 2004; Li et al., 2019). For an argument with a consecutive word span $x_i, \ldots, x_j$ and a semantic role $r \in \mathcal{R}$, we restrict all possible subtrees are single-rooted at a potential headword h within the span, which is also not realized yet. The semantic role r is assigned as the label of the dependency pointing from p to the headword. We adopt a similar strategy for non-argument spans, except that we set the label to ∅ for distinction and remove the single-root restriction.

By enumerating all possible subtrees and combing them together, the resulting tree set T_p is exponential in size. During training, we develop a span-constrained Inside algorithm to perform the enumeration (§ 3.2). Fig. 2b gives a brief example of the conversion process.

2.2 Tree → SRL Recovery

Supposing we have trained a parsing model, during the decoding phase, what we need is to recover predicate-argument structures from the outputs of the parser.

We first find all predicates via simple local label classification: a word p is recognized as a predicate if the dependency $x_0 \rightarrow p$ is labeled as PRD. Subsequently, we obtain the highest-scoring tree $\boldsymbol{t}^*$ (Fig. 2c) for p using Eisner algorithm (Eisner, 2000) with complexity $O(n^3)$:

$$\boldsymbol{t}^* = \arg \max_{\boldsymbol{t}: x_0 \xrightarrow{\text{PRD}} p \in \boldsymbol{t}} \mathrm{s}(\boldsymbol{x}, \boldsymbol{t}) \quad (1)$$

where $\mathrm{s}(\boldsymbol{x}, \boldsymbol{t})$ is the tree score, and the tree is restricted to be rooted at p. Arguments for the predicate are then recovered by collapsing subtrees headed by p into flat spans.

Concretely, we take each modifier h of p as the headword of a potential argument. If the label r of $p \rightarrow h$ is not "∅", i.e., non-argument, then an entire argument span comprises h and its descendants and takes r as the semantic role. The resulting SRL output is the collection of all predicates and corresponding recovered arguments. A recovery example is demonstrated in Fig. 2d.

[2] In this work, we assume all dependency trees are *projective*, i.e., without any crossing arcs. This property allows us to associate the subtree with its continuous argument span (Kong et al., 2015).

3 Methodology

Now we elaborate the architecture of our proposed model for training the parser. Following Dozat and Manning (2017); Zhang et al. (2020), our model consists of a contextualized encoder and a (second-order) scoring module. We further propose a span-aware TreeCRF to compute the probabilities of the converted partially-observed trees.

3.1 Neural Parameterization

Given the sentence $\boldsymbol{x} = x_0, x_1, \ldots, x_n$, we first obtain the hidden representation of each token x_i via a deep contextualized encoder.

$$\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_n = \texttt{Encoder}(x_0, x_1, \ldots, x_n) \quad (2)$$

In this work, we experiment with two alternative encoders, i.e., BiLSTMs (Gal and Ghahramani, 2016) and pretrained language models (PLMs) (Devlin et al., 2019). More setting details are available in § A.

(Second-order) Tree parameterization Following Dozat and Manning (2017), we decompose a tree $\boldsymbol{t}$ into two separate $\boldsymbol{y}$ and $\boldsymbol{r}$, where $\boldsymbol{y}$ is a skeletal tree, and $\boldsymbol{r}$ is the related strictly-ordered label sequence. For each head-modifier pair $h \rightarrow m \in \boldsymbol{y}$, we score them using two MLPs followed by a Biaffine layer (Cai et al., 2018):

$$\begin{aligned} \mathbf{r}_i^{\text{head/mod}} &= \texttt{MLP}^{\text{head/mod}}(\mathbf{h}_i) \\ \mathrm{s}(h \rightarrow m) &= \texttt{BiAF}\left(\mathbf{r}_h^{\text{head}}, \mathbf{r}_m^{\text{mod}}\right) \end{aligned} \quad (3)$$

The score of the dependency $h \rightarrow m$ with label $r \in \mathcal{R}$ is calculated analogously. We use two extra MLPs and $|\mathcal{R}|$ Biaffine layers to obtain all label scores.

We also make use of adjacent-sibling information (McDonald and Pereira, 2006) to enhance the first-order biaffine parser further. Following Wang et al. (2019); Zhang et al. (2020), we employ three extra MLPs as well as a Triaffine layer for second-order subtree scoring,

$$\begin{aligned} \mathbf{r}_i^{\text{head/mod/sib}} &= \texttt{MLP}^{\text{head/mod/sib}}(\mathbf{h}_i) \\ \mathrm{s}(h \rightarrow s, m) &= \texttt{TriAF}\left(\mathbf{r}_h^{\text{head}}, \mathbf{r}_m^{\text{mod}}, \mathbf{r}_s^{\text{sib}}\right) \end{aligned} \quad (4)$$

where s and m are two adjacent modifiers of h, and s populates between h and m.

Under the first-order factorization (McDonald et al., 2005), the score of $\boldsymbol{y}$ becomes

$$\mathrm{s}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{h \rightarrow m \in \boldsymbol{y}} \mathrm{s}(h \rightarrow m) \quad (5)$$

For the second-order case (McDonald and Pereira, 2006), we further incorporate adjacent-sibling subtree scores into tree scoring:

$$\mathrm{s}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{h \rightarrow m} \mathrm{s}(h \rightarrow m) + \sum_{h \rightarrow s, m} \mathrm{s}(h \rightarrow s, m) \quad (6)$$

The probabilities of skeletal tree $\boldsymbol{y}$ and its label sequence $\boldsymbol{r}$ are parameterized as

$$\begin{aligned} P(\boldsymbol{y} \mid \boldsymbol{x}) &= \frac{\exp\left(\mathrm{s}(\boldsymbol{x}, \boldsymbol{y})\right)}{Z(\boldsymbol{x}) \equiv \sum_{\boldsymbol{y}' \in Y(\boldsymbol{x})} \exp\left(\mathrm{s}(\boldsymbol{x}, \boldsymbol{y}')\right)} \\ P(\boldsymbol{r} \mid \boldsymbol{x}, \boldsymbol{y}) &= \prod_{h \xrightarrow{r} m \in \boldsymbol{t}} P(r \mid \boldsymbol{x}, h \rightarrow m) \end{aligned} \quad (7)$$

$Y(\boldsymbol{x})$ is the set of all possible legal unlabeled trees, and $Z(\boldsymbol{x})$ is known as the partition function. Each label r is independent of tree $\boldsymbol{y}$ and other labels, thus $P(r \mid \boldsymbol{x}, h \rightarrow m)$ is locally normalized over all $r' \in \mathcal{R}$.

Finally, we define the probability of the labeled tree $\boldsymbol{t}$ as the product of the probabilities of its two sub-components.

$$P(\boldsymbol{t} \mid \boldsymbol{x}) = P(\boldsymbol{y} \mid \boldsymbol{x}) \cdot P(\boldsymbol{r} \mid \boldsymbol{x}, \boldsymbol{y}) \quad (8)$$

3.2 Span-constrained TreeCRF

Training objective During training, we seek to maximize the probability of converted trees T_p for each predicate p. Accordingly, we define the following loss function:

$$\mathcal{L} = -\sum_p \log P(T_p \mid \boldsymbol{x}) \quad (9)$$

in which $P(T_p \mid \boldsymbol{x})$ can be further expanded as

$$\begin{aligned} P(T_p \mid \boldsymbol{x}) &= \sum_{\boldsymbol{t} \in T_p} \underbrace{P(\boldsymbol{y} \mid \boldsymbol{x}) \cdot P(\boldsymbol{r} \mid \boldsymbol{x}, \boldsymbol{y})}_{P(\boldsymbol{t} \mid \boldsymbol{x})} \\ &= \frac{1}{Z(\boldsymbol{x})} \sum_{\boldsymbol{t} \in T_p} \underbrace{\exp(\mathrm{s}(\boldsymbol{x}, \boldsymbol{y})) \cdot P(\boldsymbol{r} \mid \boldsymbol{x}, \boldsymbol{y})}_{\exp(\mathrm{s}(\boldsymbol{x}, \boldsymbol{t}))} \end{aligned} \quad (10)$$

The proposed Inside The calculation of the denominator $Z(\boldsymbol{x})$ in Eq. 10 can be accomplished by the canonical Inside algorithm. As for the numerator, we make a slight change of the formula and define the labeled tree score as:

$$\mathrm{s}(\boldsymbol{x}, \boldsymbol{t}) = \mathrm{s}(\boldsymbol{x}, \boldsymbol{y}) + \log P(\boldsymbol{r} \mid \boldsymbol{x}, \boldsymbol{y}) \quad (11)$$

In this way, the numerator is exactly the summation of exponential scores of all legal labeled trees.[3]

[3] It is noteworthy that we do not assign any label to $h \rightarrow m \in \boldsymbol{y}$ while $h \notin \{x_0, p\}$, i.e., any dependency inside an argument span, thus its logarithmic label probability is set to 0 and does not contribute to tree scoring.

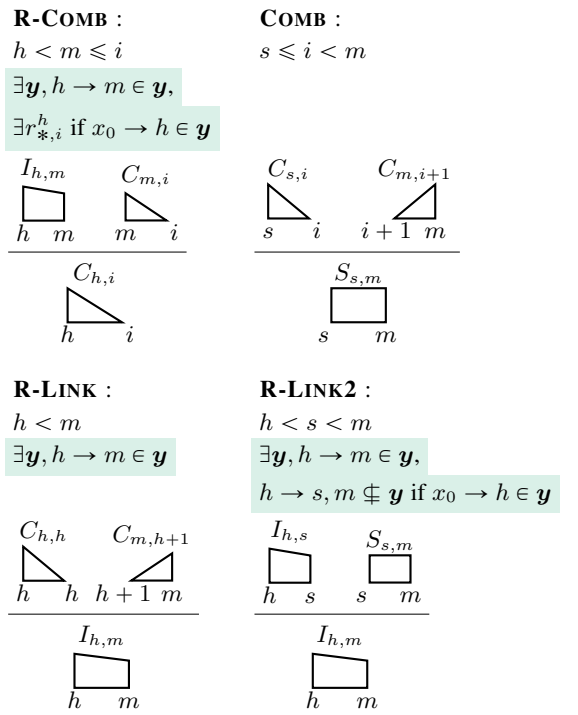


Figure 3: Deduction rules for our span-constrained Inside algorithm (**R-COMB** and **R-LINK**) and its second-order extension (**COMB** and **R-LINK2**). Our modified rule constraints are highlighted in *green color*. The condition $x_0 \rightarrow h \in \boldsymbol{y}$ means h is a predicate with x_0 as the parent. $r^{h}_{*,i}$ denotes an argument span that takes h as the predicate and ends with i. We show only R-rules, omitting the symmetric L-rules as well as initial conditions for brevity.

This differs from the traditional case of partial tree learning (Li et al., 2016) from two perspectives where the common Inside algorithm is not adequate to: 1) we impose span constraints to force the converted latent subtrees to reflect argument spans, and 2) we require the subtree ought to be single-rooted at one potential headword in the span.

To resolve this, in this work, we propose a span-constrained Inside algorithm to accommodate these constraints. We illustrate the deduction rules (Pereira and Warren, 1983) of our tailored algorithm and its second-order extension in Fig. 3.[4] Basically, we avoid the arc $h \rightarrow m$ crossing different argument spans by prohibiting merging to the relevant incomplete span $I_{h,m}$ (**R-LINK**). To prevent multiple headwords in the same argument, inspired by Zhang et al. (2021a), for predicate h, we only allow merging to the complete span $C_{h,i}$ if i is at the endpoint of an argument (**R-COMB**).

[4]We refer interested readers to § B for more details on the exact meaning of the operations in the Inside algorithm.

	#Train	#Dev	#Test	#OOD	#roles
CoNLL05	39,832	1,346	2,416	2,399	54
CoNLL12	75,187	9,603	9,479	-	63

Table 1: Data statistics for CoNLL05 and CoNLL12 datasets.

For the second-order case, we further prohibit the subtree $h \rightarrow s, m$ once s and m are located in the same argument (**R-LINK2**), since this case implies that the argument can be split into two more smaller headed spans with respect to s and m, which is not what we expect.

Time complexity analysis The proposed span-constrained Inside shares the same asymptotic time complexity of $O(n^3)$ as its canonical counterpart (Eisner, 2000). Besides, we draw on the recent development of parallelization techniques (Eisner, 2016; Zhang et al., 2020; Rush, 2020) and further reduce the complexity of the parallelized algorithm to $O(n^2)$ on GPUs. In practice, we find that our models are efficient enough compared to BIO-based and Span-based models. We make comprehensive speed comparisons in § 4.3.

4 Experiments

We measure our proposed first-order CRF and second-order CRF2O models on two SRL benchmarks: CoNLL05 and CoNLL12. Full implementation details are given in § A.

Data Table 1 lists the statistics of the datasets. For CoNLL05, we follow standard splits of Carreras and Màrquez (2005): sections 02-21 of WSJ corpus as Train data, section 24/23 as Dev/Test data, and three sections (CK01-03) of the Brown corpus as out-of-domain (OOD) data. For CoNLL12, following He et al. (2018a), we extract data from OntoNotes (Pradhan et al., 2013) and follow the data splits of the CoNLL12 shared task (Pradhan et al., 2012).[5] We adopt the same splits for both *end-to-end* and *w/ gold predicates* settings. We use the official scripts provided by CoNLL05 shared task[6] for evaluation.

4.1 Main results

Table 2 gives our main results. By default, our models work in an *end-to-end* fashion, i.e., predicting

[5]The list of file IDs for Train/Dev/Test data is available on the task webpage.

[6]https://www.cs.upc.edu/~srlconll

	CoNLL05							CoNLL12			
	Dev	WSJ			Brown			Dev	Test		
	F_1	P	R	F_1	P	R	F_1	F_1	P	R	F_1
He et al. (2017)	80.3	80.2	82.3	81.2	67.6	69.6	68.5	75.5	78.6	75.1	76.8
He et al. (2018a)	81.6	81.2	83.9	82.5	69.7	71.9	70.8	79.4	79.4	80.1	79.8
Li et al. (2019)	-	-	-	83.0	-	-	-	-	-	-	-
Zhou et al. (2020a)	82.27	-	-	-	-	-	-	-	-	-	-
CRF	83.70	83.18	85.38	84.27	70.40	72.97	71.66	81.03	**79.47**	82.80	81.10
CRF2O	**83.91**	**83.26**	**86.20**	**84.71**	**70.70**	**74.16**	**72.39**	**81.16**	79.27	**83.24**	**81.21**
Li et al. (2019)$_{\text{ELMo}}$	-	85.2	87.5	86.3	74.7	78.1	76.4	-	**84.9**	81.4	83.1
Zhou et al. (2022)$_{\text{BERT}}$	86.79	**87.15**	88.44	87.79	**79.44**	80.85	80.14	84.74	83.91	85.61	84.75
CRF$_{\text{BERT}}$	86.82	86.98	88.28	87.63	79.19	80.92	80.05	85.35	84.47	86.24	85.35
CRF2O$_{\text{BERT}}$	**87.03**	87.00	**88.76**	**87.87**	79.08	**81.50**	**80.27**	**85.53**	84.53	**86.41**	**85.45**
CRF$_{\text{RoBERTa}}$	87.31	87.20	88.67	87.93	79.29	81.48	80.38	86.08	84.98	86.86	85.91
CRF2O$_{\text{RoBERTa}}$	**87.46**	**87.35**	**89.34**	**88.33**	**79.95**	**82.32**	**81.12**	**86.34**	**85.30**	**87.02**	**86.15**
w/ gold predicates											
He et al. (2017)	81.6	83.1	83.0	83.1	72.9	71.4	72.1	81.5	81.7	81.6	81.7
Ouchi et al. (2018)	82.5	84.7	82.3	83.5	**76.0**	70.4	73.1	82.9	**84.4**	81.7	83.0
Tan et al. (2018)	83.1	84.5	85.2	84.8	73.5	74.6	74.1	82.9	81.9	83.6	82.7
Strubell et al. (2018)	-	84.7	84.24	84.47	73.89	72.39	73.13	-	-	-	-
Zhou et al. (2020a)	83.16	-	-	-	-	-	-	-	-	-	-
Zhang et al. (2021b)	84.45	85.30	85.17	85.23	74.98	73.85	74.41	82.83	83.09	83.71	83.40
CRF	84.42	85.38	85.56	85.47	75.05	74.05	74.55	83.22	83.21	83.85	83.53
CRF2O	**84.65**	**85.47**	**86.40**	**85.93**	74.92	**75.00**	**74.96**	**83.39**	83.02	**84.31**	**83.66**
Strubell et al. (2018)$_{\text{ELMo}}$	85.26	86.21	85.98	86.09	77.1	75.61	76.35	83.23	84.39	82.21	83.28
Shi and Lin (2019)$_{\text{BERT}}$	-	88.6	89.0	88.8	81.9	82.1	82.0	-	85.9	87.0	86.5
Jindal et al. (2020)$_{\text{BERT}}$	-	88.7	88.0	87.9	80.3	80.1	80.2	-	86.3	86.8	86.6
Zhang et al. (2021b)$_{\text{BERT}}$	87.38	87.70	88.15	87.93	81.52	81.36	81.44	86.27	86.00	86.84	86.42
Zhou et al. (2022)$_{\text{BERT}}$	87.54	**89.03**	88.53	88.78	**83.22**	81.81	82.51	86.97	87.26	87.05	87.15
Conia and Navigli (2020)$_{\text{BERT}}$	-	-	-	-	-	-	-	-	86.9	87.7	87.3
Blloshmi et al. (2021)$_{\text{BART}}$	-	-	-	-	-	-	-	-	**87.8**	86.8	87.3
CRF$_{\text{BERT}}$	87.76	88.93	88.58	88.76	82.87	81.67	82.27	87.33	87.45	87.56	87.51
CRF2O$_{\text{BERT}}$	**88.05**	89.00	**89.03**	**89.02**	82.81	**82.35**	**82.58**	**87.52**	87.52	**87.79**	**87.66**
CRF$_{\text{RoBERTa}}$	88.21	89.29	88.99	89.15	83.22	82.42	82.82	87.97	87.99	88.22	88.11
CRF2O$_{\text{RoBERTa}}$	**88.49**	**89.45**	**89.63**	**89.54**	**83.89**	**83.39**	**83.64**	**88.29**	**88.11**	**88.53**	**88.32**

Table 2: Results on CoNLL05 and CoNLL12 data. All results are averaged over 4 runs with different random seeds.

all predicates and their associated arguments simultaneously. However, we note that reporting the results of using gold predicates is a more prevalent practice in the SRL community (He et al., 2018a; Shi and Lin, 2019). Therefore, for comprehensive comparisons, in addition to listing most *end-to-end* results of previous works we are aware of, we also conduct experiments with gold predicates, which is achieved by only parsing trees rooted at the pre-specified predicates.[7]

The two major rows show the results of *end-to-end* and *w/ gold predicates* settings, indicating very consistent trends. We can clearly see that under the *end-to-end* setting, our LSTM-based CRF models outperform previous works by a large margin on all datasets. The second-order CRF2O further improves over CRF by 0.2, 0.4 and 0.7 F_1 scores on three CoNLL05 datasets, respectively. On CoNLL12, CRF2O shows smaller but steady gains. As revealed in § 4.2, we attribute the improvements brought by CRF and CRF2O to better performing at global consistency and long-range dependencies.

The results under the *w/ gold predicates* setting are presented in the second major row. Many PLM-based results comparable to ours are available in this setting. Among them, the BIO-based parser of Shi and Lin (2019) achieves 88.8, 82.0 and 86.5 F_1 scores on CoNLL05 WSJ, Brown and CoNLL12 Test data. The dependency (word)-based parser of Zhou et al. (2022) achieves 88.78, 82.51 and 87.15 F_1 scores. Meanwhile, the results of our first-order CRF model with BERT is $88.76_{\pm 0.18}$, $82.27_{\pm 0.26}$ and $87.51_{\pm 0.11}$. The performance gap between CRF and recent state-of-the-art parsers are negligibly small. We note that we do not uti-

[7]We eliminate the invalid $x_0 \rightarrow p$ simply via setting the dependency score to $-\infty$.

	Dev				Test	
	P	R	F_1	CM	F_1	CM
BIO	86.80	86.38	86.59	69.24	88.22	71.95
SPAN	87.68	86.75	87.21	68.43	88.44	70.22
CRF	87.89	87.62	87.76	71.59	88.76	73.01
FIRST	87.44	86.60	87.02	70.35	87.81	71.14
LAST	86.99	87.00	86.99	70.29	87.67	71.08
FLAT	85.63	82.26	83.91	63.41	83.32	62.22
CRF2O	**88.02**	**88.09**	**88.05**	**72.57**	**89.02**	**73.74**

Table 3: Finetuning results on CoNLL05 Dev and Test data under the setting of *w/ gold predicates*.

lize any word/predicate embeddings as well as LSTM layers for simplicity, which may potentially hinder the results. Despite this fact, our second-order CRF2O achieves $89.02_{\pm 0.17}$, $82.58_{\pm 0.47}$, and $87.66_{\pm 0.05}$, which outperforms the systems of Shi and Lin (2019) by 0.2, 0.6 and 1.2 F_1 scores and achieves new state-of-the-art on both CoNLL05 and CoNLL12 datasets. This implies that imposing stronger structure constraints can still bring remarkable improvements for span-style SRL even when empowered with very expressive encoders. In the bottom lines, we provide the results of utilizing RoBERTa, we can see that CRF and CRF2O augmented with RoBERTa can obtain further gains on top of BERT.

We highlight that we do not include any syntax-aware work (Xia et al., 2019; Zhou et al., 2020a) in Table 2, which has shown to deliver substantial gains for SRL (see further discussions in § C). It is still an open question to be investigated whether the benefits brought by our methods are orthogonal to linguistic syntax knowledge. We focus on pure syntax-agnostic models in this paper. So we do not list the results of this line of works in order to make fair comparisons.

4.2 Analysis

To better understand which empowers our proposed CRF and CRF2O and in what aspects they are helpful, we conduct detailed analyses on CoNLL05 Dev data. Considering that there exist many differences in model/training settings, we re-implemented the following two methods based on two widely used libraries HanLP[8] (He and Choi, 2021) and SuPar[9] for fair comparisons:

- **BIO**: BIO-based method of Zhou and Xu (2015). Following Zhang et al. (2021b), we employ linear-chain CRF (Lafferty et al., 2001) to conduct global inference during training.
- **SPAN**: span-based method of He et al. (2018a). We borrow the settings of Strubell et al. (2018) and make use of Biaffine layers for span scoring.

We adopt the same experimental setups for all implementations, i.e., finetuning on BERT and assuming all predicates are given. Results are shown in Table 3. It is clear that under the same settings, our CRF expands the advantages over BIO and SPAN, and CRF2O further improves the performance.

Impact of latent subtrees First of all, we consider three variants of our first-order CRF to verify the necessity of modeling arguments as latent trees: 1) FIRST, similar to CRF but always takes the first word as argument headword; 2) LAST, denoting the last word accordingly; 3) FLAT, similar to FIRST but directly attach other argument words to the first word. The first two variants fix the position of argument headwords. In Table 3, we observe that FIRST and LAST perform quite similarly and steadily inferior to CRF. This agrees with Zhang et al. (2021b), highlighting the importance of headwords in recognizing arguments. In contrast to CRF, FLAT completely excludes latent representations during training and restricts the height of the converted trees to 2. We can see that FLAT achieves 83.91 F_1 on Dev, a dramatic performance drop against CRF (87.76). Overall, as we expect, it seems that totally latent argument representations empower CRF a lot, performing best compared to other variants.

Structural consistency To quantify the benefits of our methods in making global decisions for SRL structures, we report the percentage of completely correct predicates (CM) (He et al., 2018a) in Table 3. We show that BIO with linear-chain CRF significantly outperforms SPAN, but still falls short of our CRF by 1.5. By explicitly modeling sibling information, CRF2O provides stronger structure constraints and goes further beyond CRF by 0.9. In terms of the performance broken down by argument length, as shown in Fig. 4a, SPAN lags largely behind BIO over length$\geqslant$8. We guess this is mainly because of their aggressive argument pruning strategy. And as expected, CRF and CRF2O demonstrate steady improvements over BIO and SPAN. We owe this to the superiority of our formulations in modeling subtree structures, thus providing more powerful argument representations and rich inter- and intra-argument dependency interactions.

[8] https://github.com/hankcs/HanLP
[9] https://github.com/yzhangcs/parser

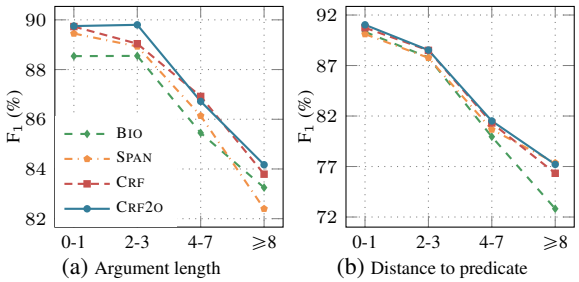


Figure 4: F_1 scores breakdown by argument length (Fig. 4a) and predicate-argument distance (Fig. 4b).

		Sents/s
Strubell et al. (2018)	BIO	50
He et al. (2018a)	SPAN	49
Li et al. (2019)	SPAN	20
Ours	CRF	**242**
	CRF2O	214
	CRF_{BERT}	136
	$\text{CRF2O}_{\text{BERT}}$	113

Table 4: Speed comparison on CoNLL05 Test data. We also list the speed of our TreeCRF models using BERT (CRF_{BERT} and $\text{CRF2O}_{\text{BERT}}$).

Long-range dependencies Fig. 4b shows the results broken down by predicate-argument distance. It is clear that the gaps between BIO and other methods become larger as the distance increases. This is reasonable since BIO lacks explicit connections for non-adjacent predicate-argument pairs, whereas ours provides direct word-to-word bilexical mappings. SPAN shows competitive results but is still inferior to ours. we speculate this is due to their inferiority in ultra-long arguments, as illustrated in Fig. 4a.

4.3 Efficiency

Table 4 compares different models in terms of parsing speed. We obtain the speed of previous works by rerunning their released code. For fair comparisons, all models are run on Intel Xeon E5-2650 v4 CPU and Nvidia GeForce GTX 1080 Ti GPU, and do not use any PLM layers.

We can clearly see that our CRF and CRF2O can parse about 242 and 214 sentences per second respectively, much faster than all previous works. In line with Strubell et al. (2018), our CRF and CRF2O consume 5,000 tokens (roughly 200 sentences) per mini-batch. However, Strubell et al. (2018) use up to 12 Transformer layers, much deeper than our 3-layer BiLSTM encoder. This explains their less efficiency from the side, as encoder layers might take up a major part of the running time, while the relative more efficient Viterbi decoding does not dominate the time-consuming. Moreover, our models are based on highly parallelized implementations (Zhang et al., 2020). We speculate that the model speed of Strubell et al. (2018) can be further improved with dedicated optimization. As for He et al. (2018a); Li et al. (2019), we adopt their default setting of 40 sentences per batch. They need to obtain the representations of all candidate argument spans, leading to high GPU memory usage. This limits us to enlarge the batch size further and significantly slows down the parsing speed. In the bottom lines of Table 4, we can see that our CRF and CRF2O enhanced with BERT achieve speeds of 136 and 113 sentences per second respectively. Overall, we can conclude that our proposed CRF and CRF2O are efficient enough and readily applicable to real-life systems.

5 Related Works

Span-style SRL Pioneered by Gildea and Jurafsky (2000), syntax has long been considered indispensable for span-style SRL (Punyakanok et al., 2008). With the advent of the neural network era, syntax-agnostic models make remarkable progress (Zhou and Xu, 2015; Tan et al., 2018; Cai et al., 2018), mainly owing to powerful model architectures like BiLSTM (Gal and Ghahramani, 2016) or Transformer (Vaswani et al., 2017). Meanwhile, other researchers also pay attention to the utilization of syntax trees, including serving as guidance for argument pruning (He et al., 2018b), as input features (Marcheggiani and Titov, 2017; Xia et al., 2019; Mohammadshahi and Henderson, 2021), or as supervisions for joint learning (Swayamdipta et al., 2018). However, to our best knowledge, very few works have been devoted to mining internal structures of shallow SRL representations. As exceptions, He et al. (2018a); Zhang et al. (2021b) take into account headwords while recognizing arguments. Beyond this, this work proposes to model full argument subtree structures rather than merely headwords and find more competitive results.

Parsing with latent variables Henderson et al. (2008, 2013) design a latent variable model to deliver syntactic and semantic interactions under the setting of joint learning. In more common situations where gold treebanks may lack, Naradowsky et al. (2012); Gormley et al. (2014) use LBP for the inference of semantic graphs and treat latent trees as global factors (Smith and Eisner, 2008)

to provide soft beliefs for reasonable predicate-arguments structures. This work differs in that we make hard constraints on syntax tree structures to conform to the SRL structures, and take only subtrees attached to predicates as latent variables. The intuition behind latent tree models (Meila and Jordan, 2000; Chu et al., 2017; Kim et al., 2017) is to utilize tree structures to provide rich structural interactions for problems with prohibitive high complexity. This idea is also common in many other NLP tasks like text summarization (Liu and Lapata, 2018), sequence labeling (Zhou et al., 2020d), and AMR parsing (Zhou et al., 2020c).

Reduction to tree parsing Researchers have investigated several ways to recover SRL structures from tree structures, due to their high coupling nature (Palmer et al., 2005). Early efforts of Cohn and Blunsom (2005) derive predicate-arguments from pruned phrase structures by using a CKY-style TreeCRF to learn parameters. Johansson and Nugues (2008a) and Choi and Palmer (2010) investigate retrieving semantic boundaries from dependency outputs. Their devised heuristics rely heavily on the quality of output trees, leading to inferior results. Our reduction is also inspired by works on other NLP tasks, including named entity recognition (NER) (Yu et al., 2020), nested NER (Fu et al., 2021; Lou et al., 2022), semantic parsing (Sun et al., 2017; Jiang et al., 2019), and EUD parsing (Anderson and Gómez-Rodríguez, 2021). As the most relevant work, Shi et al. (2020) also propose to reduce SRL to syntactic dependency parsing by integrating syntactic-semantic relations into a single dependency tree by means of joint labels. However, their approach shows non-improved results, possibly due to the label sparsity problem and high back-and-forth conversion loss. Also, they use gold treebank supervisions, while ours does not rely on any hand-annotated syntax data.

6 Discussions and Future Works

The basic idea of this work is to mimic SRL structures with a combination of multiple latent trees. This new perspective sheds light on some natural extensions of our work to other tightly related semantic parsing tasks, e.g., AMR (Zhang et al., 2019a) and UCCA (Jiang et al., 2019).[10] Tasks fall into this type exhibit very flexible graph representation schemes (e.g., *reentrancy* and *discontinuity*) (Zhang et al., 2019b), which are intractable by principled decoding algorithms like dynamic programming. We believe that employing structured inference in spirit of our approaches can provide considerable help in getting rid of greedy span/dependency selections and finding globally optimal structures.

We prefer to reduce SRL to dependency-based tree parsing rather than another paradigm, i.e., constituency parsing, partly because dependencies provide a more transparent bilexical governor-dependent encoding of predicate-argument relations (Hacioglu, 2004). We also do not pursue the way of jointly modeling dependencies and phrasal structures with lexicalized trees (Eisner and Satta, 1999; Yang and Tu, 2022; Lou et al., 2022) as our approach enjoys a lower time complexity of $O(n^3)$. Nonetheless, we admit potential advantages of this kind of modeling (Liu et al., 2022) and leave this as our future work.

There are other interesting perspectives deserve further explorations: given that span-style SRL substantially benefits from our formulation of recovering SRL structures from trees, *can the induced dependency trees learn plausible syntactic structures?* Or in other words, can they agree with linguistic-motivated annotations (Marcus et al., 1993)? We conduct thorough analyses in spirit of Gormley et al. (2014); Li et al. (2021) and give affirmative answers. Due to space limitations, we refer readers to § D and § E for details.

7 Conclusions

In this paper, we propose to reduce span-style SRL to dependency parsing by viewing flat phrasal arguments as latent subtrees, and design a novel span-constrained TreeCRF to accommodate the span structures. Taking inspirations from the parsing literature, we also build a second-order extension and find further gains. Our models are syntax-agnostic and do not rely on any external linguistic syntax knowledge. Experimental results show that, our proposed methods outperform all previous comparable works, achieving new state-of-the-art on both CoNLL05 and CoNLL12 benchmarks. Extensive analyses confirm that our approach enjoys some merits of global structural constraints, meanwhile maintaining acceptable time complexity. Furthermore, we find our modeling of latent subtrees provides effective assistance in terms of long-range dependencies and global consistency.

[10] We thank an anonymous reviewer for pointing out the connection.

8 Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments, and Prof. Zhenghua Li for very helpful feedbacks, suggestions and idea discussions. This work was supported by National Natural Science Foundation of China with Grant No. 62176173, 62076173, U1836222 and a project funded by the Priority Academic Program Development (PAPD) of Jiangsu Higher Education Institutions.

References

Mark Anderson and Carlos Gómez-Rodríguez. 2021. Splitting EUD graphs into trees: A quick and clatty approach. In *Proceedings of IWPT*, pages 167–174, Online.

Marzieh Bazrafshan and Daniel Gildea. 2013. Semantic roles for string to tree machine translation. In *Proceedings of ACL*, pages 419–423, Sofia, Bulgaria.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of EMNLP*, pages 1533–1544, Seattle, Washington, USA.

Rexhina Blloshmi, Simone Conia, Rocco Tripodi, and Roberto Navigli. 2021. Generating senses and roles: An end-to-end model for dependency- and span-based semantic role labeling. In *Proceedings of IJCAI*, pages 3786–3793.

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntactic-agnostic over syntactic-aware? In *Proceedings of COLING*, pages 2753–2765, Santa Fe, New Mexico, USA.

Jiong Cai, Yong Jiang, and Kewei Tu. 2017. CRF autoencoder for unsupervised dependency parsing. In *Proceedings of EMNLP*, pages 1638–1643, Copenhagen, Denmark.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL*, pages 152–164, Ann Arbor, Michigan.

Jinho Choi and Martha Palmer. 2010. Retrieving correct semantic boundaries in dependency structure. In *Proceedings of LAW*, pages 91–99, Uppsala, Sweden.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2010. Semantic role labeling for open information extraction. In *Proceedings of WS*, pages 52–60, Los Angeles, California.

Shanbo Chu, Yong Jiang, and Kewei Tu. 2017. Latent dependency forest models. In *Proceedings of AAAI*, pages 3733–3739.

Trevor Cohn and Philip Blunsom. 2005. Semantic role labelling with tree conditional random fields. In *Proceedings of CoNLL*, pages 169–172, Ann Arbor, Michigan.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *CL*, pages 589–637.

Simone Conia and Roberto Navigli. 2020. Bridging the gap in multilingual semantic role labeling: a language-agnostic approach. In *Proceedings of COLING*, pages 1396–1410, Barcelona, Spain (Online).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, Minneapolis, Minnesota.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of ICLR*, Toulon, France.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING*, pages 340–345.

Jason Eisner. 2000. *Bilexical Grammars and their Cubic-Time Parsing Algorithms*, pages 29–61. Dordrecht.

Jason Eisner. 2016. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Proceedings of WS*, pages 1–17, Austin, TX.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of ACL*, pages 457–464, College Park, Maryland, USA.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of EMNLP*, pages 960–970, Lisbon, Portugal.

Erick Fonseca and André F. T. Martins. 2020. Revisiting higher-order dependency parsers. In *Proceedings of ACL*, pages 8795–8800, Online.

Yao Fu, Chuanqi Tan, Mosha Chen, Songfang Huang, and Fei Huang. 2021. Nested named entity recognition with partially-observed treecrfs. In *Proceedings of AAAI*, pages 12839–12847, Online.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of ICML*, pages 1050–1059.

Daniel Gildea and Julia Hockenmaier. 2003. Identifying semantic roles using Combinatory Categorial Grammar. In *Proceedings of EMNLP*, pages 57–64.

Daniel Gildea and Daniel Jurafsky. 2000. Automatic labeling of semantic roles. In *Proceedings of ACL*, pages 512–520, Hong Kong.

Matthew R. Gormley, Margaret Mitchell, Benjamin Van Durme, and Mark Dredze. 2014. Low-resource semantic role labeling. In *Proceedings of ACL*, pages 1177–1187, Baltimore, Maryland.

Kadri Hacioglu. 2004. Semantic role labeling using dependency trees. In *Proceedings of COLING*, pages 1273–1276, Geneva, Switzerland.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL*, pages 1–18, Boulder, Colorado.

Wenjuan Han, Yong Jiang, and Kewei Tu. 2017. Dependency grammar induction with neural lexicalization and big training data. In *Proceedings of EMNLP*, pages 1683–1688, Copenhagen, Denmark.

Han He and Jinho D. Choi. 2021. The stem cell hypothesis: Dilemma behind multi-task learning with transformer encoders. In *Proceedings of EMNLP*, pages 5555–5577, Online and Punta Cana, Dominican Republic.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018a. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of ACL*, pages 364–369, Melbourne, Australia.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of ACL*, pages 473–483, Vancouver, Canada.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018b. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of ACL*, pages 2061–2071, Melbourne, Australia.

James Henderson, Paola Merlo, Gabriele Musillo, and Ivan Titov. 2008. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *Proceedings of CoNLL*, pages 178–182, Manchester, England.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *CL*, pages 949–998.

Wei Jiang, Zhenghua Li, Yu Zhang, and Min Zhang. 2019. HLT@SUDA at SemEval-2019 task 1: UCCA graph parsing as constituent tree parsing. In *Proceedings of SemEval*, pages 11–15, Minneapolis, Minnesota, USA.

Yong Jiang, Wenjuan Han, and Kewei Tu. 2016. Unsupervised neural dependency parsing. In *Proceedings of EMNLP*, pages 763–771, Austin, Texas.

Zhanming Jie and Wei Lu. 2019. Dependency-guided LSTM-CRF for named entity recognition. In *Proceedings of EMNLP-IJCNLP*, pages 3862–3872, Hong Kong, China.

Ishan Jindal, Ranit Aharonov, Siddhartha Brahma, Huaiyu Zhu, and Yunyao Li. 2020. Improved semantic role labeling using parameterized neighborhood memory adaptation.

Richard Johansson and Pierre Nugues. 2008a. Dependency-based semantic role labeling of PropBank. In *Proceedings of EMNLP*, pages 69–78, Honolulu, Hawaii.

Richard Johansson and Pierre Nugues. 2008b. Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *Proceedings of CoNLL*, pages 183–187, Manchester, England.

Richard Johansson and Pierre Nugues. 2008c. The effect of syntactic representation on semantic role labeling. In *Proceedings of COLING*, pages 393–400, Manchester, UK.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. In *Proceedings of ICLR*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*, San Diego, CA, USA.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL*, pages 478–485, Barcelona, Spain.

Lingpeng Kong, Alexander M. Rush, and Noah A. Smith. 2015. Transforming dependencies into phrase structures. In *Proceedings of NAACL-HLT*, pages 788–798, Denver, Colorado.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289, Williams College, Williamstown, MA, USA.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270, San Diego, California.

Phong Le and Willem Zuidema. 2015. Unsupervised dependency parsing: Let's use supervised parsers. In *Proceedings of NAACL-HLT*, pages 651–661, Denver, Colorado.

Tao Li, Parth Anand Jawale, Martha Palmer, and Vivek Srikumar. 2020. Structured tuning for semantic role labeling. In *Proceedings of ACL*, pages 8402–8412, Online.

Zhenghua Li, Min Zhang, Yue Zhang, Zhanyi Liu, Wenliang Chen, Hua Wu, and Haifeng Wang. 2016. Active learning for dependency parsing with partial annotation. In *Proceedings of ACL*, pages 344–354, Berlin, Germany.

Zuchao Li, Shexia He, Hai Zhao, Yiqing Zhang, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2019. Dependency or span, end-to-end uniform semantic role labeling. In *Proceedings of AAAI*, pages 6730–6737.

Zuchao Li, Hai Zhao, Shexia He, and Jiaxun Cai. 2021. Syntax role for neural semantic role labeling. *CL*, pages 529–574.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. Neural relation extraction with multi-lingual attention. In *Proceedings of ACL*, pages 34–43, Vancouver, Canada.

Ding Liu and Daniel Gildea. 2010. Semantic role features for machine translation. In *Proceedings of COLING*, pages 716–724, Beijing, China.

Tianyu Liu, Yuchen Jiang, Ryan Cotterell, and Mrinmaya Sachan. 2022. A structured span selector. In *Proceedings of NAACL*, pages 2629–2641, Seattle, United States.

Yang Liu and Mirella Lapata. 2018. Learning structured text representations. *TACL*, pages 63–75.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proceedings of ICLR*, New Orleans, LA, USA.

Chao Lou, Songlin Yang, and Kewei Tu. 2022. Nested named entity recognition as latent lexicalized constituency parsing. In *Proceedings of ACL*, pages 6183–6198, Dublin, Ireland.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of EMNLP*, pages 1506–1515, Copenhagen, Denmark.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *CL*, pages 313–330.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*, pages 91–98, Ann Arbor, Michigan.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81–88, Trento, Italy.

Marina Meila and Michael I. Jordan. 2000. Learning with mixtures of trees. *JMLR*, pages 1–48.

Alireza Mohammadshahi and James Henderson. 2021. Syntax-aware graph-to-graph transformer for semantic role labelling.

Jason Naradowsky, Sebastian Riedel, and David Smith. 2012. Improving NLP through marginalization of hidden syntactic structure. In *Proceedings of EMNLP*, pages 810–820, Jeju Island, Korea.

Joakim Nivre, Yoav Goldberg, and Ryan McDonald. 2014. Squibs: Constrained arc-eager dependency parsing. *CL*, pages 249–257.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2018. A span selection model for semantic role labeling. In *Proceedings of EMNLP*, pages 1630–1642, Brussels, Belgium.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *CL*, pages 71–106.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, RISHITA ANUBHAI, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages. In *Proceedings of ICLR*.

Fernando C. N. Pereira and David H. D. Warren. 1983. Parsing as deduction. In *Proceedings of ACL*, pages 137–144, Cambridge, Massachusetts, USA.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of CoNLL-WS*, pages 143–152, Sofia, Bulgaria.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proceedings of CoNLL-WS*, pages 1–40, Jeju Island, Korea.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *CL*, pages 257–287.

Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of COLING*, pages 1346–1352, Geneva, Switzerland.

Alexander Rush. 2020. Torch-struct: Deep structured prediction library. In *Proceedings of ACL*, pages 335–342, Online.

Yikang Shen, Yi Tay, Che Zheng, Dara Bahri, Donald Metzler, and Aaron Courville. 2021. StructFormer: Joint unsupervised induction of dependency and constituency structure from masked language modeling. In *Proceedings of ACL-IJCNLP*, pages 7196–7209, Online.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling.

Tianze Shi, Igor Malioutov, and Ozan Irsoy. 2020. Semantic role labeling as syntactic dependency parsing. In *Proceedings of EMNLP*, pages 7551–7571, Online.

David Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of EMNLP*, pages 145–156, Honolulu, Hawaii.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of EMNLP*, pages 5027–5038, Brussels, Belgium.

Weiwei Sun, Junjie Cao, and Xiaojun Wan. 2017. Semantic dependency parsing via book embedding. In *Proceedings of ACL*, pages 828–838, Vancouver, Canada.

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith. 2018. Syntactic scaffolds for semantic structures. In *Proceedings of EMNLP*, pages 3772–3782, Brussels, Belgium.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *TACL*, pages 29–41.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proceedings of AAAI*, pages 4929–4936.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *CL*, pages 161–191.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in NIPS*, pages 5998–6008, Long Beach, California, USA.

Xinyu Wang, Jingxian Huang, and Kewei Tu. 2019. Second-order semantic dependency parsing with end-to-end neural networks. In *Proceedings of ACL*, pages 4609–4618, Florence, Italy.

Qingrong Xia, Zhenghua Li, Min Zhang, Meishan Zhang, Guohong Fu, Rui Wang, and Luo Si. 2019. Syntax-aware neural semantic role labeling. In *Proceedings of AAAI*, pages 7305–7313.

Lu Xu, Zhanming Jie, Wei Lu, and Lidong Bing. 2021. Better feature integration for named entity recognition. In *Proceedings of NAACL-HLT*, pages 3457–3469, Online.

Songlin Yang, Yong Jiang, Wenjuan Han, and Kewei Tu. 2020. Second-order unsupervised neural dependency parsing. In *Proceedings of COLING*, pages 3911–3924, Barcelona, Spain (Online).

Songlin Yang and Kewei Tu. 2022. Combining (second-order) graph-based and headed-span-based projective dependency parsing. In *Findings of ACL*, pages 1428–1434, Dublin, Ireland.

Songlin Yang, Yanpeng Zhao, and Kewei Tu. 2021. Neural bi-lexicalized PCFG induction. In *Proceedings of ACL-IJCNLP*, pages 2688–2699, Online.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of ACL*, pages 201–206, Berlin, Germany.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of ACL*, pages 6470–6476, Online.

Liwen Zhang, Ge Wang, Wenjuan Han, and Kewei Tu. 2021a. Adapting unsupervised syntactic parsing methodology for discourse dependency parsing. In *Proceedings of ACL-IJCNLP*, pages 5782–5794, Online.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019a. AMR parsing as sequence-to-graph transduction. In *Proceedings of ACL*, pages 80–94, Florence, Italy.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019b. Broad-coverage semantic parsing as transduction. In *Proceedings of EMNLP-IJCNLP*, pages 3786–3798, Hong Kong, China.

Yu Zhang, Zhenghua Li, and Min Zhang. 2020. Efficient second-order TreeCRF for neural dependency parsing. In *Proceedings of ACL*, pages 3295–3305, Online.

Zhisong Zhang, Emma Strubell, and Eduard Hovy. 2021b. Comparing span extraction methods for semantic role labeling. In *Proceedings of SPNLP*, pages 67–77, Online.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of ACL-IJCNLP*, pages 1127–1137, Beijing, China.

Junru Zhou, Zuchao Li, and Hai Zhao. 2020a. Parsing all: Syntax and semantics, dependencies and spans. In *Findings of EMNLP*, pages 4438–4449, Online.

Junru Zhou, Zhuosheng Zhang, Hai Zhao, and Shuailiang Zhang. 2020b. LIMIT-BERT : Linguistics informed multi-task BERT. In *Findings of EMNLP*, pages 4450–4461, Online.

Qiji Zhou, Yue Zhang, Donghong Ji, and Hao Tang. 2020c. AMR parsing with latent structural information. In *Proceedings of ACL*, pages 4306–4319, Online.

Shilin Zhou, Qingrong Xia, Zhenghua Li, Yu Zhang, and Min Zhang. 2022. Fast and accurate span-based semantic role labeling as graph parsing. In *Proceedings of COLING*.

Yang Zhou, Yong Jiang, Zechuan Hu, and Kewei Tu. 2020d. Neural latent dependency model for sequence labeling.

Hao Zhu, Yonatan Bisk, and Graham Neubig. 2020. The return of lexical dependencies: Neural lexicalized PCFGs. *TACL*, pages 647–661.

A Implementation Details

In this work, we set up two alternative model architectures, i.e, LSTM-based and PLM-based. For the LSTM-based model, we directly adopt most settings of Dozat and Manning (2017) with some adaptions. The input vector of each token $x_i \in \boldsymbol{x}$ is the concatenation of three parts,

$$\mathbf{e}_i = \left[\mathbf{e}_i^{\text{word}}; \mathbf{e}_i^{\text{lemma}}; \mathbf{e}_i^{\text{char}}\right]$$

where $\mathbf{e}_i^{\text{word}}$ and $\mathbf{e}_i^{\text{lemma}}$ are word and lemma embeddings, and $\mathbf{e}_i^{\text{char}}$ is the outputs of a CharLSTM layer (Lample et al., 2016). We set the dimension of lemma and CharLSTM representations to 100 in our setting. We next feed the input embeddings into 3-layer BiLSTMs (Gal and Ghahramani, 2016) to get contextualized representations with dimension 800.

$$\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_n = \texttt{BiLSTMs}(\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_n)$$

Other dimension settings are kept the same as biaffine parser (Dozat and Manning, 2017). Following Zhang et al. (2020), we set the hidden size of Triaffine layer to 100 for CRF2O additionally. The training process continues at most 1,000 epochs and is early stopped if the performance on Dev data does not increase in 100 consecutive epochs. In practice, we observe that the training procedure is often stopped within 300 epochs (∼12 hours), which is efficient enough.

For PLM-based models, we opt to directly fine-tune the PLM layers without cascading word embedding and LSTM layers for the sake of simplicity. We use "`bert-large-cased`" for BERT, and "`roberta-large`" for RoBERTa respectively. We train the model for 20 epochs with roughly 1,000 tokens per batch and use AdamW (Kingma and Ba, 2015; Loshchilov and Hutter, 2019) with $\beta_1 = 0.9, \beta_2 = 0.9$ and $\lambda = 0$ for parameter optimization . The learning rate is 5×10^{-5} for PLMs, and 10^{-3} for the rest components. We adopt the warmup strategy in the first 10% of the training steps, and then apply a linear decay to the learning rate in the remaining steps.

Algorithm 1 The Second-order Inside Algorithm.

1: **Define:** $I, S, C \in \mathbf{R}^{n \times n \times b}$
2: $\triangleright$ b is batch size
3: **Initialize:** $C_{i,i} = \mathbf{0}, 0 \leqslant i \leqslant n$
4: **for** $w = 1$ **to** n **do** $\triangleright$ span width
5: *Parallelization on* $0 \leqslant i; j = i + w \leqslant n$
$I_{i,j} \leftarrow \log(\exp(C_{i,i} + C_{j,i+1})$
6: $\quad + \sum_{i<r<j} \exp(I_{i,r} + S_{r,j} + \mathrm{s}(i,r,j)))$
$\quad + \mathrm{s}(i,j)$
$I_{j,i} \leftarrow \log(\exp(C_{j,j} + C_{i,j-1})$
7: $\quad + \sum_{i<r<j} \exp(I_{j,r} + S_{r,i} + \mathrm{s}(i,r,j)))$
$\quad + \mathrm{s}(j,i)$
8: $S_{i,j} \leftarrow \log \sum_{i \leqslant r<j} \exp(C_{i,r} + C_{j,r+1})$
9: $C_{i,j} \leftarrow \log \sum_{i<r \leqslant j} \exp(I_{i,r} + C_{r,j})$
10: $C_{j,i} \leftarrow \log \sum_{i \leqslant r<j} \exp(I_{j,r} + C_{r,i})$
11: **end for**
12: **return** $C_{0,n}$

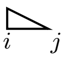
B The Inside Algorithm

We give the pseudocode of the common second-order Inside algorithm (McDonald and Pereira, 2006) in Alg. 1 as additional explanations to Fig. 3. The difference between the common second-order Inside algorithm and our proposed span-constrained one lies in the rule constraints green highlighted in Fig. 3.

In Line 3, $C_{i,i}$ corresponds to the axiom items  with initial score $\mathbf{0}$. Line 6 corresponds to two merge operations in Fig. 3. The incomplete span $I_{i,j}$ () is obtained by summing over either all pairs of complete span $C_{i,i}$ and $C_{j,i+1}$ (**R-LINK**) or pairs of the incomplete span $I_{i,r}$ and the sibling span $S_{j,r}$ (**R-LINK2**). In Line 8, the sibling span $S_{i,j}$ () is obtained by summing over all pairs of complete span $C_{i,r}$ and $C_{j,r+1}$ (**COMB**). Line 9 describes the similar merging operation on all pairs of the incomplete span $I_{i,r}$ and the complete span $C_{r,j}$, resulting a complete span

	WSJ			Brown		
	P	R	F_1	P	R	F_1
SA$^{\diamond}$	84.17	83.28	83.72	72.98	70.1	71.51
SA$^{\diamond}_{\text{ELMo}}$	86.21	85.98	86.09	77.1	75.61	76.35
G2G$^{\clubsuit}_{\text{BERT}}$	86.40	87.79	87.08	78.76	80.06	79.40
LIMIT$^{\clubsuit}_{\text{BERT}}$	86.62	89.12	87.85	79.58	83.05	81.28
ParsingAll$^{\clubsuit}_{\text{BERT}}$	86.77	88.49	87.62	79.06	81.67	80.34
ParsingAll$^{\clubsuit}_{\text{XLNet}}$	87.65	89.66	88.64	80.77	83.92	82.31
CRF$_{\text{BERT}}$	86.98	88.28	87.63	79.19	80.92	80.05
CRF2O$_{\text{BERT}}$	87.00	88.76	87.87	79.08	81.50	80.27
CRF$_{\text{RoBERTa}}$	87.20	88.67	87.93	79.29	81.48	80.38
CRF2O$_{\text{RoBERTa}}$	87.35	89.34	88.33	79.95	82.32	81.12
w/ gold predicates						
ParsingAll$^{\clubsuit}_{\text{BERT}}$	89.04	88.79	88.91	81.89	80.98	81.43
ParsingAll$^{\clubsuit}_{\text{XLNet}}$	89.89	89.74	89.81	85.35	84.57	84.96
TANL$^{\diamond}_{\text{T5}}$	-	-	89.3	-	-	82.0
CRF$_{\text{BERT}}$	88.93	88.58	88.76	82.87	81.67	82.27
CRF2O$_{\text{BERT}}$	89.00	89.03	89.02	82.81	82.35	82.58
CRF$_{\text{RoBERTa}}$	89.29	88.99	89.15	83.22	82.42	82.82
CRF2O$_{\text{RoBERTa}}$	89.45	89.63	89.54	83.89	83.39	83.64

Table 5: Comparisons with other less comparable works on CoNLL05 WSJ and Brown data. ♣ means using linguistic syntax knowledge; ◇ means different evaluation methods. SA: Strubell et al. (2018); ParsingAll: Zhou et al. (2020a); LIMIT: Zhou et al. (2020b); G2G: Mohammadshahi and Henderson (2021); TANL: Paolini et al. (2021).

$C_{i,j}$ () (**R-COMB**). Line 7 and Line 10 is the symmetric L-rules, which are omitted in Fig. 3.

C More Comparisons

In Table 5, for reference, we list the results of some works with different experimental settings and therefore less comparable. For example, Paolini et al. (2021) and Strubell et al. (2018)[11] adopt different evaluation metrics, resulting in slightly higher F_1 values than official tools. Nonetheless, we find that our CRF2O with RoBERTa achieves 89.54 F_1 on WSJ data under the *w/ gold predicates* setting, showing very competitive results when compared with T5-based model of Paolini et al. (2021). Zhou et al. (2020a) propose a joint-learning framework, integrating both (dependency/constituency) syntactic parse trees and dependency-based SRL resources to enhance their models. Their ablation studies show that using syntax trees brought an overall improvement of 1.6 F_1 score on CoNLL05 Dev data. We believe that we could achieve similar or even higher results than their syntax-aware XLNet-based models by incorporating human-annotated syntax knowledge. However, exploring different ways of injecting syntax is not the core of this paper. We take this as our future work.

[11] Under the *end-to-end* setting, different from the standard pratice (He et al., 2018a), Strubell et al. (2018) only ran the evaluation tool once, resulting in slightly higher precision values. See discussions in their code issue.

	P	R	F_1
CRF	75.28	75.24	75.26
CRF$_{\text{BERT}}$	84.70	84.39	84.54
w/ gold syntax			
Johansson and Nugues (2008c)	-	-	84.32
Li et al. (2019)$_{\text{ELMo}}$	-	-	89.20
CRF$_{\text{BERT}}$	**93.56**	**93.22**	**93.39**

Table 6: Results for dependency-based evaluation on CoNLL09 Test data under *w/o.* and *w/ gold syntax* settings.

D Dependency-based evaluation

Observing that our CRF model can conveniently determine dependencies from predicates to span headwords as by-products of constructing arguments, we therefore conduct dependency-based evaluation on CoNLL09 Test data (Hajič et al., 2009) to measure the quality of induced dependencies. As CoNLL09 data shares the same text content with CoNLL05, we directly make use of the model trained on CoNLL05 to obtain the results of CoNLL09 Test. Following Johansson and Nugues (2008b); Li et al. (2019), we also compare our CRF outputs with the upper bound of utilizing gold syntax tree to determine the headwords of predicted arguments. Since CoNLL05 contains only verbal predicates, we discard all nominal predicate-argument structures under the guidance of POS tags starting with `N*`. Word senses and self-loops are removed as well.

Results are listed in Table 6, from which we can draw some observations: 1) after using BERT, CRF outperforms LSTM-based model (75.26) by a large margin, implying BERT provides fruitful prior knowledge for dependency induction; 2) our CRF with BERT achieves 84.54 F_1 on CoNLL09 Test, exhibiting very promising performance even when compared to models using gold syntax (Johansson and Nugues, 2008b; Li et al., 2019). This indicates that the dependencies induced by CRF are highly in line with gold dependency-based annotations, illuminating potential extensions of our work on supervised dependency-based SRL.

Rules	Models	WSJ
Stanford	NL-PCFGs (Zhu et al., 2020)	40.5
	NBL-PCFGs (Yang et al., 2021)	39.1
	StructFormer (Shen et al., 2021)	46.2
	CRF	48.0
	CRF_{BERT}	**65.4**
	w/ gold POS tags (for reference)	
Collins	DMV (Klein and Manning, 2004)	39.4
	MaxEnc (Le and Zuidema, 2015)	65.8
	NDMV (Jiang et al., 2016)	57.6
	CRFAE (Cai et al., 2017)	55.7
	L-NDMV (Han et al., 2017)	59.5
	NDMV2o (Yang et al., 2020)	**67.5**

Table 7: Grammar induction results of our CRF model under different head-finding rules.

E Grammar Induction

To gain further insights, we make use of the scores defined in Eq. 5 to extract full dependency tree structures. Surprisingly, we find they are highly in agreement with expert-designed grammars (Marcus et al., 1993) when examined on the grammar induction task (Klein and Manning, 2004).

We show precise grammar induction results in Table 7. The results are not comparable to typical methods like DMV (Klein and Manning, 2004) or CRFAE (Cai et al., 2017), as they use gold POS tags as guidance, and we use Stanford Dependencies rather than Collins rules (Collins, 2003). Under similar settings, however, our learned task-specific trees perform significantly better than recent works.

Another interesting observation is that the gap between the BERT-based model and the LSTM-based model is much larger than that on SRL results. This implies LSTMs tend to be more fitted to SRL structures, while BERT is able to provide a strong inductive bias for syntax induction.