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

Anonymous ACL submission

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 explicitly considering internal argument structures, which may potentially hinder the model's expressiveness. To remedy this, we propose to reduce SRL to a dependency parsing task and regard the flat argument spans as latent subtrees. 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. Experiments on CoNLL05 and CoNLL12 benchmarks reveal that the results of our methods outperform all previous works and achieve the state-of-the-art.

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. The latter (He et al., 2018a; Ouchi et al., 2018), in contrast, directly models all predicate-argument pairs in a unified graph.

Despite ubiquity, there are some drawbacks that limit the expressiveness of the two methods. First,

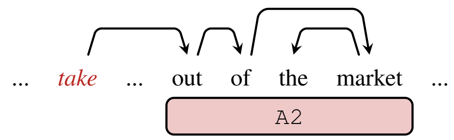


Figure 1: An argument example (below) and its related subtree structure (above) for the predicate "*take*".

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., 2020c; Xu et al., 2021). Second, for a sentence with n words, the span-based method needs to consider n^3 candidate predicate-argument pairs, thus severely suffering from data sparsity. To resolve this issue, the span-based method relies on heavy pruning to reduce the searching space, which potentially impairs the performance.

Meanwhile, both of the two formulations share some common flaws in terms of explicit modeling of internal argument structures. Internal structures appear to be beneficial to SRL, the reasons are two folds. First, the headword of a phrasal argument is very instructive to localize the semantic roles with respect to the selected predicate (Johansson and Nugues, 2008c; Punyakanok et al., 2008). In fact, many linguistic phenomena in SRL like *wh*-extraction (e.g., *who* did *what* to *whom*) can be transparently reflected by the dependency link from the predicate to the headword of its associated argument (Johansson and Nugues, 2008a; Surdeanu et al., 2008; Hajič et al., 2009). Taking Fig. 1 as an example, the semantic role "A2" of argument "out of the market" for the predicate "*take*" can be mirrored in the labeled dependency "*take* $\xrightarrow{\texttt{A2}}$ out". Second, constituent-based arguments are highly correlated with the subtrees governed by their headwords (Hacioglu, 2004; Choi and Palmer, 2010). For instance, the span boundary of the argument in Fig. 1 can be properly retrieved by the related dependency subtree. Thus considering subtree structures can intuitively provide very helpful clues for argument identification. However, to our best knowledge, very few works have made such explorations, stuck on the fact that span-style SRL has no determined internal argument structures.

Motivated by the above observations, in this work, we propose to model flat arguments as latent dependency subtrees. In this way, we can reduce SRL to a dependency parsing task seamlessly: *we view each SRL graph as partially-observed dependency trees where the exact subtree structure for each argument is not realized.* Specifically, we make use of TreeCRF (Eisner, 2000; Zhang et al., 2020) to learn the partially-observed trees and marginalize the latent structures out during training, as it provides a viable way to conduct probabilistic modeling of tree structures. While canonical TreeCRF aims to enumerate all possible trees, in our setting, we have to impose many span constraints on subtrees, aiming to correctly reflect the argument boundaries. To accommodate this, we further design a novel span-constrained TreeCRF to adapt it to our learning procedure, which explicitly forbids invalid edges across different arguments and the existence of 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 inference (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) 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; Zhang et al., 2020) without any obstacle. We note that our formulation does not need predicates or syntax trees to be pre-specified, and is thus fully *end-to-end*. Our contributions can be summarized as follows:[1]

- In aware of the importance of internal argument structures, we propose to reduce SRL to a dependency parsing task, and model internal argument structures as latent subtrees.
- We propose a novel span-constrained TreeCRF to learn the converted dependency trees, and further extend it to the second-order case.
- Experiments on CoNLL05 and CoNLL12 benchmarks reveal that the results of our proposed methods outperform others by a large margin, and achieve the state-of-the-art.

[1]Our code is available at `https://github.com`.

2 SRL as Dependency Parsing

Formally, given an input sentence $\boldsymbol{x}$ with n words $x_1, \ldots, x_n$, our goal is to predict a SRL graph $\boldsymbol{g}$. Each predicate-argument relation pair $(p, r_{i,j}) \in \boldsymbol{g}$ is composed of a predicate $p \in \boldsymbol{x}$ and a labeled argument $r_{i,j}$ that governs a consecutive word span $x_i, \ldots, x_j$ with the semantic role $r \in \mathcal{R}$, where $\mathcal{R}$ is the space of all roles. $\boldsymbol{g}$ can have more than one predicate. We denote the subgraph associated with predicate p as $\boldsymbol{g}_p$, where, as depicted in Fig. 2a, arguments belonging to p are non-overlapping.

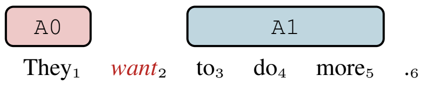
Optimal structured inference has long been deemed as intractable for the SRL task (Li et al., 2020). To sidestep this, our key idea is to decompose $\boldsymbol{g}$ into several subcomponents by predicates and find the optimal SRL structure $\boldsymbol{g}_p$ of each predicate independently. Our reduction of SRL to dependency parsing is achieved by casting the search of the optimal $\boldsymbol{g}_p$ as parsing a 1-best dependency tree for each predicate. To this end, we design back-and-forth procedures for SRL→Tree conversion and Tree→SRL recovery.

2.1 SRL→Tree Conversion

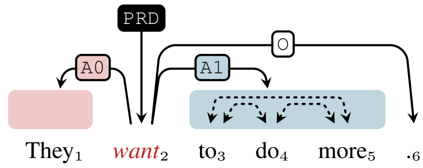
Given $\boldsymbol{x}$, we define a directed acyclic dependency tree $\boldsymbol{t}$ by assigning a head $h \in \{x_0, x_1, \ldots, x_n\}$ together with a relation label $r \in \mathcal{R}$ to each modifier $m \in \boldsymbol{x}$, where a dummy word x_0 is attached before $\boldsymbol{x}$ as the pseudo root node. The subtree governing $x_i, \ldots, x_j$ and rooted at h is denoted as $\boldsymbol{t}_{i,j}^h$.[2]

For predicate p, the goal is to convert its related SRL structure $\boldsymbol{g}_p$ to a "compatible" dependency tree $\boldsymbol{t}$. Intuitively, we say "compatible" if we can perfectly associate each predicate-argument pair $(p, r_{i,j}) \in \boldsymbol{g}_p$ with an arc $p \xrightarrow{r} h$ as well as its derived subtree $\boldsymbol{t}_{i,j}^h \subseteq \boldsymbol{t}$, which is also written as $p \xrightarrow{r} \boldsymbol{t}_{i,j}^h$ interchangeably. This implies two *span constraints* for the converted $\boldsymbol{t}$: 1) the scope of each subtree $\boldsymbol{t}_{i,j}^h$, populated by h and its descendants, must strictly correspond to an argument span; 2) for each argument span, p is allowed to point to only one headword h within the span, otherwise,

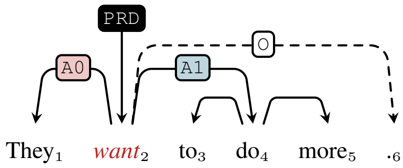
[2]In this work, we assume all dependency trees are *projective*, i.e., without any crossing arcs. This property allows us to associate the subtree with its continuous argument span.



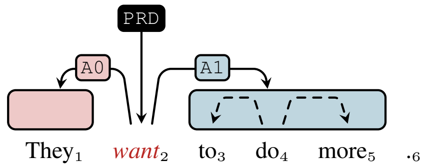
(a) Original structure: the predicate and its arguments are located in the upper half-plane of the sentence.



(b) Training: convert the predicate-argument structure to a dependency tree with (dotted) latent annotations.



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

it can be further split by separate subtrees due to the nature of tree projectivity (Ma and Zhao, 2015; Yang and Tu, 2021b).

One big challenge is that internal structures in SRL annotations are lacking, so that there are no determined subtree realizations for argument spans, and we can not simply establish a one-to-one mapping between $\boldsymbol{g}_p$ and a single $\boldsymbol{t}$ (Johansson and Nugues, 2008c; Choi and Palmer, 2010). Taking these considerations into account, we view each argument span as a blackbox, representing it as latent tree annotations during training. Specifically, we apply the following rules to construct a partially-observed tree for $\boldsymbol{g}_p$:

1. For predicate p, we first link the root node to p, forming a labeled dependency $x_0 \xrightarrow{\texttt{PRD}} p$.
2. For each argument $r_{i,j}$ of p, we view it as latent subtrees $T(p, r_{i,j})$. A completed subtree $\boldsymbol{t}^h_{i,j} \in T(p, r_{i,j})$ is restricted to be single-rooted at a headword h, and other words $\{x_i, \dots, x_j\}\backslash h$ are taken as descendants of h. The semantic role r is assigned as the label of the dependency pointing from p to headword h.
3. For a non-argument span (i, j), we label the dependency linking from p to words inside $\{x_i, \dots, x_j\}$ as "O" for distinction with semantic roles, yielding a pair $(p, \texttt{O}_{i,j})$. Then the construction process resembles that for argument spans, except that we do not limit the number of heads inside (i, j).

In this way, we can successfully convert $\boldsymbol{g}_p$ into a partially-observed tree $T(\boldsymbol{g}_p)$ with predicate p as the root. As illustrated by Fig. 2b, the predicate "*want*" links to only one word of each argument span, taking the semantic role as the dependency label; all dependencies crossing different argument spans are explicitly prohibited, and all word pairs inside an argument are deemed feasible to form a directed dependency. Please note that we do not assign any label to dependencies inside arguments as they have no realistic correspondences to SRL structures.

2.2 Tree→SRL Recovery

Supposing we have trained our dependency parser, during the evaluation phase, for predicate p, we produce a 1-best tree $\boldsymbol{t}^*$ with the constraint that x_0 points to the predicate word. Then, we divide $\boldsymbol{t}^*$ into multiple subtrees headed by p. For each subtree lying on the span (i, j), we decide whether it corresponds to an argument according to the label of dependency $p \xrightarrow{r} h$, where h is the span headword. If r is not "O", then h and its descendants form an entire argument span with semantic role r. In this case, we recursively make a top-down traversal starting from h, and collapse the subtree into a flat structure, forming a predicate-argument pair $(p, r_{i,j})$. Finally, we collate all arguments of predicate p into a single graph $\boldsymbol{g}^*_p$. The resulting SRL output becomes $\boldsymbol{g}^* = \{\boldsymbol{g}^*_p\}$. A recovery example is demonstrated in Fig. 2d.

3 Methodology

Now we elaborate the model architecture of our proposed dependency 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 calculate the probabilities of the converted partially-observed dependency 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.

$$\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_n = \text{Encoder}(x_0, x_1, \ldots, x_n) \quad (1)$$

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). We leave more setting details in § A.

(Second-order) Tree parameterization Dozat and Manning (2017) decompose $\boldsymbol{t}$ 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. We denote the head of the modifier m as $\boldsymbol{y}_m$. For each head-modifier pair $h \rightarrow m \in \boldsymbol{y}$, we score them using two MLPs followed by a Biaffine layer:

$$\begin{aligned}\mathbf{r}_i^{\text{head/mod}} &= \text{MLP}^{\text{head/mod}}(\mathbf{h}_i) \\ \text{s}(h \rightarrow m) &= \text{BiAF}\left(\mathbf{r}_h^{\text{head}}, \mathbf{r}_m^{\text{mod}}\right)\end{aligned} \quad (2)$$

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 further enhance the first-order biaffine parser. 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,

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

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

$$\text{s}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{h \rightarrow m \in \boldsymbol{y}} \text{s}(h \rightarrow m) \quad (4)$$

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

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

Then we parameterize the probability of skeletal tree $\boldsymbol{y}$ and its label sequence $\boldsymbol{r}$ as

$$\begin{aligned}P(\boldsymbol{y} \mid \boldsymbol{x}) &= \frac{\exp\left(\text{s}(\boldsymbol{x}, \boldsymbol{y})\right)}{Z(\boldsymbol{x}) \equiv \sum_{\boldsymbol{y}' \in Y(\boldsymbol{x})} \exp\left(\text{s}(\boldsymbol{x}, \boldsymbol{y}')\right)} \\ P(\boldsymbol{r} \mid \boldsymbol{x}, \boldsymbol{y}) &= \prod_{h \xrightarrow{r} m \in \boldsymbol{t}} P(r \mid \boldsymbol{x}, h \rightarrow m)\end{aligned} \quad (6)$$

where $Y(\boldsymbol{x})$ 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.

$$P(\boldsymbol{t} \mid \boldsymbol{x}) = P(\boldsymbol{y} \mid \boldsymbol{x}) \cdot P(\boldsymbol{r} \mid \boldsymbol{x}, \boldsymbol{y}) \quad (7)$$

3.2 Training and Inference

During training, we seek to maximize the probability $P(\boldsymbol{g} \mid \boldsymbol{x})$ of the SRL graph $\boldsymbol{g}$, which is then decomposed by predicates. Accordingly, we define the training objective as

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{g}) = -\sum_{p} \log P(\boldsymbol{g}_p \mid \boldsymbol{x}) \quad (8)$$

where $P(\boldsymbol{g}_p \mid \boldsymbol{x})$ is further expanded as the summation of probabilities of all compatible trees $T(\boldsymbol{g}_p)$ defined in § 2.1.

$$\begin{aligned}P(\boldsymbol{g}_p \mid \boldsymbol{x}) &= \sum_{\boldsymbol{t} \equiv (\boldsymbol{y}, \boldsymbol{r}) \in T(\boldsymbol{g}_p)} P(\boldsymbol{y} \mid \boldsymbol{x}) \cdot P(\boldsymbol{r} \mid \boldsymbol{x}, \boldsymbol{y}) \\ &= \frac{1}{Z(\boldsymbol{x})} \sum_{\boldsymbol{t}} \underbrace{\exp(\text{s}(\boldsymbol{x}, \boldsymbol{y})) \cdot P(\boldsymbol{r} \mid \boldsymbol{x}, \boldsymbol{y})}_{\exp(\text{s}(\boldsymbol{x}, \boldsymbol{t}))}\end{aligned} \quad (9)$$

The denominator $Z(\boldsymbol{x})$ can be efficiently computed via (second-order) Inside algorithm in $O(n^3)$ time complexity (Eisner, 2016; Zhang et al., 2020; Rush, 2020). As for the numerator, we make a slight change of the formula and define the labeled tree score as:

$$\text{s}(\boldsymbol{x}, \boldsymbol{t}) = \text{s}(\boldsymbol{x}, \boldsymbol{y}) + \log(P(\boldsymbol{r} \mid \boldsymbol{x}, \boldsymbol{y})) \quad (10)$$

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

In support of *end-to-end* learning, we conduct predicate identification by assigning a special label "O" to $x_0 \rightarrow p$ if p is a non-predicate word,

[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.

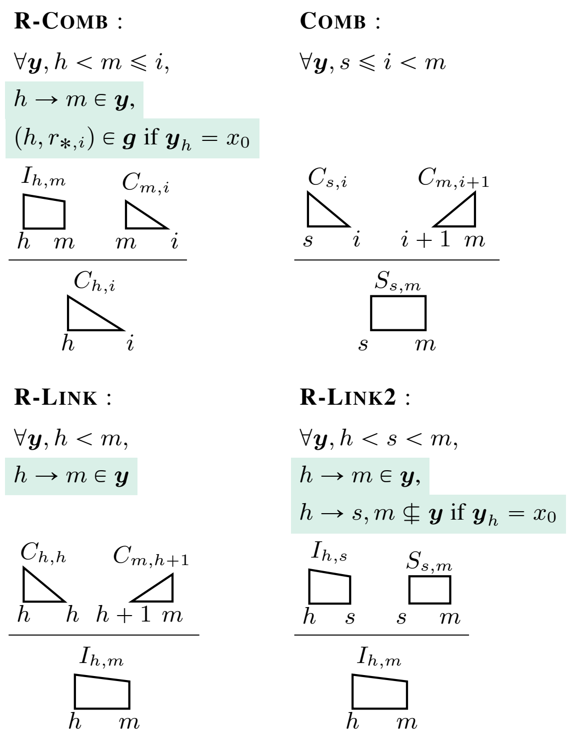


Figure 3: Deduction rules for our span-constrained Inside/Eisner algorithm (**R-COMB** and **R-LINK**) and its second-order extension (**COMB** and **R-LINK2**). Our modified rule constraints are highlighted in *green color*. We only show R-rules, and the symmetric L-rules are omitted for brevity.

and marginalize unobserved parts out during training. As a result, $P(\boldsymbol{g}_p \mid \boldsymbol{x})$ is exactly the label probability of $x_0 \rightarrow p$, i.e., $P(\text{O} \mid \boldsymbol{x}, x_0 \rightarrow p)$.

Span-constrained TreeCRF The calculation of the numerator of Eq. 9 differs from the traditional case of partial tree learning (Li et al., 2016) in that we impose span constraints mentioned in § 2.1 to the tree space $T(\boldsymbol{g}_p)$, where the common Inside algorithm is not adequate to. To resolve this, in this work, we propose a span-constrained TreeCRF to accommodate these constraints. We illustrate the deduction rules (Pereira and Warren, 1983) of our tailored TreeCRF and its second-order extension in Fig. 3. Basically, we avoid the arc $h \rightarrow m$ crossing different spans by prohibiting the merge operation of its related incomplete span $I_{h,m}$ (**R-LINK**). To prevent multiple headwords in the same argument, inspired by Zhang et al. (2021a), for predicate p, we only allow merging the complete span $C_{p,m}$ if m is the endpoint of an argument (**R-COMB**). For second-order case, we further prohibit the subtree $p \rightarrow s, m$ once s and m are located in the same argument (**R-LINK2**).

Inference During inference, as shown in Fig. 2c, for each candidate predicate p, we use (second-order) Eisner algorithm to parse an optimal projective tree rooted at p,

$$\boldsymbol{t}^* = \arg \max_{\boldsymbol{t} \equiv (\boldsymbol{y}, \boldsymbol{r}), \text{s.t.}, \boldsymbol{y}_p = x_0} \text{s}(\boldsymbol{x}, \boldsymbol{t}) \qquad (11)$$

where the score $\text{s}(\boldsymbol{x}, \boldsymbol{t})$ is calculated by Eq. 4. In practice, we first predict the label sequence $\boldsymbol{r}$ locally as it is independent of the skeletal tree, and then only realize the tree with the label of $x_0 \rightarrow p$ being PRD for efficiency. Dependencies with the label "O" are discarded.

4 Experiments

Following previous works, we measure our proposed first-order CRF and second-order CRF2O on two SRL benchmarks: CoNLL05 and CoNLL12. For CoNLL05, we adopt the standard data split of Carreras and Màrquez (2005). For CoNLL12, we extract data from OntoNotes (Pradhan et al., 2013), and follow the split of Pradhan et al. (2012). We adopt the official scripts provided by CoNLL05 shared task[4] for evaluation. More experimental setups are presented in § A.

4.1 Main results

Table 1 gives our main results on CoNLL05 in-domain WSJ data, out-of-domain Brown data, and CoNLL12 Test data. By default, our model works in an *end-to-end* fashion, i.e., predicting all predicates and associated arguments simultaneously. For comparisons with previous works, we report the results of using gold predicates as well. We achieve this by only parsing trees rooted at the pre-specified predicates.

We can clearly see that our CRF outperforms previous works by a large margin on all three datasets, especially showing larger gains on out-of-doamin Brown data. The second-order CRF2O further brings 0.4, 0.7, and 0.1 F_1 improvements over CRF on the three datasets, respectively. As shown in § 4.2, we attribute the improvements to better performing at global consistency and long-range dependencies.

The results under the setting of *w/ gold predicates* show similar trends. The bottom lines show the results of utilizing PLMs. Compared with the previous state-of-the-art BIO-based parser (Shi and Lin, 2019), our CRF model enhanced with BERT

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

		CoNLL05						CoNLL12		
		WSJ			Brown			Test		
		P	R	F_1	P	R	F_1	P	R	F_1
He et al. (2017)		80.2	82.3	81.2	67.6	69.6	68.5	78.6	75.1	76.8
He et al. (2018a)		81.2	83.9	82.5	69.7	71.9	70.8	79.4	80.1	79.8
Strubell et al. (2018)*		81.77	83.28	82.51	68.58	70.10	69.33	-	-	-
CRF		83.18	85.38	84.27	70.40	72.97	71.66	**79.47**	82.80	81.10
CRF2O		**83.26**	**86.20**	**84.71**	**70.70**	**74.16**	**72.39**	79.27	**83.24**	**81.21**
Li et al. (2019)	+ ELMo	85.2	87.5	86.3	74.7	78.1	76.4	84.9	81.4	83.1
CRF	+ BERT	86.53	88.02	87.27	78.77	80.71	79.73	84.08	85.79	84.92
CRF2O	+ BERT	86.78	88.59	87.67	79.29	82.10	80.67	84.19	86.21	85.19
CRF	+ RoBERTa	87.23	88.81	88.01	80.02	82.22	81.10	84.95	**87.00**	85.97
CRF2O	+ RoBERTa	**87.43**	**89.32**	**88.36**	**80.14**	**82.49**	**81.30**	**85.09**	86.99	**86.03**
w/ gold predicates										
He et al. (2017)		82.0	83.4	82.7	69.7	70.5	70.1	80.2	76.6	78.4
Ouchi et al. (2018)		84.7	82.3	83.5	76.0	70.4	73.1	84.4	81.7	83.0
Tan et al. (2018)		84.5	85.2	84.8	73.5	74.6	74.1	81.9	83.6	82.7
Strubell et al. (2018)*		84.70	84.24	84.47	73.89	72.39	73.13	-	-	-
Zhang et al. (2021b)		85.30	85.17	85.23	74.98	73.85	74.41	83.09	83.71	83.40
CRF		85.38	85.56	85.47	74.39	73.76	74.07	**83.21**	83.85	83.53
CRF2O		**85.47**	**86.40**	**85.93**	**74.92**	**75.00**	**74.96**	83.02	**84.31**	**83.66**
Shi and Lin (2019)	+ BERT	88.6	89.0	88.8	81.9	82.1	82.0	85.9	87.0	86.5
Zhang et al. (2021b)	+ BERT	87.70	88.15	87.93	81.52	81.36	81.44	86.00	86.84	86.42
CRF	+ BERT	88.70	88.32	88.51	82.58	81.47	82.02	87.20	87.38	87.29
CRF2O	+ BERT	88.80	88.88	88.84	82.95	82.82	82.89	87.35	87.78	87.57
CRF	+ RoBERTa	89.38	89.15	89.27	83.80	82.96	83.38	88.00	88.39	88.19
CRF2O	+ RoBERTa	**89.57**	**89.69**	**89.63**	**83.83**	**83.60**	**83.72**	**88.05**	**88.59**	**88.32**

Table 1: Results on CoNLL05 WSJ, Brown, and CoNLL12 Test data. All results are averaged over 4 runs with different random seeds. Strubell et al. (2018) incidentally report results with error (higher) precisions (see discussions in their code issue), and the above results marked by * are obtained by rerunning their released models.

shows competitive performance without using any pretrained word embeddings as well as LSTM layers. The results of CRF2O are 88.84, 82.89, and 87.57 respectively. By utilizing RoBERTa, our CRF2O obtains further gains and achieves new state-of-the-art on all three datasets under the *end-to-end* setting. As a reference, the recent best-performing syntax-related work (Zhou et al., 2020a) achieve 88.64 and 89.81 respectively on CoNLL05 Test data after using BERT and XLNet (Yang et al., 2019). Our models show very competitive results.

4.2 Analysis

To better understand in what aspects our CRF and CRF2O are helpful, we conduct detailed analyses on CoNLL05 Dev data. To this end, we re-implement the BIO-based model (BIO) of Strubell et al. (2018) and the span-based model (SPAN) of He et al. (2018a). For BIO, we follow Zhang et al. (2021b) by further employing linear-chain CRF (Lafferty et al., 2001) during training and perform Viterbi decoding to satisfy BIO constraints. For fair comparisons, we adopt the same experimental setups in that we report the results of finetuning on BERT and assume all predicates are given.

	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	**87.71**	87.49	87.60	70.99	88.51	72.40
CRF2O	**87.71**	**87.91**	**87.81**	**71.99**	**88.84**	**73.40**

Table 2: Finetuning results on CoNLL05 data under the setting of *w/ gold predicates*.

We first make comprehensive comparisons for the four methods in Table 2. We can observe that under the same settings, our CRF widens the advantages over BIO and SPAN, and CRF2O further improves the performance on Dev data.

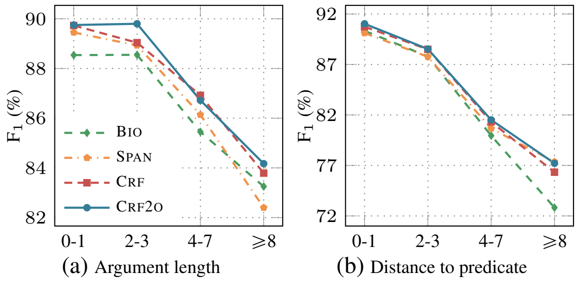


Figure 4: F_1 scores breakdown by argument length (Fig. 4a) and predicate-argument distance (Fig. 4b).

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 2. We show that the BIO model with linear-chain CRF significantly outperforms SPAN, but still falls short of our CRF by 1.75. By explicitly modeling sibling information, CRF2O goes further beyond CRF by 1 point. In terms of the performance broken down by argument length, as shown in Fig. 4a, SPAN lags largely behind BIO over length⩾8. We guess this is mainly because of their aggressive argument pruning strategy. Correspondingly, 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 rich inter- and intra-argument dependency interactions*.

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 mapping. SPAN shows competitive results but is still inferior to us. we speculate this is due to their inferiority in ultra-long arguments, as illustrated in Fig. 4a.

4.3 Dependency-based evaluation

Given above analyses, we have concluded that span-style SRL can substantially benefit from our formulation of recovering SRL from induced trees. This naturally opens up the question: *can induced trees learn plausible structures?*

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. We obtain the results by making predictions on CoNLL09 Test with the parser trained on CoNLL05 as they share the same text content.

	P	R	F_1
FIRST	61.16	63.51	62.31
LAST	60.61	62.95	61.76
CRF	72.20	74.98	73.56
CRF + BERT	82.30	84.04	83.16
CRF + BERT (*gold*)	**90.92**	**92.85**	**91.88**
w/ gold predicates			
CRF	75.28	75.24	75.26
CRF + BERT	84.70	84.39	84.54
CRF + BERT (*gold*)	**93.56**	**93.22**	**93.39**

Table 3: Results for dependency-based evaluation on CoNLL09 Test data under *w/o* and *w/ gold predicates settings*.

We set up two baselines: 1) FIRST, denoting always take the first word as argument head; 2) LAST, denoting the last word. 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 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. Table 3 lists the results.

We can clearly see that FIRST performs significantly better than LAST. This is reasonable since English is often regarded as right-branching, and headwords often precede their dependents. The results of CRF substantially outperform FIRST and LAST, especially after utilizing BERT. It also exhibits promising performance even when compared with the upper bound of utilizing gold syntax. This indicates that the induced dependencies of CRF are highly in line with dependency-based annotations (Johansson and Nugues, 2008a; Surdeanu et al., 2008). This sheds lights on extensions of our work on supervised dependency-based SRL, which we leave for future work.

To gain further insights, we also make use of scores defined in Eq. 4 to extract dependency trees. Surprisingly, we find they are highly in agreement with expert-designed grammars (Marcus et al., 1993) when examined on grammar induction task (Klein and Manning, 2004; Gormley et al., 2014).

Further discussions are available in § B.

5 Related Works

Span-style SRL Pioneered by Gildea and Jurafsky (2002), syntax has long been considered indispensable for the span-style SRL task (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; He et al., 2018a), 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), or as supervisions for joint learning (Swayamdipta et al., 2018). However, to our knowledge, very few works have been devoted to mining internal structures of shallow SRL representation. Zhang et al. (2021b) explore headword distributions by doing attention over words in argument spans. Beyond this, this work proposes to model full argument subtree structures by directly reducing SRL to a dependency parsing task, 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 graph, and take only subtrees attached to predicates as latent variables. The intuition behind latent tree models (Chu et al., 2017; Meila and Jordan, 2000; 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., 2020c), and AMR parsing (Zhou et al., 2020b).

Reduction to syntactic parsing Researchers have investigated several ways to recover SRL structures from syntactic trees, due to their high coupling nature (Palmer et al., 2005). Early efforts of Cohn and Blunsom (2005) derive predicate-arguments to pruned phrase structures equipped with 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), semantic dependency parsing (Sun et al., 2017), 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.

We prefer to reduce SRL to dependency parsing rather than another paradigm, i.e., constituency parsing, partly due to the fact that dependency trees can offer a more transparent bilexical governor-dependent encoding of predicate-argument relations (Hacioglu, 2004). We also do not take the way of jointly modeling dependencies and phrasal structures with lexicalized trees (Eisner and Satta, 1999; Yang and Tu, 2021a) as our approach enjoys a lower time complexity of $O(n^3)$. Nonetheless, we admit the potential advantages of this kind of modeling and leave this as our future work.

6 Conclusions

In this paper, we propose to reduce SRL to a dependency parsing task. Specifically, we view flat phrasal arguments as latent subtrees and design a novel span-constrained TreeCRF to accommodate the span structures. We also borrow traditional second-order parsing techniques and find further gains. Experimental results show that our proposed methods achieve state-of-the-art performance on both CoNLL05 and CoNLL12 benchmarks. Extensive analyses reveal the advantages of our modeling of latent subtrees in terms of long-range dependencies and global consistency. The evaluation over induced dependencies validates their agreement with linguistic motivated annotations.

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. Association for Computational Linguistics.

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

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. Association for Computational Linguistics.

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

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. Association for Computational Linguistics.

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

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. Association for Computational Linguistics.

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

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

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

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*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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

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

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

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. Association for Computational Linguistics.

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

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. Association for Computational Linguistics.

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

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

Daniel Gildea and Daniel Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, pages 245–288.

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. Association for Computational Linguistics.

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

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. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018a. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 364–369, Melbourne, Australia. Association for Computational Linguistics.

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. Association for Computational Linguistics.

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. Association for Computational Linguistics.

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. Coling 2008 Organizing Committee.

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.

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

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. Association for Computational Linguistics.

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

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

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

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

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.

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*, page 282–289, Williams College, Williamstown, MA, USA. Morgan Kaufmann.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL*, pages 2475–2485, San Diego, California. Association for Computational Linguistics.

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

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. Association for Computational Linguistics.

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. Association for Computational Linguistics.

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 the AAAI*, pages 6730–6737.

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

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

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.

Xuezhe Ma and Hai Zhao. 2015. Probabilistic models for high-order projective dependency parsing.

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. Association for Computational Linguistics.

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. Association for Computational Linguistics.

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

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

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. Association for Computational Linguistics.

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. Association for Computational Linguistics.

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

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

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. Association for Computational Linguistics.

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. Association for Computational Linguistics.

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

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

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. Association for Computational Linguistics.

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. Association for Computational Linguistics.

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

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. Association for Computational Linguistics.

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

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL*, pages 159–177, Manchester, England. Coling 2008 Organizing Committee.

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. Association for Computational Linguistics.

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.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*, pages 5998–6008, Long Beach, California, USA. Neural Information Processing Systems Foundation, Inc.

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. Association for Computational Linguistics.

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 Conference*, pages 7305–7313. AAAI Press.

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

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). International Committee on Computational Linguistics.

Songlin Yang and Kewei Tu. 2021a. Combining (second-order) graph-based and headed span-based projective dependency parsing.

Songlin Yang and Kewei Tu. 2021b. Headed span-based projective dependency parsing.

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

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in NIPS*, pages 5753–5763. Curran Associates, Inc.

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. Association for Computational Linguistics.

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

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. Association for Computational Linguistics.

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

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

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. Association for Computational Linguistics.

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

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

Yang Zhou, Yong Jiang, Zechuan Hu, and Kewei Tu. 2020c. 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

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

Table 4: Data statistics for CoNLL05 and CoNLL12 datasets.

Data Table 4 lists detailed data statistics of CoNLL05 and CoNLL12 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 use data splits of the CoNLL12 shared task (Pradhan et al., 2012), where the list of file IDs for Train/Dev/Test data can be found on the task webpage.[5] We adopt the same splits for both *end-to-end* and *w/ gold predicates* settings.

Model settings 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

[5] https://cemantix.org/conll/2012/download/ids/english/coref/

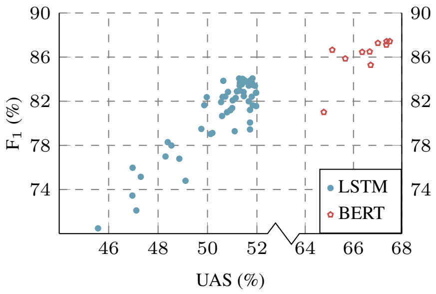


Figure 5: Relationships between grammar induction and SRL results on WSJ data, where the x-axis represents the UAS scores, and y-axis shows the corresponding SRL F_1 values.

(2017) with some adaptions. For each token $x_i \in \boldsymbol{x}$, its input vector 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.

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

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.

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 10 epochs with batch size of roughly 1000 tokens and use AdamW (Loshchilov and Hutter, 2019) 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 epoch followed by a linear decay in other epochs for the learning rate.

B Grammar Induction

Can we associate the SRL results with the quality of induced trees? Fig. 5 presents a scatter plot to show the correlations between the results of grammar induction and SRL. Specifically, we obtain UAS of induced trees by first picking up all SRL model checkpoints and then parsing 1-best trees using Eisner algorithm given scores defined in Eq. 4. We can see that the two results are highly positively correlated, showing a roughly linear relationship.

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

We show precise grammar induction results in Table 5. 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, and we use Stanford Dependencies instead of Collins rules (Collins, 2003). However, our learned task-specific trees perform significantly better than recent works under similar settings.

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.