Fast and Accurate Span-based Semantic Role Labeling as Graph Parsing

Anonymous ACL submission

Abstract

Currently, BIO-based and Tuple-based approaches perform quite well on the span-based semantic role labeling (SRL) task. However, the BIO-based approach usually needs to encode a sentence once for each predicate when predicting its arguments, and the Tuple-based approach has to deal with a huge search space of $O(n^3)$, greatly reducing the training and inference efficiency. Moreover, both BIO-based and Tuple-based approaches usually consider only local structural information when making predictions. This paper proposes to cast end-to-end span-based SRL as a graph parsing task. Based on a novel graph representation schema, we present a fast and accurate SRL parser on the shoulder of recent works on high-order semantic dependency graph parsing (SDGP). Moreover, we propose a constrained Viterbi procedure to ensure the legality of the output graph. Experiments on CoNLL05, CoNLL12, and Chinese Proposition Bank 1.0 (CPB1.0) datasets show that our model achieves new state-of-the-art results and can parse over 600 sentences per second.

1 Introduction

As a fundamental natural language processing (NLP) task, semantic role labeling (SRL) aims to represent the semantic meaning of an input sentence as predicate-argument structures. SRL structure is shown to be helpful for many downstream NLP tasks, such as machine translation (Liu and Gildea, 2010; Marcheggiani et al., 2018) and question answering (Wang et al., 2015a).

There exist two forms of concrete SRL formalism in the community, i.e., dependency-based (or word-based) and span-based, depending on whether an argument consists of a single word or multiple words. This work focuses on the more complex span-based SRL task. Figure 1 shows an example sentence, consisting of two predicates. The argument corresponds to a span containing one or more words. Semantic roles of arguments are distinguished with edge labels, such as agent "A0" and patient "A1".

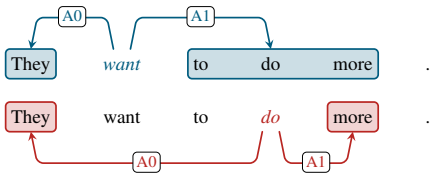


Figure 1: An example sentence for illustrating span-based SRL, consisting of two predicates, i.e., "want" and "do".

In recent years, the span-based SRL has achieved significant progress thanks to the impressive capability of deep neural networks in context representation. The BIO-based approach of Zhou and Xu (2015) and the Tuple-based approach of He et al. (2018) are two most representative neural network models. The BIO-based approach first predicts the predicates and then finds arguments for each predicate independently by labeling every word with BIO tags, like "B-A0" or "I-A0". The major weakness of the BIO-based approach is that a sentence usually has to be encoded and decoded for multiple times, each time for one predicate (Zhou and Xu, 2015; Shi and Lin, 2019), thus proportionally reducing the training and inference efficiency.

The Tuple-based approach (He et al., 2018; Li et al., 2019) directly considers word spans as arguments (Tuples) and links whole arguments to predicates. However, the Tuple-based approach also suffers from a severe inefficiency problem, since the search space is as high as $O(n^3)$, which is composed of $O(n)$ potential predicates and $O(n^2)$ possible arguments. Previous works usually have to resort to pruning techniques to improve efficiency, however with very limited effect and making the model more complex as well.

Another common disadvantage of both the BIO-based and Tuple-based approaches is that they

make use of quite local structural information when making decisions. For instance, arguments and labels are separately predicted for each predicate without inter-predicate interaction.

Inspired by the resemblance between SRL and semantic dependency graph parsing (SDGP, Oepen et al. 2014), and motivated by the recent progress in SDGP models, we cast end-to-end span-based SRL as a SDGP task. In order to decompose arguments into graph nodes, we propose a novel graph representation schema to transform original span-based SRL structure into a word-level graph. Based on the schema, we build a fast and accurate end-to-end model upon recently proposed high-order graph parsing model (Wang et al., 2019), which introduces three second-order sub-trees via mean field variational inference (MFVI). This makes our model consider inter-predicate interactions beyond local edges. In addition, since the vanilla graph parsing model cannot guarantee the legality of the output graph in the sense of corresponding SRL structure, we propose a simple post-processing method based on constrained Viterbi to make sure that the output graph can be recovered back to a proper SRL structure. In summary, we make the following contributions.

- We for the first time cast span-based SRL as a SDGP task. Based on a new graph representation schema, we present a fast and accurate end-to-end span-based SRL parser on the shoulder of recent successful SDGP models.
- We propose a simple constrained Viterbi procedure for post-processing illegal graphs.
- Experiments on CoNLL05, CoNLL12, and CPB1.0 show that our approach achieves new state-of-the-art performance under both settings of w/o and w/ pre-trained language models (PLMs). Detailed analysis reveals clear and interesting insights. Moreover, our parser can naturally support the dependency-based SRL and also achieves SOTA performance on the CoNLL09 dataset[1].
- Our parser is more than one magnitude faster than previous parsers and can analyze over 600 sentences per second.

We will release our code, configuration files, and major models at github.

[1]The results are shown in § E

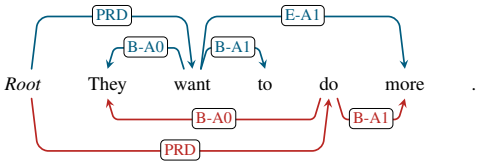


Figure 2: The graph representation corresponding to the original predicate-argument structure of Figure 1.

2 Proposed Approach

This work proposes to cast end-to-end span-based SRL as a word-level semantic dependency graph parsing task. The key challenge is to design a suitable graphical representation to encode span-based semantic role annotations for all predicates in a sentence.

2.1 Graph Representation

SRL-to-Graph Transformation. We propose to transform the original span-based SRL structure into a word-level graph, as depicted in Figure 2. First, we add a pseudo "*Root*" node at the beginning of the sentence and link all the predicates to it with "PRD" as the edge label. Please note that a predicate always corresponds to a single word in SRL datasets (He et al., 2018). Then, we attach each semantic argument, denoted as $a = w_i, ..., w_j, (i \leq j)$, to its corresponding predicate (denoted as w_k). Specifically, we add two edges, one from w_k to w_i and the other from w_k to w_j, with "B-r" and "E-r" as their labels. If an argument contains one word, i.e., $i = j$, we only add the "B-r" edge. $r \in \mathcal{R}$ is the original semantic role label and $\mathcal{R}$ is the set of role labels. We denote the new composite label as ℓ, and the new label set as $\mathcal{L}$. Except the "BE" schema, we also tried another "BII" schema where every word in argument are linked to the predicate with labels "B-r I-r I-r ...". However, our preliminary experiments show that the performance of "BII" is much inferior to "BE", so we finally choose the "BE" schema as our representation schema.

Graph-to-SRL Recovery. After generating the word-level graph through our model, we need to recover it to the corresponding SRL structure. Given a graph that is legal in the sense of SRL structure, we can obtain the corresponding SRL representation straightforwardly. Specifically, all children nodes (words) of the pseudo "*Root*" are treated as predicates. Then, for each predicate, we recover all its argument based on the edge labels. An argu-

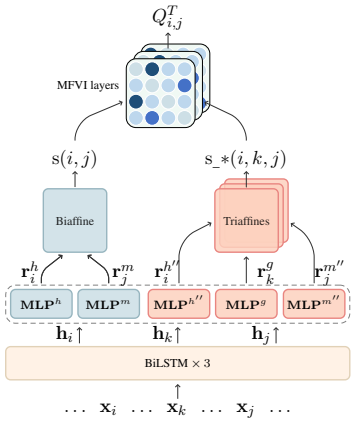


Figure 3: Illustration of our model. $\mathrm{s\_}*(i,k,j)$ corresponds to the second-order scores, where $* \in \{\mathrm{sib}, \mathrm{cop}, \mathrm{grd}\}$.

ment corresponds to either a paired labels, such as "B-A0" and "E-A0", or a single beginning label, such as "B-A0". Unfortunately, vanilla graph parsing cannot guarantee the legality of output graphs. To this end, in Section 2.5, we propose a simple yet effective constrained Viterbi decoding procedure to handle the issues.

2.2 First-order Model (O1)

Based on our designed graph representation, we can address span-based SRL as a graph parsing problem. In this work, we adopt the framework of Dozat and Manning (2018) which consists of two stages: 1) predicting all edges and 2) assigning labels for each edge.

Encoder. In this work, we use a three-layer BiLSTM to get the contextualized hidden representation $\mathbf{h}_i$ for each input token w_i. A more detailed description can be found in § A.

$$\mathbf{h}_i = \mathrm{BiLSTM}(w_i) \tag{1}$$

Edge scoring and classification. We treat edge prediction as a binary 0/1 classification task, where 1 means that there is an edge between the given word pair and 0 otherwise.

Following Dozat and Manning (2018), we use two MLPs to get representation vectors of a word as a head or a modifier respectively, and then use BiAffine and Sigmoid to compute edge scores and probabilities.

$$\begin{aligned}
\mathbf{r}^{\mathrm{h}}_i; \mathbf{r}^{\mathrm{m}}_i &= \mathrm{MLP}^{\mathrm{h}}(\mathbf{h}_i); \mathrm{MLP}^{\mathrm{m}}(\mathbf{h}_i) \\
\mathrm{s}(i,j) &= \begin{bmatrix} \mathbf{r}^{\mathrm{m}}_j \\ 1 \end{bmatrix}^{\top} \mathbf{W} \mathbf{r}^{\mathrm{h}}_i \\
\mathrm{p}(i,j) &= \sigma(\mathrm{s}(i,j)) = \frac{\exp(s(i,j))}{\exp(s(i,j)) + 1}
\end{aligned} \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{(d+1)\times d}$; $\mathrm{s}(i,j)$ represents the edge score of $i \rightarrow j$[2], and $\mathrm{p}(i,j)$ is the probability of the existence of the edge after Sigmoid function σ. During inference, only edges that have $\mathrm{p}(i,j) > 0.5$ are retained.

Label scoring and classification. The skeleton of the graph is decided after the edge classification step. Similar to edge scoring, we use two extra MLPs and a set of Biaffines to compute the label scores.

$$\begin{aligned}
\mathbf{r}^{\mathrm{h}'}_i; \mathbf{r}^{\mathrm{m}'}_i &= \mathrm{MLP}^{\mathrm{h}'}(\mathbf{h}_i); \mathrm{MLP}^{\mathrm{m}'}(\mathbf{h}_i) \\
\mathrm{s}(i,j,\ell) &= \begin{bmatrix} \mathbf{r}^{\mathrm{m}'}_j \\ 1 \end{bmatrix}^{\top} \mathbf{W}^{\mathtt{label}}_{\ell} \begin{bmatrix} \mathbf{r}^{\mathrm{h}'}_i \\ 1 \end{bmatrix} \\
\mathrm{p}(\ell|i,j) &= \frac{\exp(\mathrm{s}(i,j,\ell))}{\sum_{\ell' \in \mathcal{L}} \exp(\mathrm{s}(i,j,\ell'))}
\end{aligned} \tag{3}$$

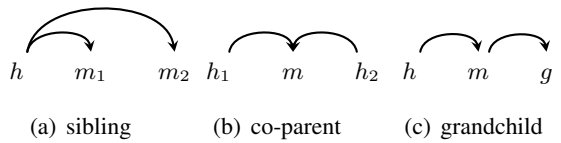
where $s(i,j,\ell)$ is the score of the label ℓ for the edge (i,j); $\mathrm{p}(\ell|i,j)$ is the probability after softmax over all labels. Each label has its own Biaffine parameters $\mathbf{W}^{\mathtt{label}}_{\ell} \in \mathbb{R}^{(d+1)\times(d+1)}$.

2.3 Second-order Model (O2)

The difference between our second-order model and first-order model lies in the edge classification module. An obvious limitation of the first-order model is its strong assumption that edges are mutually independent and thus it only considers the information between the current two words when scoring the edge. One natural extension is to exploit scores of sub-trees consisting of multiple edges when determining the unlabeled graph. We consider three types of sub-trees, as shown in Figure 4. Here, second-order means that scoring sub-trees containing two edges.

Figure 4(a) shows a **sibling sub-tree** where two words depend on the same head word. This corresponds to three cases: 1) two words are both predicates, and depend on "*Root*"; 2) two words are the beginning and ending words of an argument

[2]For convenience, we abbreviate the edge $i \rightarrow j$ as (i,j) in the remaining part of the paper.



(a) sibling (b) co-parent (c) grandchild

Figure 4: Three types of second-order sub-trees.

of some predicates; and 3) two words belong to two arguments of the same predicate.

Figure 4(b) shows a **co-parent sub-tree** where two words govern the same word. This corresponds to two cases: 1) w_{h_1} and w_{h_2} are two predicates; 2) one of w_{h_1} and w_{h_2} is "*Root*", and the other is a predicate.

Figure 4(c) shows a **grandchild sub-tree** in which three words form a head-modifier-grandchild chain. This also covers two cases: 1) w_h is "*Root*", w_m is a predicate, and w_g is the beginning or ending word of an argument which belongs to w_m; 2) w_h is a predicate, w_m is not only the beginning or ending word in an argument but also another predicate, and w_g is the beginning or ending word in an argument which belongs to predicate w_m.

We can see that the three types of sub-trees capture a rich set of edge interaction cases, allowing the model to evaluate graphs from a more global view.

Second-order scoring. First, we use three new MLPs to get representations of each word for playing different roles in second-order sub-trees, respectively.

$$\mathbf{r}_i^{\mathrm{h}''};\mathbf{r}_i^{\mathrm{m}''};\mathbf{r}_i^{\mathrm{g}} = \mathrm{MLP}^{\mathrm{h}''/\mathrm{m}''/\mathrm{g}}(\mathbf{h}_i) \quad (4)$$

where $\mathbf{r}_i^{h''};\mathbf{r}_i^{m''};\mathbf{r}_i^{g}$ denote the representation vectors of w_i as head, modifier, and grandchild respectively. Then, a TriAffine scorer (Zhang et al., 2020) taking the three vectors as input is applied to compute the score of the corresponding second-order structure,

$$\mathrm{TriAFF}(\mathbf{v}_1,\mathbf{v}_2,\mathbf{v}_3) = \begin{bmatrix}\mathbf{v}_3\\1\end{bmatrix}^{\top}\mathbf{v}_1{}^{\top}\mathbf{W}'\begin{bmatrix}\mathbf{v}_2\\1\end{bmatrix} \quad (5)$$

where $\mathbf{W}' \in \mathbb{R}^{(d'+1)\times d'\times(d'+1)}$ and $\mathbf{v}_i \in \mathbb{R}^{d'}, i \in \{1,2,3\}$. Finally, scores of the three types of sub-trees can be computed as follows respectively.

$$\mathrm{s\_sib}(i,j,k) = \mathrm{TriAFF1}(\mathbf{r}_i^{\mathrm{h}''},\mathbf{r}_j^{\mathrm{m}''},\mathbf{r}_k^{\mathrm{m}''}) \quad (6)$$

$$\mathrm{s\_cop}(i,j,k) = \mathrm{TriAFF2}(\mathbf{r}_i^{\mathrm{h}''},\mathbf{r}_j^{\mathrm{m}''},\mathbf{r}_k^{\mathrm{h}''}) \quad (7)$$

$$\mathrm{s\_grd}(i,j,k) = \mathrm{TriAFF3}(\mathbf{r}_i^{\mathrm{h}''},\mathbf{r}_j^{\mathrm{m}''},\mathbf{r}_k^{\mathrm{g}}) \quad (8)$$

It should be noted that for symmetrical sibling sub-trees and co-parent sub-trees, we compute their corresponding scores only once, i.e., $\mathrm{s\_sib}(i,j,k) = \mathrm{s\_sib}(i,k,j)$ and $\mathrm{s\_cop}(i,j,k) = \mathrm{s\_cop}(k,j,i)$.

Approximate inference using MFVI. Given scores of edges and second-order sub-trees, the most straightforward choice is directly searching for the optimal graph with the highest accumulated score, which however is NP-hard, because there is no efficient algorithm to compute the score of the graph for all shapes. Therefore, we follow Wang et al. (2019) and employ approximate inference (MFVI) for both training and evaluation.

Concretely, we first define a confidence variable Q_{ij} for each edge (i,j) to estimate the probability of the edge being in the correct semantic graph. MFVI approximates the true probability iteratively as follows.

$$\begin{aligned}\mathcal{M}_{ij}^{(t-1)} &= \sum_{k\neq i,j} Q_{ik}^{(t-1)}\mathrm{s\_sib}(i,j,k)\\ &\quad + Q_{kj}^{(t-1)}\mathrm{s\_cop}(i,j,k)\\ &\quad + Q_{jk}^{(t-1)}\mathrm{s\_grd}(i,j,k)\\ Q_{ij}^{(t)} &= \sigma(\mathrm{s}(i,j) + \mathcal{M}_{ij}^{(t-1)})\end{aligned} \quad (9)$$

where $t \in [1,T]$ is the iteration number. $\mathcal{M}_{ij}$ is an intermediate variable that stores information from second-order sub-tree scores. $Q_{ij}^{(0)}$ is initialized with the $\mathrm{p}(i,j)$ in equation 2. We define the score of edge not being in the graph as 0 and normalize $Q_{ij}^{(t)}$ via Sigmoid operation σ at each iteration. Following Wang et al. (2019), we stop computation after $T = 3$ iterations. During inference, Q_{ij}^{T} is directly used as $\mathrm{p}(i,j)$.

The intuitive explanation is that the probability of edge's existence is affected by both local information, i.e., the first-order score and non-local information, i.e., the higher-order score. And through $T = 3$ times of iteration, MFVI collects rich historical decision information which is helpful for the model to make more accurate final decision.

2.4 Training

The loss of our system comes from both edge and label classification modules. Given one sentence X and its gold graph G, the fully connected graph

of X is denoted as C.

$$L_e(\theta) = -\sum_{(i,j)\in G} \log \mathrm{p}'(i,j) - \sum_{(i,j)\in C\backslash G} \log\left(1-\mathrm{p}'(i,j)\right)$$

$$L_l(\theta) = -\sum_{(i,j)\in G} \log \mathrm{p}(\hat{\ell}|i,j)$$

where θ denotes model's parameters; $C\backslash G$ is the set of incorrect edges; $\hat{\ell}$ is the gold label of edge (i,j). In our first-order model, $\mathrm{p}'(i,j)$ equals the probability of the edge's existence $\mathrm{p}(i,j)$ computed in equation 2. In the second-order model, it equals to the final posterior distribution, i.e., Q_{ij}^T. The final loss of our system is the weighted sum of the two losses:

$$L(\theta) = \lambda L_l(\theta) + (1-\lambda) L_e(\theta)$$

where $0 < \lambda < 1$ is set to 0.06.

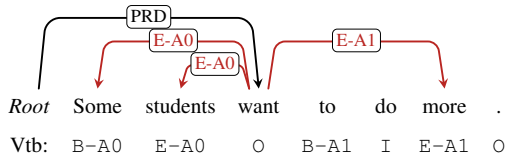
2.5 Inference

During inference, we first use the edge classification module to build the graph skeleton, and then use the label classification module to assign labels to predicted edges. If the generated graph is legal, we can directly recover the corresponding SRL structure through Graph-to-SRL procedure described in 2.1.

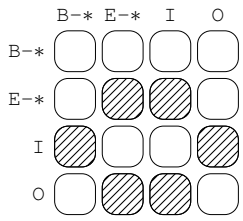
However, since the label classification module handles each edge independently, the resulting graph may contain conflicts, as shown in the upper part of Figure 5(a). First, if two consecutive edges are both labeled as "E−∗", then it is impossible to recover the corresponding arguments. Another conflicting scene is when there exists a single outlier edge labeled as "E−∗".

Conflict resolution via constrained Viterbi. We propose to employ constrained decoding to handle conflicts shown in Figure 5(a). Concretely, when conflicts occur during recovering arguments for a predicate in the output graph, we re-label all words in the sentence for the predicate.[3] The output labels are shown in the second row starting with "Vtb", where the two new labels "O/I" mean outside/inside an argument respectively. The idea of constrained Viterbi is to control the transition matrix to make sure that the resulting label sequence is always correct. For example, as shown in Figure 5(b), we only allow transitions from "B−∗" and "I" to "E−∗", and disallow transitions from "E−∗" and "O" to "E−∗".

[3]We have also tried to directly perform constrained Viterbi on the edges, instead of all words in the sentence. However, the performance is much inferior.



(a) A conflicting example. Edges in red cause conflicts, and the label sequence below is the corrected sequence via our constrained Viterbi.



(b) Transition matrix.

Figure 5: A conflicting example and our transition matrix. B−∗ and E−∗ represent all the composite beginning and ending labels. Cells with fence denote the prohibited transitions.

In fact, constrained Viterbi is a widely used technique in BIO-based SRL models. However, it is not trivial to apply constrained Viterbi to our SDGP framework as a post-processing step. The main challenge is how to make use of the probabilities computed by our SDGP model. We propose to combine the probabilities of the edge classification and label classification modules as follows:

$$\mathrm{p}''(\ell|i,j) = \mathrm{p}(i,j)\cdot\mathrm{p}(\ell|i,j)$$
$$\mathrm{p}''(\mathtt{O}|i,j) = \mathrm{p}''(\mathtt{I}|i,j) = 1-\mathrm{p}(i,j)$$

where $\mathrm{p}''(\ell|i,j)$ is the probability for the normal label such as "B−A0". $\mathrm{p}''(\mathtt{O}|i,j)$ and $\mathrm{p}''(\mathtt{I}|i,j)$ share the same value because they both mean that the word is neither the beginning nor the ending word of an argument, but "I" has an extra indication that there is an unpaired "B−∗" in the left side.

3 Experiments

Data. We conduct experiments on CoNLL05 (Palmer et al., 2005) and larger-scale CoNLL12 (Pradhan et al., 2012), which are two widely used English SRL datasets. For Chinese, we use Chinese Proposition Bank 1.0 (CPB1.0) (Xue, 2008) as our dataset. Following previous works on span-based SRL, we omit predicate sense prediction (Zhou and Xu, 2015; He et al., 2017).

Model	Type	Sents/sec
He et al. (2018)	Tuple-based	44
Strubell et al. (2018)	BIO-based	45
Li et al. (2019)	Tuple-based	19
Our O1		**726**
Our O2		611
Our O1 +BERT		252
Our O2 +BERT		228

Table 1: Speed comparison on the CoNLL05-dev.

Evaluation metrics. We mainly focus on end-to-end setting, and jointly predict both predicates, arguments, and the corresponding roles. We use the official evaluation scripts[4]. We choose seeds randomly to run our model for 3 times and report the average results. For significance test, we follow Xia et al. (2019b) and use their released scripts of Dan Bikel's randomized parsing evaluation comparator. We adopt most of the hyper-parameters settings used in Wang et al. (2019). The difference is detailed in § B. We denote our first-order model and second-order model as O1 and O2. Please kindly notice that this work is a pure modeling study. So we do not compare with syntax-aware works (Roth and Lapata, 2016; Xia et al., 2019b; Zhou et al., 2020).

3.1 Efficiency Comparison

Table 1 compares different models in terms of decoding speed. For fair comparison, we re-run all previous models on the same GPU environment (Nvidia GeForce 1080 Ti 11G).

We can see that our models improve the efficiency of previous span-based SRL models by at least one order of magnitude. Compared with the Tuple-based approach (He et al., 2018; Li et al., 2019), our graph-based parser only has a $O(n^2)$ search space. As for the BIO-based model of Strubell et al. (2018), the encoder contains 12 self-attention layers, and they adopts a pipeline framework by first predicting all predicates via sequence labeling and then recognizing arguments, leading to its low parsing speed.

Our second-order model is only 15% slower than the first-order model, showing that the computing of second-order sub-tree scores and the MFVI inference procedure are both very fast via large tensor computation on GPUs. And when augmented with BERT, our methods can still parse over 200 sentences per second.

[4] http://www.cs.upc.edu/~srlconll/st05/st05.html

3.2 Main Performance Results

Table 2 shows performance comparison on both CoNLL05 and CoNLL12 test datasets. For the sake of fair comparison, we split the table into three major rows, i.e., without PLMs, with ELMo, and with BERT. Due to space constraints, we leave the experiments on CPB1.0 to § C.

First, we can see that our proposed second-order model surpass previous BIO-based and Tuple-based methods, achieving new SOTA F_1 scores on all three test datasets and under all three settings. The Tuple-based model of He et al. (2018) is very competitive in its performance. Our second-order parser outperforms it by relatively large margin in F_1 only on CoNLL05-WSJ w/o PLMs (1.26) and on CoNLL05-Brown w/ ELMo (0.93). On other datasets and settings, the performance gap is in $[0.2, 0.3]$.

Second, we can see that the second-order model outperforms the first-order model in both precision and recall on almost all datasets and settings, showing that high-order structural information is always helpful. More concretely, under the setting of w/o PLMs, improvements in F_1 on CoNLL05-WSJ (0.7), on CoNLL05-Brown (0.4), and on CoNLL12 (0.7) are all significant at a confidence level of $p < 0.05$. Under the settings of w/ BERT, the improvement is 0.5 in F_1 on CoNLL12 at a more significant level of $p < 0.001$. And we find an interesting phenomenon that our model consistently achieves much higher precision scores but lower recall scores than that of He et al. (2018). We give the detailed analysis in Section 3.3.

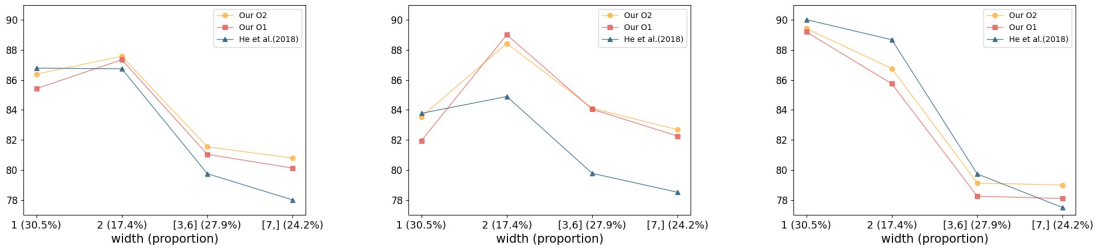
3.3 Performance Regarding Argument Width and Argument Type

In order to explore the differences between our method and previous methods, and the advantages of high-order model over first-order model, we make an+ in-depth analysis from the perspectives of argument width and argument type.

Performance Regarding Argument Width. As shown in Figure 6, we divide arguments into four categories according to their width, i.e., the number of words included, and report F_1 scores, precision and recall for each category. The proportion of each category in the gold-standard data is also reported. We obtain results of He et al. (2018) by re-running evaluation with their released model. We draw three clear and important findings.

Model	CoNLL05-WSJ				CoNLL05-Brown			CoNLL12			
	Dev.F_1	P	R	F_1	P	R	F_1	Dev.F_1	P	R	F_1
He et al. (2017)†	80.3	80.2	82.3	81.2	67.6	69.6	68.5	75.5	78.6	75.1	76.8
Strubell et al. (2018)† *	81.72	81.77	83.28	82.51	68.58	70.10	69.33	-	-	-	-
He et al. (2018)‡	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	-	-	-	-	-	-	-
Our O1	81.68	83.08	83.05	83.06	71.42	69.77	70.59	79.33	80.71	78.13	79.40
Our O2	**82.47**	**83.97**	83.56	**83.76**	**71.82**	70.19	**70.99**	**80.00**	**80.75**	79.46	**80.10**
+ELMo											
Strubell et al. (2018)† *	84.73	83.86	85.98	84.91	73.01	75.61	74.31	-	-	-	-
He et al. (2018)‡	85.3	84.8	87.2	86.0	73.9	**78.4**	76.1	83.0	81.9	**84.0**	82.9
Li et al. (2019)‡	-	85.2	**87.5**	**86.3**	74.7	78.1	76.4	-	**84.9**	81.4	83.1
Our O1	85.26	85.74	86.69	86.21	75.70	78.00	76.83	83.04	82.51	83.48	82.99
Our O2	**85.51**	**85.80**	86.80	**86.30**	**76.44**	77.63	**77.03**	**83.18**	82.79	83.45	**83.12**
+BERT											
Our O1	**86.14**	86.28	87.71	86.99	77.92	79.49	78.70	84.08	83.00	84.55	83.77
Our O2	**86.14**	**86.37**	**87.93**	**87.14**	**78.18**	**79.91**	**79.04**	**84.28**	**83.30**	**85.26**	**84.27**

Table 2: Results on CoNLL05 and CoNLL12 datasets. We mark BIO-based models by † and tuple-based ones by ‡. Moreover, we mark the results of Strubell et al. (2018) by * to indicate that we report corrected evaluation results after re-testing their released syntax-agnostic models, since they incidentally used a wrong evaluation procedure in their original paper, leading to much higher precision scores.



(a) F_1 score (b) P (precision) (c) R (recall)

Figure 6: Analysis of the arguments with different width. The horizontal axis denotes the width of arguments and the proportion of arguments of the same width in the data set. The vertical axis denotes the corresponding metrics, i.e., F_1, P, R.

First, both our first-order and second-order models perform better on multi-word arguments than He et al. (2018). This is kind of surprising, considering that the Tuple-based approach can explicitly represent whole arguments, whereas our graph parsing approach only models argument beginning and ending positions.

Second, compared with He et al. (2018), our second-order model achieves much higher precision scores on all multi-word arguments, while the drop in recall scores are relatively slight, 1.92 on two-word arguments, 0.6 on arguments containing $[3, 6]$ words. This directly explains why our models perform better in precision and worse in recall. Obviously, the reason is that our models predict less multi-word arguments with higher precision than He et al. (2018).

Third, we can see that the second-order model is always superior to the first-order model, except for precision over two-word arguments, indicating the high-order structural information is stably helpful.

Performance Regarding Argument Type. Figure 7 shows the performance of our models and He et al. (2018) on several different types of arguments with the highest frequency. First, by comparing our first-order and second-order models, we can see that second-order model is better than first-order in all kinds of arguments. Second, compared with He et al. (2018), we find another interesting phenomenon. Our model has a higher improvement on major arguments such as A0 and A1, especially on A2 (3.27 in F_1) . However, the advantage of our model in adjunct arguments such as AM-TMP, AM-MOD, and AM-ADV are not obvious. We think that this may be caused by the difference in

the width of different arguments. Considering the above analysis of arguments with different widths, which revealed that our model is better at dealing with long arguments. We compare the width of different arguments and find that the average width of major arguments and adjunct arguments are respectively 5.82 and 3.27. In particular, most A2 arguments have a width of 2, and most AM-ADV arguments have a width of 1. As shown in the Figure 6(a), our model performs better on arguments with width 2 and slightly worse on arguments with width 1.

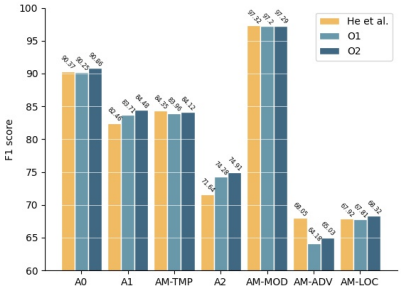


Figure 7: F_1 score of different types of arguments.

4 Related Works

Span-based SRL models. As two mainstream neural models, the BIO-based and Tuple-based approaches handle SRL in different ways. The BIO-based approach first recognizes predicates and then determines arguments for each predicate via sequence labeling. Zhou and Xu (2015) employ multi-layer BiLSTMs as the encoder and apply a CRF layer to find the best label sequence for each predicate. He et al. (2017) propose to use highway BiLSTMs (Srivastava et al., 2015) to alleviate the vanishing gradient problem, and use recurrent dropout (Gal and Ghahramani, 2016) to reduce over-fitting. Shi and Lin (2019) concatenate each predicate word after the original sentence to form the new input and use BERT(Devlin et al., 2019) and BiLSTM as the encoder.

He et al. (2018) propose the Tuple-based approach. The idea is directly predicting relations between candidate predicates (words) and arguments (word spans). Compared with the BIO-based approaches, the Tuple-based approach has the advantage of being able to flexibly represent whole argument. Li et al. (2019) extend the Tuple-based model to support both span-based and dependency-based SRL tasks. Zhou et al. (2020) propose a multi-task learning framework that does the SRL, dependency parsing, and constituent parsing simultaneously, and prove that semantic and syntax can benefit from each other.

SDGP models. SDGP (Oepen et al., 2014, 2015) uses graph to represent the semantic information of a sentence. Nodes correspond to single words, whereas edges and their labels denote semantic relationships. As a mainstream approach, the graph-based model finds the best graph from the fully connected graph. Dozat and Manning (2018) propose a simple and efficient SDGP parser. Wang et al. (2019) extend the model of Dozat and Manning (2018) by introducing second-order information. They compare two approximate high-order inference methods, i.e., mean filed variational inference and loopy belief propagation and find similar performance. In this work, we build our parser on the shoulder of these SDGP works.

The **dependency-based SRL** model of Li et al. (2020) is also related to our work. They directly apply the SDGP model of Wang et al. (2019) to the simpler dependency-based SRL. Please note that they adopt a pipeline (*not end-to-end*) framework by first predicting predicates with an independently trained sequence labeling model, and then recognizing arguments of all predicates via graph parsing. We give more discussion and performance comparison in the § D and § E.

5 Conclusions

This paper proposes a new graph representation schema for transforming raw span-based SRL structures to word-level graphs. Based on the schema, we cast the span-based SRL as a SDGP task and present a fast and accurate end-to-end parser. Moreover, we propose a simple post-processing method based on constrained Viterbi to handle conflicts in the output graphs. Experiments show that our parser 1) is much more efficient than previous parsers, and can parse over 600 sentences per second; 2) reaches new state-of-the-art performance on CoNLL05, CoNLL12, and CPB1.0 datasets. The in-depth analysis shows that compared with the representative and competitive Tuple-based approach of He et al. (2018), our graph parsing model is superior in recognizing multi-word arguments and able to recall fewer arguments with much higher precision. This clear finding may lead to some interesting future works, e.g., combining the power of the two different approaches.

References

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

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

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of ACL*, pages 484–490.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of NeuralPS*, pages 1019–1027.

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.

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

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.

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

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

Zuchao Li, Hai Zhao, Rui Wang, and Kevin Parnow. 2020. High-order semantic role labeling. In *Findings of EMNLP*, pages 1134–1151.

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

Diego Marcheggiani, Jasmijn Bastings, and Ivan Titov. 2018. Exploiting semantics in neural machine translation with graph convolutional networks. In *Proceedings of NAACL-HLT*, pages 486–492.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NeuralPS*, pages 3111–3119.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. Semeval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of SemEval*, pages 915–926.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of SemEval*, pages 63–72.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

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 EMNLP-CoNLL*, pages 1–40.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of ACL*, pages 1192–1202.

Lei Sha, Sujian Li, Baobao Chang, Zhifang Sui, and Tingsong Jiang. 2016. Capturing argument relationship for Chinese semantic role labeling. In *Proceedings of EMNLP*, pages 2011–2016.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Proceedings of NeuralPS*, pages 2377—-2385.

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.

Weiwei Sun, Zhifang Sui, Meng Wang, and Xin Wang. 2009. Chinese semantic role labeling with shallow parsing. In *Proceedings of EMNLP*, pages 1475–1483.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2015a. Machine comprehension with syntax, frames, and semantics. In *Proceedings of ACL-IJCNLP*, pages 700–706.

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.

Zhen Wang, Tingsong Jiang, Baobao Chang, and Zhifang Sui. 2015b. Chinese semantic role labeling with bidirectional recurrent neural networks. In *Proceedings of EMNLP*, pages 1626–1631.

Qiaolin Xia, Lei Sha, Baobao Chang, and Zhifang Sui. 2017. A progressive learning approach to Chinese SRL using heterogeneous data. In *Proceedings of ACL*, pages 2069–2077.

Qingrong Xia, Zhenghua Li, and Min Zhang. 2019a. A syntax-aware multi-task learning framework for Chinese semantic role labeling. In *Proceedings of EMNLP-IJCNLP*, pages 5382–5392.

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

Nianwen Xue. 2008. Labeling Chinese predicates with semantic roles. *Computational Linguistics*, 34(2):225–255.

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

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.

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

A Encoder

Input vectors. Following standard practice for SRL, the input of the i-th word is the concatenation of word embedding $\mathbf{e}_i^{word}$, lemma embedding $\mathbf{e}_i^{lemma}$, and charLSTM representation vector:

$$\mathbf{x}_i = \mathbf{e}_i^{word} \oplus \mathbf{e}_i^{lemma} \oplus \mathbf{e}_i^{char} \quad (10)$$

where $\mathbf{e}_i^{char}$ is the output vector of a one-layer BiLSTM that encodes the character sequence (Lample et al., 2016).

BiLSTM encoder. Then, a three-layer BiLSTM encoder produces a context-aware vector representation for each word.

$$\mathbf{h}_i = \mathbf{f}_i \oplus \mathbf{b}_i \quad (11)$$

where $\mathbf{f}_i$ and $\mathbf{b}_i$ respectively denote the output vectors of top-layer forward and backward LSTMs for w_i.

Model	F_1
end2end	
Xia et al. (2019a)	79.29
Our O1	79.36
Our O2	**80.42**
w/ pre-identified predicate	
Sun et al. (2009)	74.12
Wang et al. (2015b)	77.59
Sha et al. (2016)	77.69
Xia et al. (2017)	79.67
Xia et al. (2019a)	80.48
Our O1	80.06
Our O2	**81.30**

Table 3: F_1 scores on CPB1.0 test set.

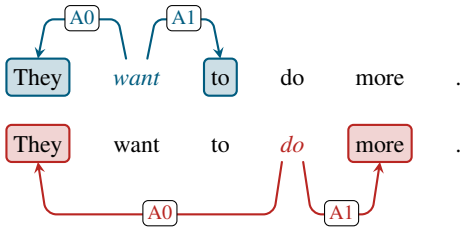
B Hyper-parameter settings

We employ 300-dimension English word embeddings from GloVe (Pennington et al., 2014) for English experiments. For Chinese, we train the word embeddings on Chinese Gigaword dataset[5] with word2vec (Mikolov et al., 2013). We directly adopt most hyper-parameters of the SDGP work of Wang et al. (2019), except that we reduce the dimension of Char-LSTM from 400 to 100 to save the memory, which only slightly influence performance. And under the setting of w/o PLMs, the number of parameters of the first-order model and second-order model is 189M and 200M respectively. For experiments with PLMS, we adopt ELMo[6] (Peters et al., 2018) and BERT[7] (Devlin et al., 2019) to get contextual word representation to boost the performance of our model. Following most of previous works (He et al., 2018; Xia et al., 2019b), for ELMo, we froze its parameters and concatenate its output with $\mathbf{x}_i$ to form the new input for the BiLSTM encoder. For BERT, we directly use it as our encoder and fine tune its parameters during training.

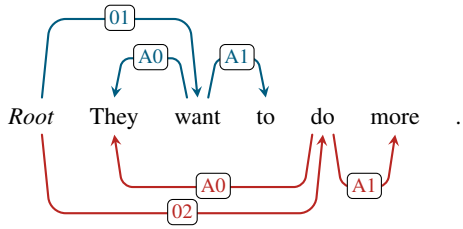
C Experiments on CPB1.0

Table 3 shows the comparison between our work and previous works on CPB1.0 test set. Because most of the previous work carried out experiments with given predicates, in order to compare with them, we also report the results of given predicates. Under the setting of given pre-identified predicate, we directly mask the output of our models with given predicates. Concretely, we use the given predicates as the predicted predicates. Then, we delete

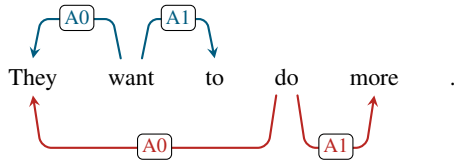
[5]https://catalog.ldc.upenn.edu/LDC2003T09
[6]https://allennlp.org/elmo
[7]https://huggingface.co/bert-large-uncased



(a) The original dependency-based SRL structure of the example sentence. "want" with sense label "01" and "do" with sense label "02" are two predicates.



(b) The graph representation in our model.



(c) The graph representation in Li et al. (2020). Li et al. (2020) only use it to predict arguments, and the predicates are predicted by another sequence labeling model.

Figure 8: The original SRL structure and its corresponding graph representation in our model and Li et al. (2020).

the arguments which belong to the wrongly predicted predicates. From the table, we can see that our second-order model has made important improvements compared with previous models both under the *end-to-end* and *w/ pre-identified predicate* setting. Specifically, 1.13 under *end-to-end* setting and 0.82 under *w/ pre-identified predicate* setting. In addition, consistent with the results on CoNLL05 and CoNLL12, the performance of our second-order model is also better than that of first-order model.

D Graph Representation of Dependency-based SRL

Figure 8(a) shows the original predicate-argument structure of the dependency-based SRL. Different from the span-based SRL, arguments in dependency-based SRL are only single words.

Consistent with our practice in span-based SRL, we also cast the dependency-based SRL task as a SDGP task. As shown in the Figure 8(b), we add a pseudo node "*Root*" and link all the predicates to it with their senses as edge labels. Then the argument words are linked to their corresponding predicate words with their semantic roles as edge labels. Since arguments contain only one word, and there exist no conflicts that are mentioned in span-based SRL, so we can directly recover the generated graph to the corresponding SRL structure with similar strategy used in span-based SRL.

Li et al. (2020) also form the dependency-based SRL task as a graph parsing task and introduce high-order information to their model too. But they only focus on dependency-based SRL. Figure 8(c) shows the graph representation in their model. First, unlike we predict predicates and arguments simultaneously by adding pseudo "*Root*" nodes, they need to predict predicates with another sequence labeling model in advance. The graph parsing model is only used to predict arguments in their approach. Second, the high-order information in their model is not as rich as that in our model since the lacking of the second-order structures regarding "*Root*", such as the grandchildren structure grd(*Root*, want, They) and the sibling structure sib(*Root*, want, do).

Model	WSJ			Brown		
	P	R	F_1	P	R	F_1
Cai et al. (2018)	84.70	85.20	85.00	-	-	72.50
Li et al. (2019)	-	-	85.10	-	-	-
Li et al. (2020)	86.26	86.06	86.16	74.76	73.65	74.20
Our O1	86.85	85.70	86.27	**76.02**	74.14	75.07
Our O2	**86.74**	**86.21**	**86.48**	75.83	**74.60**	**75.21**
+ELMo						
Li et al. (2019)	84.5	86.1	85.3	74.6	73.8	74.2
Li et al. (2020)	-	-	87.12	-	-	76.65
Our O1	87.54	88.41	87.97	**78.01**	78.65	78.33
Our O2	**87.70**	**88.73**	**88.21**	77.97	**79.31**	**78.63**
+BERT						
Li et al. (2020)	**88.77**	88.62	88.70	**80.01**	79.80	79.90
Our O1	87.01	**90.22**	88.59	78.62	**82.59**	**80.55**
Our O2	87.61	90.20	**88.89**	78.99	82.18	**80.55**

Table 4: Results on CoNLL09-en.

E Experiments on Dependency-based SRL

Experiments are conducted on the widely used CoNLL09 English dataset (Hajič et al., 2009) to verify the effectiveness of our approach on dependency-based SRL. We focus on end-to-end setting jointly predicting both predicates, the sense of predicates, arguments, and semantic roles of arguments. The hyper-parameters are the same as

that in the span-based SRL.

Table 4 shows the comparison between our models and previous state-of-the-art models. We can see that both our first-order model and second-order model outperform previous best models and achieve new state-of-art results on all datasets under all settings. Besides, as in the span-based SRL, our second-order always performs better than the first-order model except on Brown under the BERT setting, verifying the effectiveness of high-order information.

Compared with Li et al. (2020) which also introduces high-order information, our model performs better. We attribute it to the fact that their model is not a complete end-to-end model, i.e., they use another independently trained sequence labeling model to predict the predicates. So the high-order information cannot be used to help predicate prediction, and errors happen in predicate prediction will affect the subsequent argument prediction procedure, namely error propagation. However, in our model we conduct the predicate prediction and the argument prediction simultaneously and the predicate prediction procedure can also benefit from high-order information. In addition, there are no second-order structures that contain the node "*Root*" in their model, which leads to the high-order information their model can use is not as rich as ours.