End-to-end Semantic Role Labeling with Neural Transition-based Model

Hao Fei,1 Meishan Zhang,2 Bobo Li,1 Donghong Ji1∗

1 Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan, China
2 Department of School of New Media and Communication, Tianjin University, Tianjin, China
{hao.fei, boboli, dhji}@whu.edu.cn, mason.zms@gmail.com

Abstract
End-to-end semantic role labeling (SRL) has been received increasing interest. It performs the two subtasks of SRL: predicate identification and argument role labeling, jointly. Recent work is mostly focused on graph-based neural models, while the transition-based framework with neural networks which has been widely used in a number of closely-related tasks, has not been studied for the joint task yet. In this paper, we present the first work of transition-based neural models for end-to-end SRL. Our transition model incrementally discovers all sentential predicates as well as their arguments by a set of transition actions. The actions of the two subtasks are executed mutually for full interactions. Besides, we suggest high-order compositions to extract non-local features, which can enhance the proposed transition model further. Experimental results on CoNLL09 and Universal Proposition Bank show that our final model can produce state-of-the-art performance, and meanwhile keeps highly efficient in decoding. We also conduct detailed experimental analysis for a deep understanding of our proposed model.

Introduction
Semantic role labeling (SRL), as one of the core tasks to identify the semantic predicates in text as well as their semantic roles, has sparked much interest in natural language processing (NLP) community (Pradhan et al. 2005; Lei et al. 2015; Xia et al. 2019). SRL is a shallow semantic parsing, aiming to uncover the predicate-argument structures, such as ‘who did what to whom, when and where’. The task can be beneficial for a range number of downstream tasks, such as information extraction (Christensen et al. 2011; Bastianelli et al. 2013), question answering (Shen and Lapata 2007; Berant et al. 2013) and machine translation (Xiong, Zhang, and Li 2012; Shi et al. 2016).

Traditionally, SRL is accomplished via two pipeline steps: predicate identification (Scheible 2010) and argument role labeling (Pradhan et al. 2005). More recently, there is growing interest in end-to-end SRL, which aims to achieve both two subtasks by a single model (He et al. 2018). Given a sentence, the goal is to recognize all possible predicates together with their arguments jointly. Figure 1 shows an example of end-to-end SRL. The end-to-end joint architecture can greatly alleviate the error propagation problem, and meanwhile simplify the overall decoding process, thus receives increasing attention. Graph-based models have been the mainstream methods to end-to-end SRL, which are achieved by enumerating all the possible predicates and their arguments exhaustively (He et al. 2018; Cai et al. 2018; Li et al. 2019). Their results show that end-to-end modeling can obtain better SRL performance.

Alternatively, the transition-based framework offers another solution for end-to-end modeling, which is totally orthogonal to the graph-based models. Transition-based models have been widely exploited for end-to-end sequence labeling (Zhang and Clark 2010; Lyu, Zhang, and Ji 2016; Zhang, Zhang, and Fu 2018), structural parsing (Zhou et al. 2015; Dyer et al. 2015; Yuan, Jiang, and Tu 2019) and relation extraction (Wang et al. 2018; Zhang et al. 2019), which are closely related to SRL. These models can also achieve very competitive performances for a range of tasks, and meanwhile maintain high efficiencies with linear-time decoding complexity.

In this work, we present the first work of exploiting the neural transition-based architecture to end-to-end SRL. The model handles SRL incrementally by predicting a sequence of transition actions step by step, which are used to detect all predicates as well as their semantic roles in a given sentence. The two subtasks of end-to-end SRL, predicate identification and argument role labeling, are performed mutually in a single model to make full interactions of them. For argument role labeling, the recognition is conducted in a close-first way, where the near-predicate roles are processed first. The partial outputs of the incremental processing are denoted as transition states. In addition, we suggest explicit high-order compositions to enhance our transition-based model, lever-
aging the precedent partially-recognized argument-predicate structures for the current action classification.

Our neural transition system is built upon standard embedding-based word representations, and then is improved with dependency-aware representations by using recursive TreeLSTM (Tai, Socher, and Manning 2015). Concretely, we embed the surface words, characters, POS tags and dependency structures as input representations. During decoding, we represent the transition states by using standard BiLSTM (Hochreiter and Schmidhuber 1997) and Stack-LSTM (Dyer et al. 2015) to encode the elements in buffers and stacks, respectively. Finally, we predict transition actions incrementally based on the state representations.

We conduct experiments on dependency-based SRL benchmarks, including CoNLL09 (Hajič et al. 2009) for the English language, and Universal Proposition Bank (Akbik et al. 2015; Akbik and Li 2016) for seven other languages. Our end-to-end neural transition model wins the best results against the baselines, giving the state-of-the-art performances on both the predicate identification and argument role labeling, meanwhile keeping efficient on decoding. We also show that with recent contextualized word representations, e.g., ELMo (Devlin et al. 2019), BERT (Peters et al. 2018) or XLNet (Yang et al. 2019), the overall SRL performances can be further improved. In-depth analysis is conducted to uncover the important components of our final model, which can help comprehensive understanding of our model. Following we summarize our contributions:

- We fill the gap in the literature of employing neural transition-based model for end-to-end SRL. We also enhance the parsing procedure with a close-first scheme.
- We compose the high-order features (i.e., with one more predicate-role attachments from multiple predicates) in our transition framework for end-to-end SRL to model long-term substructure information explicitly.
- Our transition framework wins new state-of-the-art performances against all current graph-based methods on benchmark datasets, meanwhile being faster on decoding.

## Related Work

Gildea and Jurafsky (2000) pioneer the task of semantic role labeling, as a shallow semantic parsing. Approaches for SRL can be largely divided into two folds. Earlier efforts are paid for designing hand-crafted discrete features with machine learning classifiers (Pradhan et al. 2005; Punyakanok, Roth, and Yih 2008; Zhao et al. 2009). Later, A great deal of work takes advantages of neural networks with automatic distributed features (FitzGerald et al. 2015; Roth and Lapata 2016; Marcheggiani and Titov 2017; Strubell et al. 2018; Fei, Zhang, and Ji 2020). Also it is worth noticing that many previous work shows that integrating syntactic tree structure can greatly facilitate the SRL (Marcheggiani, Flovø, and Titov 2017; Zhang, Wang, and Si 2019; Fei et al. 2020).

Prior works traditionally separate the SRL into two individual subtasks, i.e., predicate disambiguation and argument role labeling. They mainly conduct argument role labeling based on the pre-identified predicate oracle. More recently, several researches consider the end-to-end solution that handles both two subtasks by one single model. All of them employs graph-based neural model, exhaustively enumerating all the possible predicate and argument mentions, as well as their relations (He et al. 2018; Cai et al. 2018; Li et al. 2019; Fei, Ren, and Ji 2020). The distinctions among them are quite slight, mostly lying in the encoder, or relation scorer. For example, He et al. (2018) use the BiLSTM (Hochreiter and Schmidhuber 1997) as base encoder, and use a feedforward layer as predicate-argument relation scorer, while Cai et al. (2018) employ a biaffine scorer, and Li et al. (2018) enhance the BiLSTM with highway connections.

Graph-based and transition-based models are always two mainstream approaches for structural learning. In this work, we employ a neural transition model, an algorithm that has been extensively exploited for a wide range of NLP parsing tasks, e.g., part-of-speech (POS) tagging (Zhang and Clark 2010; Lyu, Zhang, and Ji 2016), word segmentation (Zhang, Chang, and Fu 2016, 2018), dependency parsing (Zhou et al. 2015; Dyer et al. 2015; Yuan, Jiang, and Tu 2019) and information extraction (Wang et al. 2018; Zhang et al. 2019) etc. Note that we have no bias on the graph-based and transition-based models, and both the two kinds of models have their own advantages and disadvantages. Compared with graph-based models, one big advantage of transition-based approaches is that high-order features (i.e., subtrees, partial parsed results) can be explicitly modeled conveniently by using transition states (i.e., state representation). Currently there is no work for transition-based neural SRL. Besides, transition-based model can achieve highly competitive task performances meanwhile with good efficiency, i.e., linear-time encoding complexity.

It worth noticing that Choi and Palmer (2011) employ a transition module for handling SRL. Unfortunately their work is based on discrete features, while ours is in the literature the first neural transition-based work. We also present two enhancements for the task, i.e., 1) close-first parsing scheme, and 2) high-order feature composition. Besides, our work focuses on end-to-end SRL, but Choi and Palmer (2011) assume that predicates are already given.

## Method

Following prior end-to-end SRL work (He et al. 2018; Li et al. 2019), we treat the task as predicate-argument-role triplets prediction. Given an input sentence $S = \{w_1, \ldots, w_n\}$, the transition system is expected to output a set of triplets $Y \in P \times A \times R$, where $P = \{p_1, \ldots, p_k\}$ are all possible predicate tokens, $A = \{a_1, \ldots, a_l\}$ are all associated argument tokens, and $R$ are the corresponding role labels for each $a_i$. Technically, we model $Y$ as a graph structure, where the nodes are predicates or arguments, and the directed edges represent the roles from predicate to argument. For example the sentence in Figure 1, the final outputs are four triplets: `<stay, silent, A3>`, `<stay, victims, A1>`, `<changes, stay, A-TMP>` and `<changes, nothing, A1>`.

In this section, we first elaborate the transition system including the states and actions. We then describe the vanilla neural transition model, upon which we further introduce the high order model.

12804
<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>$\sigma_l$</th>
<th>$\alpha_l$</th>
<th>$\lambda_p$</th>
<th>$\alpha_r$</th>
<th>$\sigma_r$</th>
<th>$\beta$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>[1]</td>
<td>[1]</td>
<td>Null</td>
<td>[1]</td>
<td>[9, 2]</td>
<td>[1]</td>
<td>$\Phi$</td>
</tr>
<tr>
<td>1</td>
<td>NO-PRD</td>
<td>[1]</td>
<td>[1]</td>
<td>Null</td>
<td>[1]</td>
<td>[9, 3]</td>
<td>[2, 9]</td>
<td>$\Phi$</td>
</tr>
<tr>
<td>4</td>
<td>PRD-GEN</td>
<td>[1, 2, 3, 4]</td>
<td>[5]</td>
<td>[9, 8, 7, 6]</td>
<td>[5, 9, 3]</td>
<td>$\Phi$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>NO-ARC</td>
<td>[1, 2, 3]</td>
<td>[4]</td>
<td>[9, 8, 7, 6]</td>
<td>[5, 9, 3]</td>
<td>$\Phi$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Right-ARC</td>
<td>[1, 2, 3]</td>
<td>[5]</td>
<td>[9, 8, 7]</td>
<td>[5, 9, 3]</td>
<td>$\Phi$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>LEFT-ARC</td>
<td>[1]</td>
<td>[4, 3, 2]</td>
<td>[5, 6]</td>
<td>[9, 8]</td>
<td>[5, 9, 3]</td>
<td>$\Phi$</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>SHIFT</td>
<td>[1, 5]</td>
<td>[1]</td>
<td>Null</td>
<td>[1]</td>
<td>[9, 8]</td>
<td>[6, 7, 8, 9]</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>NO-PRD</td>
<td>[1, 6]</td>
<td>[1]</td>
<td>Null</td>
<td>[1]</td>
<td>[9, 8]</td>
<td>[7, 8, 9]</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>PRD-GEN</td>
<td>[1, 8]</td>
<td>[1]</td>
<td>[9]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LEFT-ARC</td>
<td>[1, 7]</td>
<td>[8]</td>
<td>[9]</td>
<td>[1]</td>
<td>[9]</td>
<td>$\Phi$</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>LEFT-ARC</td>
<td>[1, 2, 3, 4]</td>
<td>[8, 6, 5]</td>
<td>[9]</td>
<td>[1]</td>
<td>[9]</td>
<td>$\Phi$</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>SHIFT</td>
<td>[1, 9]</td>
<td>[1]</td>
<td>Null</td>
<td>[1]</td>
<td>[1]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) The transition system.

Figure 2: Illustration of the transition framework, where the input sequence is the same as that in Figure 1. For brevity, here we omit the role labeling operation, as it is performed with LEFT/RIGHT-ARC actions synchronously.

### Transition System

Our transition framework transforms the structural learning into a sequence of action predictions. The system consists of actions and states. The actions are used to control state transitions, while the states store partially parsed results.

#### States

We define the transition states as $s = (\sigma_l, \alpha_l, \lambda_p, \alpha_r, \sigma_r, \beta, Y)$. $\sigma_l$ and $\sigma_r$ are stacks holding processed arguments in left-side ($l$) and right-side ($r$), respectively, $\alpha_l$ and $\alpha_r$ are the corresponding stacks temporarily storing the arguments popped out of $\sigma_r$ ($* \in \{l, r\}$) which will be pushed back later. $\lambda_p$ is a variable referring to a candidate predicate. $\beta$ is a buffer loading the unprocessed words in a sentence. $Y$ stores the output triplets.

#### Actions

We design total six actions as follows.

- **NO-PRD**, current candidate $w_i$ is not a predicate, so moves $w_i$ from $\beta$ to $\sigma_l$ while popping the top element out of $\sigma_r$.
- **PRD-GEN**, current candidate $w_i$ is a predicate (i.e., $p_j$), so generate representation for $p_j$ onto $\lambda_p$.
- **LEFT-ARC**, the top element in $\sigma_l$ is an argument (i.e, $a_i$), so assign an arc between $p_j$ and $a_i$, and pop the top element out of $\sigma_l$ into $\alpha_l$.
- **RIGHT-ARC**, the top element in $\sigma_r$ is a argument (i.e, $a_i$), so assign an arc between $p_j$ and $a_i$, and pop the top element out of $\sigma_r$ into $\alpha_r$.
- **NO-ARC**, no arc between $p_j$ and the top element in $\sigma_l$ or $\sigma_r$, so pop the top elements out of $\sigma_l$ or $\sigma_r$ into $\alpha_l$ or $\alpha_r$.
- **SHIFT**, end of the detection of arguments for $p_j$. Pop all the elements out of $\alpha_{*}$ into $\sigma_{*}$, clear $\lambda_p$, and moves $w_i$ from $\beta$ to $\sigma_l$ while popping the top element out of $\sigma_r$.

### Parsing scheme

The vanilla transition framework generally process the words in a sentence incrementally from left to right, while such reading scheme may be rigid for SRL parsing. Intuitively, it is always easier for humans to first grasp the core ideas then dig into more details. To this end, we consider a close-first parsing procedure (Goldberg and Elhadad 2010; Cai and Lam 2019; Kurita and Sogada 2019). We first perform recognition for any possible predicate, e.g., $p_j$, and then the process of detecting the arguments of $p_j$ starts down the path from near-to-$p_j$, to far-to-$p_j$, until all the arguments are determined.

Given a sentence, before conducting the transitions, the $\sigma_r$ will pre-load all the words from $\beta$ except the first word in reverse order. When a predicate is determined, the system starts searching for its arguments in leftward and rightward alternatively, from near to far. Once an arc is confirmed, a labeler assigns a role label for the argument. It is worth noticing that only a subset of actions are legal to a certain transition state. And invalid actions can lower encoding efficiency, and lead to bad directed graph structure. For example, a NO-PRD action must be followed by either NO-PRD or PRD-GEN action, and *-ARC must starts by a PRD-GEN action. We thus also design some constraints for each action. Figure 2(a) shows the transition process.

### The Vanilla Neural Model

#### Word representation

We first use the word form representation under two types of settings: $v^w_{i}$, $\tilde{v}^w_{i}$, where $v^w_i$ is the representation initialized with a pre-trained embedding vector for word $w_i$, which further can be trained, and $\tilde{v}^w_i$ is the fixed version. We also use convolutional neural networks (CNNs) to encode characters inside a word $w_i$ into character-level representation $v^c_i$. In addition, given the in-
put sentence $S$, we exploit the POS tag of the tokens, and use the embedding $v^w_{t}$ via a lookup table.

**Dependency features enhancement.** Besides, we employ a TreeLSTM (Tai, Socher, and Manning 2015; Miwa and Bansal 2016) to encode the dependency structural feature representation $v^{syn}_{t}$ for each corresponding word $w_t$. We concatenate these representations into unified input representation: $x_t = [v^{w}_{t}; v^{a}_{t}; v^{i}_{t}; v^{syn}_{t}; v^{pos}_{t}]$. Then, a BiLSTM is used as encoder for learning the context representations. We then concatenate two hidden states at each time step $i$ as the sequence representation: $h_i = \{h_1, h_2, \cdots, h_n\}$, which will be used for state representation learning.

**State representation.** The actions are predicted based on the neural transition state representations, as depicted in Figure 2(b). We first use a BiLSTM to generate representations $p_{a}$ for each word $w_t$ in $\beta$. Another BiLSTM is used for generating predicate representation $p_{p}$ at $\lambda$. Taking the 9-th transition step as example, the current representations of predicate ‘stay’ in $\lambda$ and the word ‘witnesses’ in $\beta$ are from BiLSTMs, respectively.

We use Stack-LSTM (Zhang et al. 2019; Yuan, Jiang, and Tu 2019) to encode the elements in stacks (e.g., $\sigma_{t}$, $\alpha_{t}$). For instance, the representations of the top elements in stacks (i.e., ‘When’ in $\sigma_{t}$, ‘witnesses’ in $\alpha_{t}$, ‘nothing’ in $\sigma_{r}$, ‘.’ in $\alpha_{r}$) is obtained from the Stack-LSTM, respectively. Besides of the regular states introduced in $s$, we also maintain the trace of action histories $\delta$ via a stack. We summarize all these state representations via concatenation as the initiation for the next action prediction:

$$g_t = [p^\sigma_{t}; p^\alpha_{t}; p^\sigma_{r}; p^\alpha_{r}; p^\lambda_{p}; p^\beta_{p}; p^\delta_{p}].$$  \hspace{1cm} (1)

**Action prediction.** Based on state $g_t$, the model predicts the action probability, as in Figure 2(b):

$$P^a = \text{softmax}(\text{FFNs}(g_t)),$$  \hspace{1cm} (2)

where FFNs() is a feed-forward network. At the meantime, once an arc is determined, a labeler$^1$ will assign a role label for the argument:

$$P^r = \text{softmax}(\text{FFNs}(g_t)).$$  \hspace{1cm} (3)

**Training.** During training, we set the goal to maximize the likelihood of actions from the current states under the gold-standard actions. We first convert gold output structures of training data into action sequences. We minimize the following loss:

$$\mathcal{L} = -\sum_{e} \hat{\varphi}_t \log p(\varphi_t | \Theta) + \frac{\zeta}{2} \| \Theta \|^2,$$  \hspace{1cm} (4)

where $\hat{\varphi}$ denotes gold-standard actions, $\Theta$ is the set of all model parameters, and $\zeta$ is a co-efficient. Instead of greedily choosing the output with maximum probability, we apply beam-search algorithm (Zhang and Clark 2008; Lyu, Zhang, and Ji 2016) for decoding, where top-B partial outputs are cached for each transition decision, avoiding achieving sub-optimal results.

**The High-Order Model**

In the vanilla transition model, the actions are incrementally predicted with the first-order local features, considering the dependencies to the predicates only. Here we present a high-order model further, leveraging the previously recognized argument-predicate information into the current decision. Figure 3 shows the overall network structure of our high-order features. We exploit the argument-predicate scoring distributions as the major sources for high-order compositions, which is mostly motivated by Lyu, Cohen, and Titov (2019). Two types of argument-predicate scoring distributions (i.e., $I^a$ and $I^r$) are exploited for action ($a$) and role ($r$) label predictions, respectively, and their network compositions are exactly the same. In the following, we describe the high-order networks in detail.

Concretely, the high-order features are calculated as follows. First, we derive a sequence of distribution-based features (i.e., $M^r = \{I^r_1, \cdots, I^r_{t-1}, I^r_t\}$ ) from the historical argument-predicates. For each $i \in [1, t]$, we have:

$$I^r_i = \text{softmax}(\text{FFN}(W_{a}^{r} [r_{a}^{i}; r_{p}^{i}])), \hspace{1cm} (5)$$

where $r_{a}, r_{p}$ are the current argument and predicate representation from $\sigma_{t}$ and $\lambda_{t}$, respectively, $r \in \{a, r\}$ indicating that the goal of the feature is for either action predication or role labeling. Note that the output dimension is the same as their final classification goal for $P^{r}$ (see Eq 2 and 3).

Second, we use an attention mechanism guided by the first-order representation $g_{t+1}$ to obtain the high-order feature vector:

$$u^r_{k} = \tan(W_{2}^{r} g_{t+1} + W_{2}^{r} I^r_{k});$$

$$\sigma^r_{k} = \text{softmax}(u^r_{k});$$  \hspace{1cm} (6)

where $\sigma^r_{k}$ is the desired representation, and $W_{1}^{r}$ and $W_{2}^{r}$ are parameters. Finally, we concatenate the first-order representation $g_{t+1}$ with high-order features $u^r_{k}$ into a single representation $h_{t+1}$:

$$h_{t+1} = g_{t+1} \oplus \sigma^r_{k},$$  \hspace{1cm} (7)
Table 1: Results on CoNLL09 English in-domain test set. Decode means decoding speed (token per second).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>P</td>
<td>R</td>
<td>F1</td>
<td>F1</td>
</tr>
<tr>
<td>He et al. (2018)</td>
<td></td>
<td>89.0</td>
<td>87.6</td>
<td>88.9</td>
<td>91.3</td>
</tr>
<tr>
<td>Cai et al. (2018)</td>
<td></td>
<td>89.5</td>
<td>88.4</td>
<td>89.2</td>
<td>94.6</td>
</tr>
<tr>
<td>Li et al. (2019)</td>
<td>89.9</td>
<td>89.2</td>
<td>89.6</td>
<td>95.0</td>
<td></td>
</tr>
<tr>
<td>Ours (Vanilla model)</td>
<td>88.0</td>
<td>88.9</td>
<td>88.4</td>
<td>92.3</td>
<td></td>
</tr>
<tr>
<td>Ours (High-Order model)</td>
<td>89.4</td>
<td>91.1</td>
<td>88.9</td>
<td>95.2</td>
<td></td>
</tr>
<tr>
<td><strong>Arg. Prd.</strong></td>
<td></td>
<td><strong>91.0</strong></td>
<td><strong>96.3</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Ablation results (F1 score).

<table>
<thead>
<tr>
<th></th>
<th>Arg.</th>
<th>Prd.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours (High-Order model)</td>
<td><strong>91.0</strong></td>
<td><strong>96.3</strong></td>
</tr>
</tbody>
</table>

and high-order feature representation together (i.e., \( \hat{g}_{t+1} = [g_{t+1}, \alpha_{t+1}] \)) for transition action and role label predictions.

In the high-order model, the action prediction, decoding and learning processing are kept consistent with the vanilla neural model in §. But with the above high-order feature, the recognition for the following and farther arguments will be informed by the earlier decisions, avoiding the error accumulation in the vanilla transition system. It also worth noticing that with beam-search strategy, we are able to facilitate the use of high-order feature, making the information communication between arguments most efficient.

### Experiments

#### Settings

We employ two dependency-based SRL benchmarks: CoNLL09 English, and Universal Proposition Bank (UPB) for other total seven languages. We use the pre-trained fasttext word embeddings for each language as default word representation. The hidden sizes in BiLSTM, TreeLSTM and Stack-LSTM are 200. We use the two layer version of BiLSTM, Stack-LSTM. The dimension of transition states is 150 universally. We adopt the Adam optimizer with initial learning rate of 1e-5. \( \zeta \) is 0.2, and beam size \( B \) is 32, according to our developing experiments. We train the model by mini-batch size in [16,32] with early-stop strategy. We evaluate the contextualized word representations, i.e., ELMo\(^3\), BERT (base-cased-version)\(^4\) and XLNet (base-version)\(^5\). Each dataset comes with its own train, develop and test sets. We use the precision (P), recall (R) and F1 score as the metric. We conduct significance tests via Dan Bikel’s randomized evaluation comparator. Our codes are open at https://github.com/scofield7419/TransitionSRL.

#### Ablation Study

We study the contributions from each part of our methods based on CoNLL09 in-domain developing set, in Table 2.

**Input features.** When removing the character encoder, POS tags and syntactic dependencies, respectively, the performances drop consistently. We notice that the dependency structural knowledge is of the greatest importance for both the argument and predicate recognition, among other features, which coincides with the prior findings of the structural syntax information (Marcheggiani, Frolov, and Titov 2017; Zhang, Wang, and Si 2019; Fei et al. 2020).

**High-order feature.** We ablate the argument-predicate scoring distribution \( \alpha^a \) and \( \alpha^r \) to see the contribution of the proposed interaction mechanism. We can find that both these two distributions plays key role for arguments role labeling, especially the \( \alpha^r \). When removing such incremental high-order feature away (i.e., vanilla model w/o \( \alpha^a \& \alpha^r \)), the results drop significantly. This verifies the usefulness of the proposed high-order feature.

**Transition parsing order.** Our model performs argument parsing around the predicate down the ‘from-near-to-far’ direction. Compared with the traditional ‘left-to-right’ reading order, or the reversed ‘right -to-left’ order, we can find that the close-first parsing scheme is much more effective.

#### Main Results

**CoNLL09 in-domain results.** We mainly make comparisons with the recent previous end-to-end SRL models. In

---

\(^3\)https://allennlp.org/elmo

\(^4\)https://github.com/google-research/bert

\(^5\)https://github.com/zihangdai/xlnet
Table 3: End-to-end SRL on UPB data. Values are F1 scores for argument recognition and role labeling.

<table>
<thead>
<tr>
<th></th>
<th>Arg.</th>
<th>Prd.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE</td>
<td>FR</td>
<td>IT</td>
</tr>
<tr>
<td>He et al. (2018)</td>
<td>68.5</td>
<td>84.3</td>
</tr>
<tr>
<td>Cai et al. (2018)</td>
<td>69.9</td>
<td>85.5</td>
</tr>
<tr>
<td>Li et al. (2019)</td>
<td>69.7</td>
<td>85.0</td>
</tr>
<tr>
<td>Ours (Vanilla model)</td>
<td>68.9</td>
<td>86.2</td>
</tr>
<tr>
<td>Ours (High-Order model)</td>
<td><strong>70.5</strong></td>
<td><strong>87.8</strong></td>
</tr>
</tbody>
</table>

Table 4: Results on out-of-domain data of CoNLL09.

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhao et al. (2009)</td>
<td>-</td>
<td>-</td>
<td>85.4</td>
</tr>
<tr>
<td>Björkelund et al. (2010)</td>
<td>87.1</td>
<td>84.5</td>
<td>85.8</td>
</tr>
<tr>
<td>FitzGerald et al. (2015)</td>
<td>-</td>
<td>-</td>
<td>86.7</td>
</tr>
<tr>
<td>Roth and Lapata (2016)</td>
<td>88.1</td>
<td>85.3</td>
<td>86.7</td>
</tr>
<tr>
<td>Marcheggiani and Titov (2017)</td>
<td>89.1</td>
<td>86.8</td>
<td>88.0</td>
</tr>
<tr>
<td>He et al. (2018)</td>
<td>89.7</td>
<td>89.6</td>
<td>89.7</td>
</tr>
<tr>
<td>Cai et al. (2018)</td>
<td>89.9</td>
<td>90.2</td>
<td>90.0</td>
</tr>
<tr>
<td>Li et al. (2019)</td>
<td><strong>90.9</strong></td>
<td>90.2</td>
<td>90.5</td>
</tr>
<tr>
<td>Ours (High-Order model)</td>
<td>-</td>
<td>-</td>
<td><strong>90.7</strong></td>
</tr>
<tr>
<td>Ours (High-Order model)</td>
<td>-</td>
<td>-</td>
<td><strong>90.7</strong></td>
</tr>
</tbody>
</table>

Table 5: Pipeline argument role labeling on CoNLL09.

Table 6: Results with contextualized word representation.

Table 1, first of all, we find that the syntactic dependency feature configurations universally contribute to both the argument recognition and predicate disambiguation, compared with the syntax-agnostic models. Most importantly, our high-order model outperforms these baselines on both two subtasks. With dependency features, the improvements are more significant, i.e., 90.2% (Arg.) and 95.5% (Prd.). Also our model is especially better at predicate disambiguation. Besides, the transition systems beat competitors with higher decoding speed, nearly two times faster.

Results for other languages. Table 3 shows the performances on UPB for other languages (with dependency features). Overall, our high-order model wins the best averaged F1 score (74.2%). Also the high-order feature universally helps to improve the vanilla transition model with average 1.3%(=74.2-72.9) F1 score. The improvements so far by our model demonstrate its effectiveness.

CoNLL09 out-of-domain results. Table 4 compares the performances on out-of-the-domain test set of CoNLL09 data with dependency features. Overall, the similar trends are kept as that on in-domain dataset. Our model still performs the best, yielding 80.0% (Arg.) and 82.7% (Prd.) F1 scores, verifying its generalization capability on capturing the semantic preferences of the task.

Argument role labeling. We next compare the results in Table 5 for standalone argument role labeling, where the models are given by the gold pre-identified predicates as input, and output the argument boundaries and roles. Compared with the baseline pipeline systems, the end-to-end systems can perform better, while our transition model gives a new state-of-the-art 90.7% F1 score.

Contextualized word representation. Finally, we take advances of the recent contextualized word representations, ELMo, BERT and XLNet. As shown in Table 6, all the end-to-end SRL models can benefit a lot. With enhanced word representations, we see that the graph-based model by Li et al. (2019) can achieve better results (91.5%) with ELMo, while our model can obtain the best performances by BERT (92.2%), and XLNet (92.0%).

12808
I expect to send them out to you sometime tomorrow.

Necessity of back refining. We introduce the incremental high order interaction for improving the argument recognition, by informing the detection of following and farther arguments with previous recognition history information, that is, the later decisions are influenced by former decisions. Such uni-directional propagation naturally spurs a potential question: Is it necessary to use the later refined decisions to inform and refine back the former decisions? Theoretically speaking, the negative influences only exist at the initial transition decisions, because our proposed interaction mechanism combined with beam search strategy can largely refine the future decisions. One the other hand, the close-first parsing scheme can ensure that the nearer-to-predicate arguments recognized earlier are more likely to be correct. In addition, in our experiment we proves the unnecessity of the back refining. Figure 4 shows the performance under varying surface distance from predicate to arguments. It is clear that within the distances of 3-4, the performances by our full model changes very little. Even when removing the interaction mechanism (into vanilla model), the decreases under short surface distances is not much significant. Lastly, our high-order model handles better the long-distance dependency issues, compared with baselines.

High-order features. We now look into the incremental high order interaction mechanism. We consider conducting empirical visualization of the transition parsing steps. We compare the transition results by high-order model, and the vanilla model without high-order feature, as illustrated in Figure 5. At the earlier steps with short distance between predicate and argument, e.g., second and fourth iteration steps, both two transition models can make correct prediction. At 7th step, the vanilla model wrongly assigns a A3 for argument ‘you’, and further falsely determines the ‘some-time’ as AM-TMP argument at 8th step. On the contrary, our high-order model with such interaction can perform correct role labeling, and yield accurate argument detection even in remote distance.

Role violation. To further explore the strengths of our transition model, we study the argument role violation (Punyakanok et al. 2004; FitzGerald et al. 2015; He et al. 2018). Consider categorizing the role labels into three types: (U) unique core roles (A0-A5, AA), (C) continuation roles and (R) reference roles. 1) If a core role appears more than once, U is violated. 2) If C-X role not precedes by the X role (for some X), C is violated; 3) R is violated if R-X role does not appear. Table 7 shows the role violation results.

First of all, our high-order model gives the most consistent results with gold ones, with minimum role violations. This partially indicates that the system can learn the latent constraints. But without the incremental high order interaction, performances by vanilla model for continuation and reference roles get hurt. This is reasonable, since generally the non-core roles are more remote from their predicate. We also find that our transition model with close-first parsing order performs better, especially for the unique core role, compared with the ‘left-to-right’ parsing order. Finally, the graph-based baselines tend to predict more duplicate core arguments, and also recognize fewer reference argument roles.

Table 7: Results for argument role violation.

<table>
<thead>
<tr>
<th></th>
<th>U</th>
<th>C</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold</td>
<td>55</td>
<td>0</td>
<td>88</td>
</tr>
<tr>
<td>Ours (High-Order model)</td>
<td>82</td>
<td>8</td>
<td>98</td>
</tr>
<tr>
<td>Ours (Vanilla model)</td>
<td>106</td>
<td>10</td>
<td>122</td>
</tr>
<tr>
<td>left-to-right Trans.</td>
<td>212</td>
<td>13</td>
<td>139</td>
</tr>
<tr>
<td>He et al. (2018)</td>
<td>242</td>
<td>14</td>
<td>208</td>
</tr>
<tr>
<td>Cai et al. (2018)</td>
<td>210</td>
<td>11</td>
<td>162</td>
</tr>
</tbody>
</table>

Discussion

We investigated a neural transition model for end-to-end semantic role labeling. We designed the transition system where the predicates were first discovered one by one, and then the associated arguments were determined in a from-near-to-far order. We proposed to use the incremental high-order feature, leveraging the previously recognized argument-predicate scoring distribution into the current decision. Experimental results on the dependency-based SRL benchmarks, including CoNLL09 and Universal Proposition Bank datasets, showed that the transition model brings state-of-the-art performances, meanwhile keeping higher decoding efficiency. Further analysis demonstrated the usefulness of the high-order feature and the close-first parsing order.
Acknowledgments
We thank the anonymous reviewers for providing constructive comments and suggestions. This work is supported by the National Natural Science Foundation of China (No. 61772378 and 61602160), the National Key Research and Development Program of China (No. 2017YFC1200500), the Research Foundation of Ministry of Education of China (No. 18JZD015).

References


