Span-based Semantic Role Labeling as Lexicalized Constituency Tree Parsing

Anonymous ACL submission

Abstract

Semantic Role Labeling (SRL) is a critical task that focuses on identifying predicate-argument structures in sentences. Span-based SRL, a prominent paradigm, is often tackled using BIO-based or graph-based methods. However, these approaches often fail to capture the inherent relationship between syntax and semantics. While syntax-aware models have been proposed to address this limitation, they heavily rely on pre-existing syntactic resources, limiting their general applicability. In this work, we propose a lexicalized tree representation for span-based SRL, which integrates constituency and dependency parsing to explicitly model predicate-argument structures. By structurally representing predicates as roots and arguments as subtrees directly linked to the predicate, our approach bridges the gap between syntactic and semantic representations. Experiments on standard benchmarks (CoNLL05 and CoNLL12) demonstrate that our model achieves competitive performance, with particular improvement in predicate-given settings.

1 Introduction

Semantic Role Labeling (SRL) is a fundamental task in natural language processing, aiming to identify the arguments of predicates in a sentence and assign them appropriate semantic roles. There are two dominant paradigms in SRL: word-based (dependency-based) SRL, which assigns roles to individual words, and span-based SRL, which identifies multi-word argument spans. This paper focuses on span-based SRL.

Span-based SRL is typically approached through two primary methods: BIO-based tagging and (span-based) graph parsing. The BIO-based approach formulates SRL as a sequence labeling task (Zhou and Xu, 2015; Strubell et al., 2018), where each word in the sentence is assigned one of three labels $\{\mathrm{B}, \mathrm{I}, \mathrm{O}\}$, denoting the beginning, inside, or outside of an argument span. To ensure the coherence of argument spans, structural constraints, such as linear-chain Conditional Random Fields (CRFs), are applied. On the other hand, graph parsing directly identifies predicate-argument pairs by linking argument spans to predicates (He et al., 2018a; Li et al., 2019). While this method excels at capturing the full scope of predicate-argument structures, it faces significant challenges, including large search spaces ($O(n^3)$) and potential conflicts between argument spans due to the absence of structural constraints.

A long-standing hypothesis in SRL research is that syntactic information plays a critical role in accurate predicate-argument identification and classification (Gildea and Jurafsky, 2002). Syntactic structures provide key insights into predicate-argument structures, as evidenced in Propbank-style SRL (Bonial et al., 2010), where arguments are often syntactic constituents. Additionally, the process of establishing semantic relationships between predicates and arguments mirrors the construction of dependency relationships between words. These observations have led to the development of syntax-aware SRL models that leverage syntactic information, such as dependency and constituency trees, to improve SRL performance.

Syntax-aware models typically involve using syntactic trees to guide argument pruning (He et al., 2018b), integrating syntactic features derived from these trees (Xia et al., 2019), or employing multi-task learning frameworks to jointly learn syntax and semantics (Zhou et al., 2020). These approaches consistently demonstrate that incorporating syntactic knowledge leads to improved SRL performance. However, a significant limitation of most existing syntax-aware SRL models is their reliance on pre-existing syntactic resources, hindering their applicability to low-resource domains and languages.

Recent advances in structured representations for span-based SRL have sought to address this limitation by directly modeling argument structures with tree-based representations. Liu et al. (2022, 2023) introduced a framework where argument structures are represented as constituent trees, with each argument span corresponding to a tree node. Similarly, Zhang et al. (2022) explored the internal structures of argument spans, treating them as latent subtrees in a dependency representation. These approaches highlight the potential of structured representations to effectively capture the hierarchical and relational nature of predicate-argument structures.

In this work, we introduce a novel **lexicalized tree representation** for span-based SRL, seamlessly integrating the strengths of constituency and dependency parsing. A lexicalized tree captures the local domain of a headword—in this case, the predicate along with all its semantic arguments. This allows predicate-argument structures to be naturally modeled as (flat) lexicalized trees, where the predicate serves as the root, arguments are organized as constituent subtrees, and dependencies explicitly link arguments to the predicate. Our tree-based representation preserves the full complexity of predicate-argument structures, akin to graph-based models, while maintaining the coherence of argument spans, as seen in BIO-based models. Our main contributions are as follows:

- We treat span-based SRL as lexicalized tree parsing, combining the strengths of constituency and dependency parsing into a more integrated and interpretable framework.
- The tree-based modeling retains the global expressiveness needed to capture intricate predicate-argument relationships, while simultaneously enforcing span-level structural constraints.
- The proposed model achieves competitive performance on standard span-based SRL benchmarks (CoNLL05 and CoNLL12), outperforming existing methods in predicate-given settings.

Our code will be released on GitHub.

2 Background

2.1 Preliminaries and Notations

Formally, an input sentence is defined as $\boldsymbol{x} = x_0, x_1, \ldots, x_n$, where x_0 is a pseudo-root token and x_i is the i-th word in the sentence.

Semantic Role Labeling The span-based SRL aims to identify the spans of words that correspond to the arguments of predicates, and assign semantic roles representing the relationships between predicates and their argument spans. The predicate-argument structures are formally represented as: $\mathcal{A} = \{(i, j, p, r) \mid 1 \leq i \leq j \leq n, p \notin [i, j], r \in \mathcal{R}\}$, where (i, j, p, r) denotes an argument span of predicate p from the i-th word to the j-th word with semantic role r. For simplicity, we represent each predicate p separately: $\mathcal{A}_p = \{(i, j, r) \mid 1 \leq i \leq j \leq n, r \in \mathcal{R}\}$. Figure 1a shows an example sentence with a predicate and its corresponding arguments.

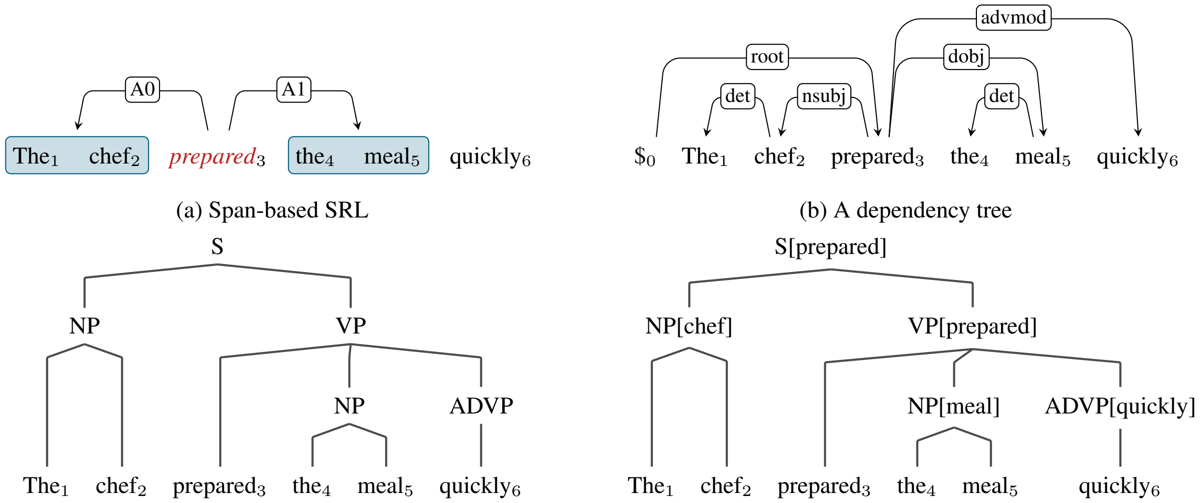
Dependency Parsing Dependency parsing (d-parsing) captures syntactic relationships between words in a sentence by constructing a dependency tree. A dependency tree is a directed graph where nodes represent words, and edges represent syntactic relations between them. A dependency tree is defined as: $\boldsymbol{d} = \{(h, m, r) \mid 0 \leq h \leq n, 1 \leq m \leq n, r \in \mathcal{R}\}$ where (h, m, r) denotes a dependency arc from head word h to dependent word m with dependency relation r. Figure 1b illustrates an example of a dependency tree.

Constituency Parsing Constituency parsing (c-parsing) produces a parse tree that captures the hierarchical phrase structure of a sentence. In this tree, internal nodes represent syntactic constituents, while leaf nodes correspond to individual words. A constituency tree is defined as: $\boldsymbol{c} = \{(i, j, \ell) \mid 1 \leq i \leq j \leq n, \ell \in \mathcal{L}\}$ where (i, j, ℓ) represents a constituent spanning from the i-th word to the j-th word with syntactic category ℓ. Figure 1c provides an example of a constituency tree.

2.2 The Connection between SRL and Syntax

SRL is closely tied to syntactic theory. For instance, predicates are typically verbs, where the Agent role often corresponds to the subject of the verb, and the Patient role aligns with the object. From a syntactic perspective, argument spans in SRL frequently map to constituents in a constituency tree, such as noun phrases or prepositional phrases. Additionally, identifying predicate-argument relations is conceptually similar to establishing head-dependent relations in a dependency tree.

Building on these insights, we propose a lexicalized tree representation for span-based SRL. This representation integrates two fundamental types of syntactic structures—constituency and dependency structures—into a unified framework. By seamlessly modeling predicate-argument structures, the lexicalized tree representation provides a more structured and interpretable perspective for SRL.



(a) Span-based SRL

(b) A dependency tree

(c) A constituency tree

(d) A lexicalized tree

Figure 1: Semantic and syntactic representations for the sentence "The chef prepared the meal quickly".

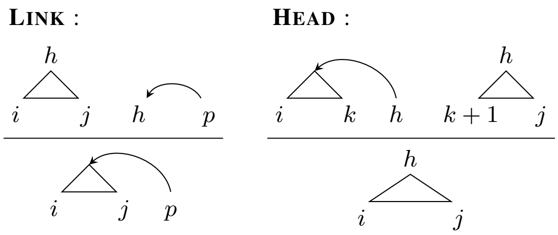


Figure 2: Deduction rules for lexicalized tree (LINK and HEAD). We show only left-rules, omitting the symmetric right-rules as well as initial conditions for brevity.

3 Proposed Method

This section presents our proposed method for span-based SRL using a lexicalized tree representation. We begin by introducing the foundational concepts of lexicalized trees, followed by detailing the construction of lexicalized trees from span-based SRL and their conversion into predicate-argument structures.

3.1 Lexicalized Tree Representation

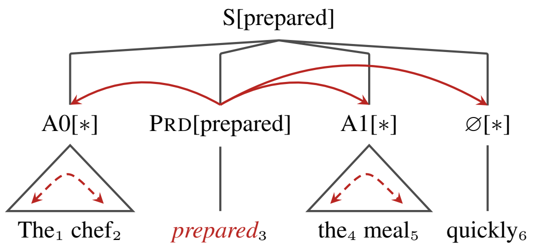
A lexicalized tree is a specialized constituency tree where each constituent is annotated with a headword, capturing the most important information within the constituent. For example, in Figure 1d, the headword of the "S" constituent is the word "prepared", which serves as the core of the sentence. Headwords propagate from the leaves to the root of the tree, and based on the dependency relations between these headwords, a dependency tree can be derived. Thus, a lexicalized tree y can be viewed as a combination of a constituency tree $\boldsymbol{c}$ and a dependency tree $\boldsymbol{c}$: $\boldsymbol{y} = \boldsymbol{c} \cup \boldsymbol{d}$.

While a lexicalized tree can be directly factorized into constituency spans and dependency arcs, its construction follows a specific process, as illustrated in Figure 2:

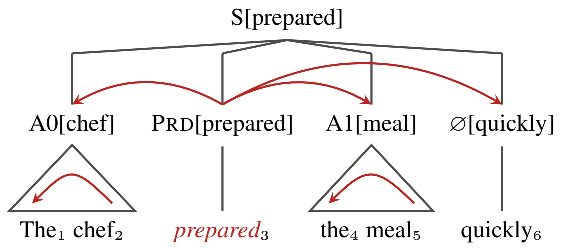
- A span (i, j) can be linked by an external anchor word $p \notin [i, j]$ via a dependency arc, forming a *linked span* (i, j, p). For simplicity, we omit the semantic role here.
- Each word is headed by itself, forming a single-word headed span (h, h, h).
- Linked spans can combine with sibling headed spans to form larger *headed spans* (i, j, h), where $h \in [i, j]$ is the anchor word of the linked span.
- By recursively combining these spans, a complete lexicalized tree is constructed.

These two types of spans—linked spans and headed spans—are closely related to predicate-argument structures. A predicate-argument pair can be viewed as a special case of linked spans, where the predicate acts as the external anchor word. Similarly, arguments can be represented by single words, as in word-based SRL, making argument spans a specific instance of headed spans. This connection makes it natural to use a lexicalized tree to represent predicate-argument structures. In such a tree, the predicate serves as the root, with each subtree corresponding to an argument span attached to it.

For the remainder of this section, we demonstrate how our method: (1) transforms span-based SRL into a lexicalized tree, and (2) reconstructs the resulting tree into predicate-argument structures.



(a) A latent lexicalized tree



(b) The optimal lexicalized tree

Figure 3: Illustration of our lexicalized tree representation. (a) A latent lexicalized tree derived from SRL, with the symbol $*$ indicating unrealized headwords. (b) The optimal lexicalized tree to be reconstructed into SRL.

3.2 Converting SRL to Lexicalized Tree

The lexicalized parsing method operates within a two-stage framework, where predicate-argument structure identification and semantic role classification are addressed separately and then combined to form the final result. To construct the lexicalized tree, we *individually* process each predicate. Given a sentence x and the argument structure $\mathcal{A}_p$ for a predicate p, the (unlabeled) lexicalized tree is constructed through the following steps:

1) **Span Partition**: The sentence is divided into four types of spans: a) Predicate span: The predicate itself. b) Argument span: Spans requiring semantic role labeling. c) Non-argument span: Spans between the predicate and argument spans, often representing complements or adjuncts. d) Sentence span: The entire sentence.
2) **Head Assignment**: Each span is assigned a head word: a) The head of the predicate span and the sentence span is the predicate itself. b) For argument spans and non-argument spans, the head is treated as a latent variable and inferred during training. Additionally, the internal dependency structure of these spans is also latent. Figure 3a provides an example.
3) **Span Linking**: Dependency relations are established between spans. Argument spans and non-argument spans are linked to the predicate span via dependency arcs. The predicate span is linked to the root token x_0.

Notably, the constructed lexicalized tree is *latent*, meaning its internal dependency structure is not explicitly annotated. A latent lexicalized tree corresponds to a forest of possible trees, each representing a realization of syntactic structure.

For semantic role classification, semantic roles are assigned to linked spans (i, j, p) rather than to constituent spans (i, j) or head spans (i, j, h). This is because semantic roles are inherently tied to predicates and are meaningless without their involvement. If the linked span is an argument, the corresponding argument role is assigned. If the linked span is a non-argument, the semantic role is set to $\varnothing$. The linked predicate span is assigned the special role PRD, which is used to identify the predicate in the next stage. If the gold-standard predicate is provided, this step can be skipped.

3.3 Recovering SRL from Lexicalized Tree

Assuming we have a trained parser that predicts the (unlabeled) lexicalized tree, and a labeler that assigns semantic roles to linked spans, the recovery of predicate-argument structures can be achieved through the following steps:

1) **Predicate Identification**: The predicate is identified as the (single-word) span directly linked to the root token with the label PRD. If the predicate is provided, this step is skipped.
2) **Optimal Tree Prediction**: For each predicate p, the optimal lexicalized tree (as shown in Figure 3b) is predicted using Equation 12, which ensures distinct results for different predicates.
3) **Argument Recovery**: All spans linked to the predicate span are extracted and assigned their optimal semantic roles. Non-argument spans (labeled as $\varnothing$) are discarded, while argument spans are retained with their semantic roles.

The final result contains all predicates and their associated argument spans, fully recovering the predicate-argument structures. Notably, our method operates in an *end-to-end* manner, meaning that predicate identification and argument recovery are seamlessly integrated within a single model, trained jointly and decoded step by step.

4 Model

In this section, we describe the detailed implementation of our lexicalized parsing model. Our approach consists of four key components: (1) scoring mechanism, (2) model architecture, (3) training, and (4) inference.

4.1 Scoring Factorization

As mentioned earlier, we adopt a two-stage framework for lexicalized parsing, following Dozat and Manning (2017) and Zhang et al. (2020). The tree skeletons and semantic roles are scored separately. For a unlabeled lexicalized tree $\boldsymbol{y}$, the basic (or first-order, 1O) score is the sum of the constituency and dependency scores:

$$\begin{aligned} s^{1\text{O}}(\boldsymbol{y}) &= s(\boldsymbol{c} \cup \boldsymbol{d}) \\ &= \sum_{(i,j) \in \boldsymbol{c}} s(i,j) + \sum_{(h,m) \in \boldsymbol{d}} s(h,m) \end{aligned} \tag{1}$$

where $s(i,j)$ is the score of a constituent span (i,j) and $s(h,m)$ is the score of a dependency arc (h,m).

To account for additional structural features related to predicate-argument structures, we extend the scoring factorization by including headed spans (i,j,h) and linked spans (i,j,p):

$$\begin{aligned} s(\boldsymbol{y}) = s^{1\text{O}}(\boldsymbol{y}) &+ \sum_{(i,j,h) \in \boldsymbol{y}} s(i,j,h) \\ &+ \sum_{(i,j,p) \in \boldsymbol{y}} s(i,j,p) \end{aligned} \tag{2}$$

4.2 Model Architecture

For an input sentence $\boldsymbol{x} = x_0, x_1, \ldots, x_n$, we first obtain contextual representations for each word x_i using the BERT model (Devlin et al., 2019).

$$\mathbf{r}_i = \text{BERT}(x_i) \tag{3}$$

These representations are then used to compute scores for constituent spans and dependency arcs using biaffine attention mechanisms (Dozat and Manning, 2017). For constituents, we use two separate MLPs to compute left and right boundary representations:

$$\begin{aligned} \mathbf{r}_i^{\text{left}} &= \text{MLP}^{\text{left}}(\mathbf{r}_i) \\ \mathbf{r}_j^{\text{right}} &= \text{MLP}^{\text{right}}(\mathbf{r}_j) \end{aligned} \tag{4}$$

Similarly, for dependencies, we compute head and dependent representations:

$$\begin{aligned} \mathbf{r}_h^{\text{head}} &= \text{MLP}^{\text{head}}(\mathbf{r}_h) \\ \mathbf{r}_m^{\text{mod}} &= \text{MLP}^{\text{mod}}(\mathbf{r}_m) \end{aligned} \tag{5}$$

Scores for constituency spans (i,j) and dependency arcs (h,m) are computed using two separate biaffine layers:

$$\begin{aligned} s(i,j) &= \text{BiAFF}^{\text{con}}(\mathbf{r}_i^{\text{left}}, \mathbf{r}_j^{\text{right}}) \\ s(h,m) &= \text{BiAFF}^{\text{dep}}(\mathbf{r}_h^{\text{head}}, \mathbf{r}_m^{\text{mod}}) \end{aligned} \tag{6}$$

For headed spans (i,j,h) and linked spans (i,j,p), we use the triaffine attention mechanism (Wang et al., 2019) to compute the scores:

$$\begin{aligned} \mathbf{r}_i^{\text{head/link}} &= \text{MLP}^{\text{head/link}}(\mathbf{r}_i) \\ s(i,j,h) &= \text{TriAFF}^{\text{head}}(\mathbf{r}_i^{\text{left}}, \mathbf{r}_j^{\text{right}}, \mathbf{r}_h^{\text{head}}) \\ s(i,j,p) &= \text{TriAFF}^{\text{link}}(\mathbf{r}_i^{\text{left}}, \mathbf{r}_j^{\text{right}}, \mathbf{r}_p^{\text{link}}) \end{aligned} \tag{7}$$

Additionally, semantic roles $s(i,j,p,r)$ are scored using a similar triaffine attention mechanism. Notably, semantic roles are scored after decoding the optimal unlabeled tree (with the exception of predicate identification). This technique reduces the computational cost to linear time by only scoring the predicted linked spans.

4.3 Training

We optimize the model using a structured learning objective, i.e., maximizing the likelihood of gold-standard trees. Given a lexicalized tree $\boldsymbol{y}$, its probability under CRF is defined as:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{\exp(\text{s}(\boldsymbol{y}))}{\boldsymbol{Z}(\boldsymbol{x}) \equiv \sum_{\boldsymbol{y}' \in \mathcal{Y}} \exp(\text{s}(\boldsymbol{y}'))} \tag{8}$$

where $\boldsymbol{Z}(\boldsymbol{x})$ is the partition function, summing over all possible trees $\mathcal{Y}$ for sentence $\boldsymbol{x}$.

However, this probability cannot be directly used to supervise the model because the headwords of argument spans and non-argument spans are latent variables (i.e., not yet realized). By enumerating all possible headwords, a forest $\mathcal{F}$ can be constructed, where each tree $\boldsymbol{y}^* \in \mathcal{F}$ is a realization of the latent headwords. We then optimize over the forest $\mathcal{F}$:

$$\begin{aligned} p(\mathcal{F}|\boldsymbol{x}) &= \frac{\sum_{\boldsymbol{y}^* \in \mathcal{F}} \exp(\text{s}(\boldsymbol{y}^*))}{\boldsymbol{Z}(\boldsymbol{x})} \\ L_{\text{skeleton}}(\theta) &= -\log p(\mathcal{F}|\boldsymbol{x}) \end{aligned} \tag{9}$$

The summation process is performed using a inside version of Eisner-Satta algorithm (Eisner and Satta, 1999), as described in Algorithm 1 in Appendix.

For semantic roles, we use cross-entropy loss:

$$\begin{aligned} p(r|i,j,p) &= \frac{\exp(s(i,j,p,r))}{\sum_{r' \in \mathcal{R}} \exp(s(i,j,p,r'))} \\ L_{\text{label}}(\theta) &= -\sum_{(i,j,p) \in \boldsymbol{y}} \log p(r^*|i,j,p) \end{aligned} \tag{10}$$

The final loss combines both objectives:

$$L(\theta) = L_{\text{skeleton}}(\theta) + L_{\text{label}}(\theta) \tag{11}$$

4.4 Inference

For a given sentence $\boldsymbol{x}$, we first identify all predicates within it. A word is considered a predicate if it can be linked to the root token x_0, determined by checking whether the role of the linked span satisfies $\hat{r}_{i,i,0} = \textsc{Prd}$. Then, for each predicate p, we use the Eisner-Satta algorithm to decode the optimal lexicalized tree:

$$\hat{\boldsymbol{y}} = \arg \max_{\boldsymbol{y}: x_0 \to p \in \boldsymbol{y}} s(\boldsymbol{y}) \tag{12}$$

By constraining the predicate to be attached to the root token, we ensure distinct optimal trees for different predicates, eliminating the need for repeated scoring across multiple predicates.

Finally, the optimal semantic role for each linked span (i, j, p) is predicted as:

$$\hat{r}_{i,j,p} = \arg\max_{r \in \mathcal{R}} s(i, j, p, r) \tag{13}$$

The predicate-argument structures are then recovered following the steps outlined in Section 3.3.

5 Experiments

5.1 Setup

Data The experiments are conducted on two widely used benchmarks for span-based SRL: CoNLL05 (Carreras and Màrquez, 2005) and CoNLL12 (Pradhan et al., 2012), with the standard splits. WSJ portion of the Penn Treebank (Marcus et al., 1993) is used for CoNLL05. To evaluate cross-domain generalization, we also test on the Brown corpus for CoNLL05.

Evaluation Metrics We evaluate model performance using Precision (P), Recall (R), and F1-score, following the official CoNLL05 evaluation script [1]. The results are averaged over three runs with different random seeds.

Hyperparameters Following previous work, we use the `bert-large-cased` version of BERT[2]. Additional implementation details, including hyperparameter settings, and optimization strategies are provided in Appendix A.1.

[1]https://www.cs.upc.edu/ srlconll

[2]https://huggingface.co/bert-large-cased

5.2 Main Results

Span-based SRL is evaluated under two settings: *end-to-end* and *predicate-given*. In the end-to-end setting, the model is required to predict both predicates and their arguments. In the predicate-given setting, predicates are provided as input, and the model is tasked with predicting the arguments. Table 1 presents the main results of our model compared to previous work.

End-to-End As shown in the upper part of Table 1, our method achieves competitive performance on CoNLL12, matching state-of-the-art models. However, on CoNLL05, our model slightly lags behind the best-performing model, with a 0.71 F1-score gap on the WSJ test set and a 0.8 F1-score gap on the Brown corpus. Notably, our model exhibits higher recall but lower precision, suggesting a tendency to identify more argument spans at the cost of some incorrect predictions.

Predicate-Given In the lower part of Table 1, we observe that our model maintains its strong ability to recall arguments. While our model performs slightly worse than Liu et al. (2023) on WSJ test set, it outperforms previous state-of-the-art models on both the Brown corpus and the CoNLL12 test set, achieving improvements of 0.22 and 0.25 F1-score, respectively. This demonstrates the robustness and generalization capability of our approach, particularly in cross-domain and large-scale settings.

5.3 Speed Efficiency

Table 2 compares the parsing speeds of different models on the CoNLL05 test set under the end-to-end setting. We adopt the speed benchmarks from Zhou et al. (2022) and Zhang et al. (2022), which were run on an Nvidia GTX 1080 Ti GPU. For He et al. (2018a) and Li et al. (2019), the batch size is set to approximately 2000 tokens, while other models use a batch size of 5000 tokens. Note that our model is run on a single Nvidia V100 GPU, which may result in a slight increase in speed.

Earlier models achieve parsing speeds below 50 sentences per second due to either heavy network architectures (Strubell et al., 2018) or large search spaces. Zhou et al. (2022) reduce the search space to $O(n^2)$ by using a word-based graph representation, leading to the fastest parsing speed. Zhang et al. (2022) model span-based SRL as dependency trees and use the Eisner algorithm (Eisner, 1996) for inference, which has a complexity of $O(n^3)$.

Model	CoNLL05							CoNLL12			
	Dev	WSJ			Brown			Dev	Test		
	F_1	P	R	F_1	P	R	F_1	F_1	P	R	F_1
					End-to-End						
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	–	–	–	–	–	–	–
Jia et al. (2022)	–	–	–	86.70	–	–	78.58	–	–	–	84.22
Zhou et al. (2022)	86.79	87.15	88.44	87.79	79.44	80.85	80.14	84.74	83.91	85.61	84.75
Zhang et al. (2022)	**87.03**	87.00	**88.76**	87.87	79.08	81.50	80.27	**85.53**	84.53	**86.41**	**85.45**
Liu et al. (2023)	–	**88.05**	88.61	**88.33**	**81.13**	81.58	**81.36**	–	**84.95**	85.85	85.40
Ours	87.00	86.59	88.68	87.62	78.68	**82.54**	80.56	84.99	84.00	86.34	85.15
					Predicate-Given						
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
Li et al. (2019)†	–	87.9	87.5	87.7	80.6	80.4	80.5	–	85.7	86.3	86.0
Shi and Lin (2019)	–	88.6	89.0	88.8	81.9	82.1	82.0	–	85.9	87.0	86.5
Jindal et al. (2020)	–	88.7	88.0	87.9	80.3	80.1	80.2	–	86.3	86.8	86.6
Conia and Navigli (2020)	–	–	–	–	–	–	–	–	86.9	87.7	87.3
Blloshmi et al. (2021)	–	–	–	–	–	–	–	–	87.8	86.8	87.3
Liu et al. (2022)	–	–	–	–	–	–	–	–	–	–	87.5
Jia et al. (2022)	–	–	–	88.25	–	–	81.90	–	–	–	87.18
Zhou et al. (2022)	87.54	89.03	88.53	88.78	83.22	81.81	82.51	86.97	87.26	87.05	87.15
Zhang et al. (2022)	**88.05**	89.00	89.03	89.02	82.81	82.35	82.58	87.52	87.52	87.79	87.66
Liu et al. (2023)	–	**89.77**	88.46	**89.11**	**83.96**	81.76	82.85	–	**88.10**	87.38	87.74
Ours	88.04	88.51	**89.07**	88.79	82.64	**83.51**	**83.07**	**87.71**	87.62	**88.36**	**87.99**

Table 1: Results on the CoNLL05 and CoNLL12 datasets. † indicates that no BERT-series models are used.

Model	Type	Sents/sec
Strubell et al. (2018)	BIO	50
He et al. (2018a)	GRAPH	49
Li et al. (2019)	GRAPH	20
Zhou et al. (2022)	GRAPH	**252**
Zhang et al. (2022)	TREE	113
Ours	TREE	116

Table 2: Speed comparison on CoNLL05-WSJ.

Model	CoNLL05		CoNLL12
	WSJ	Brown	Test
1o	87.49	79.87	84.89
Ours	**87.62**	**80.56**	**85.15**

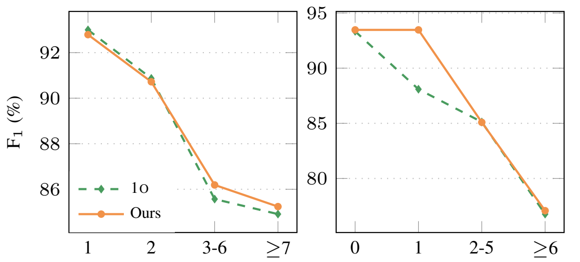
Table 3: Ablation study on the CoNLL05 and CoNLL12 datasets. Only the F_1 scores are reported.

Our approach employs the Eisner-Satta algorithm with a complexity of $O(n^4)$. However, after applying efficient batch processing on GPUs, both our algorithm and the Eisner algorithm operate at $O(n)$ complexity in practice, resulting in no significant difference in speed.

5.4 Analysis

Ablation Study A simplified version (named 1o) of our model can be obtained by removing the head span and linked span components, which are responsible for modeling predicate-argument structures. As these components are crucial to capture structural dependencies, we expect their removal to negatively impact performance. Table 3 shows the results of the ablation study under the end-to-end setting. We observe a performance drop across all datasets: a 0.13 F1-score reduction on WSJ, 0.69 F1-score on Brown, and 0.26 F1-score on CoNLL12. These results confirm the effectiveness of the full model.

Argument Width Figure 4a shows results broken down by argument width. Our model slightly underperforms 1o on shorter arguments. However, as the argument width increases, our model outperforms 1o more significantly.



(a) Argument width (b) predicate-argument distance

Figure 4: F_1 scores breakdown by argument width and predicate-argument distance on CoNLL05-WSJ.

Predicate-Argument Distance Figure 4b presents results broken down by predicate-argument distance, defined as the number of words between the predicate and the argument. It is clear that the gap between 1O and our model is marginal, except for the distance of 1, where our model outperforms 1O by a large margin.

6 Related Work

6.1 Span-based SRL

Span-based SRL is primarily approached through two paradigms: BIO-based and graph-based methods. The BIO-based approach first predicts predicates, then identifies their arguments via sequence labeling (Zhou and Xu, 2015; Strubell et al., 2018; Shi and Lin, 2019). This approach requires encoding the sentence multiple times for different predicates, as additional indicators (e.g., predicate embeddings) are used to locate them (Zhou and Xu, 2015). In contrast, graph-based approaches predict relations between predicates and argument spans directly (He et al., 2018a; Li et al., 2019). However, these models often suffer from two main challenges: (1) the large search space ($O(n^3)$ predicate-argument pairs) due to n^2 argument spans for each candidate predicate n, and (2) potential conflicts in argument identification, as each predicate-argument pair is predicted independently. To address these issues, He et al. (2018a, 2019); Jia et al. (2022) employ pruning strategies to reduce the search space. Additionally, Zhou et al. (2022) introduce a word-based graph representation with a BIO tagging scheme to reduce the search space to $O(n^2)$.

6.2 Syntax-aware SRL

The connection between syntax parsing and SRL has led to numerous studies on syntax-aware SRL. Since Gildea and Jurafsky (2002), syntax has been considered essential for span-based SRL. Researchers have explored various ways of integrating syntactic trees into SRL, such as guiding argument pruning (He et al., 2018b), using syntactic features as additional inputs, and jointly learning syntax and semantics through multi-task frameworks (Zhou et al., 2020). Recent advancements include syntax-enhanced self-attention mechanisms (Marcheggiani and Titov, 2017; Strubell et al., 2018), which consistently improve SRL performance. Despite these advancements, most syntax-aware SRL models rely on pre-existing syntactic resources rather than directly modeling argument structures.

6.3 SRL as Structured Parsing

Structured parsing has been successfully applied to various NLP tasks, including nested named entity recognition and sentiment analysis. For example, Fu et al. (2020) model nested entities as constituency trees, while Lou et al. (2022) extend this by introducing latent lexicalized tree structures. Similarly, Zhou et al. (2023) treat structured sentiment analysis as a dependency parsing problem.

In the context of SRL, structured parsing has also been explored. Liu et al. (2022, 2023) propose a constituency tree-based representation for SRL. Closest to our work, Zhang et al. (2022) cast span-based SRL as a dependency parsing task, representing argument spans as latent subtrees. Our approach integrates both constituency and dependency structures, offering a more comprehensive and structured perspective for SRL.

7 Conclusion

In this work, we introduce a novel lexicalized tree representation for span-based SRL, which integrates constituency and dependency parsing to model predicate-argument structures effectively. By structurally representing predicates as roots and arguments as subtrees explicitly linked to the predicate, our approach bridges the gap between syntactic and semantic representations. Experiments on standard benchmarks (CoNLL05 and CoNLL12) demonstrate that our model achieves competitive performance, particularly excelling in predicate-given settings. The ablation studies and analysis of predicate-argument structures further validate the effectiveness of our approach in capturing complex semantic relationships.

Limitations

Complexity The proposed lexicalized tree representation introduces additional complexity to the span-based SRL task. Constructing these trees requires integrating both constituency and dependency parsing. Training and inference rely on the Eisner-Satta algorithm to identify the optimal lexicalized tree structure, which has a space complexity of $O(n^3)$ and a time complexity of $O(n^4)$. While GPU acceleration can reduce the time complexity to $O(n)$, the model still demands significant memory resources, limiting its scalability to longer sentences.

References

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 the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 3786–3793. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Claire Bonial, Olga Babko-Malaya, Jinho D Choi, Jena Hwang, and Martha Palmer. 2010. Propbank annotation guidelines. *Center for Computational Language and Education Research, CU-Boulder*, 9:90.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, Michigan. Association for Computational Linguistics.

Simone Conia and Roberto Navigli. 2020. Bridging the gap in multilingual semantic role labeling: a language-agnostic approach. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1396–1410, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, 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 *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Yao Fu, Chuanqi Tan, Mosha Chen, Songfang Huang, and Fei Huang. 2020. Nested named entity recognition with partially-observed treecrfs. In *AAAI Conference on Artificial Intelligence*.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

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 the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada. Association for Computational Linguistics.

Shexia He, Zuchao Li, and Hai Zhao. 2019. Syntax-aware multilingual semantic role labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5350–5359, Hong Kong, China. 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 the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2061–2071, Melbourne, Australia. Association for Computational Linguistics.

Zixia Jia, Zhaohui Yan, Haoyi Wu, and Kewei Tu. 2022. Span-based semantic role labeling with argument pruning and second-order inference. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):10822–10830.

Ishan Jindal, Ranit Aharonov, Siddhartha Brahma, Huaiyu Zhu, and Yunyao Li. 2020. Improved semantic role labeling using parameterized neighborhood memory adaptation. *CoRR*, abs/2011.14459.

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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6730–6737.

Tianyu Liu, Yuchen Jiang, Ryan Cotterell, and Mrinmaya Sachan. 2022. A structured span selector. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2629–2641, Seattle, United States. Association for Computational Linguistics.

Wei Liu, Songlin Yang, and Kewei Tu. 2023. Structured mean-field variational inference for higher-order span-based semantic role labeling. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 918–931, Toronto, Canada. Association for Computational Linguistics.

Chao Lou, Songlin Yang, and Kewei Tu. 2022. Nested named entity recognition as latent lexicalized constituency parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6183–6198, Dublin, Ireland. Association for Computational Linguistics.

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

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

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2018. A span selection model for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1630–1642, Brussels, Belgium. 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 *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea. Association for Computational Linguistics.

Peng Shi and Jimmy Lin. 2019. Simple BERT models for relation extraction and semantic role labeling. *CoRR*, abs/1904.05255.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Xinyu Wang, Jingxian Huang, and Kewei Tu. 2019. Second-order semantic dependency parsing with end-to-end neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7305–7313.

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

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

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1127–1137, Beijing, China. Association for Computational Linguistics.

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

Shilin Zhou, Qingrong Xia, Zhenghua Li, Yu Zhang, Yu Hong, and Min Zhang. 2022. Fast and accurate end-to-end span-based semantic role labeling as word-based graph parsing. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4160–4171, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Yulin Zhou, Yiren Zhao, Ilia Shumailov, Robert Mullins, and Yarin Gal. 2023. Revisiting automated prompting: Are we actually doing better? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1822–1832, Toronto, Canada. Association for Computational Linguistics.

Algorithm 1 Eisner-Satta Algorithm

Input: scores of constituency spans $s(i, j)$ and dependency arcs $s(h \rightarrow m)$
Input: scores of linked spans $s(i, j, p)$ and headed spans $s(i, j, h)$
Define: $\mathrm{H}, \mathrm{L} \in \mathbb{R}^{n \times n \times (n+1)}$
Initialize: $\mathrm{H}_{:,:,:} = 0, \mathrm{L}_{:,:,:} = 0$
for $w = 1, \ldots, n$ **do**
    for $i = 1, \ldots, n - w$ **do**
        $j = i + w$
        for $h = i, \ldots, j$ **do**
            $\mathrm{H}_{i,j,h} = s(i, j, h) + s(i, j) + \max_{i \leq k \leq j}(\mathrm{H}_{i,k,h} + \mathrm{L}_{k+1,j,h}; \mathrm{L}_{i,k,h} + \mathrm{H}_{k+1,j,h})$
        end for
        for $p = 0, \ldots, n$ **do**
            $\mathrm{L}_{i,j,p} = s(i, j, p) + \max_{i \leq h \leq j}(\mathrm{H}_{i,j,h} + s(p \rightarrow h))$
        end for
    end for
end for
return $\mathrm{L}_{1,n,0}$

	Train	Dev	Test	Brown
CoNLL05	39,832	1,346	2,416	2,399
CoNLL12	75,187	9,603	9,479	–

Table 4: Data statistics for CoNLL05 and CoNLL12 datasets.

A Appendix

A.1 Implementation Details

The BERT (`bert-large-cased`) model is fine-tuned to obtain word representations. The dimensions of attention layers are set to 500 for tree skeletons and 100 for semantic roles. For training, the dropout rate is set to 0.1 for BERT and 0.33 for the other model components. The learning rate for BERT is set to 5×10^{-5}, while the learning rate for the other layers is set to 1×10^{-3}. We train the model for 10 epochs with a batch size of 1000 tokens using the AdamW optimizer. A linear warmup scheduler is used for the first 10% of the training steps. All experiments are run on NVIDIA V100 GPUs.