Better Combine Them Together! Integrating Syntactic Constituency and Dependency Representations for Semantic Role Labeling

Hao Fei[1] and **Shengqiong Wu**[1] and **Yafeng Ren**[2] and **Fei Li**[1] and **Donghong Ji**[1*]
1. Department of Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, China
2. Laboratory of Language and Artificial Intelligence, Guangdong University of Foreign Studies, China
{hao.fei,whuwsq,renyafeng,lifei_csnlp,dhji}@whu.edu.cn

Abstract

Structural syntax knowledge has been proven effective for semantic role labeling (SRL), while existing works mostly use only one singleton syntax, such as either syntactic dependency or constituency tree. In this paper, we explore the integration of heterogeneous syntactic representations for SRL. We first consider a TreeLSTM-based integration, collaboratively learning the phrasal boundaries from the constituency and the semantic relations from dependency. We further introduce a label-aware GCN solution for simultaneously modeling the syntactic edges and labels. Experimental results demonstrate that by effectively combining the heterogeneous syntactic representations, our methods yield task improvements on both span-based and dependency-based SRL. Also our system achieves new state-of-the-art SRL performances, meanwhile bringing explainable task improvements.

1 Introduction

Semantic role labeling (SRL) aims to disclose the predicate-argument structure of a given sentence. Such shallow semantic structures have been shown highly useful for a wide range of downstream tasks in natural language processing (NLP), such as information extraction (Fader et al., 2011; Bastianelli et al., 2013), machine translation (Xiong et al., 2012; Shi et al., 2016) and question answering (Maqsud et al., 2014; Xu et al., 2020). Based on whether to recognize the constituent phrasal span or the syntactic dependency head token of an argument, prior works categorize SRL into two types: the span-based SRL popularized in CoNLL05/12 shared tasks (Carreras and Màrquez, 2005; Pradhan et al., 2013), and the dependency-based SRL introduced in CoNLL08/09 shared tasks (Surdeanu

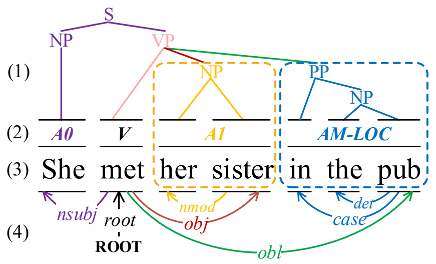


Figure 1: The mutual benefit to integrate both the (1) syntactic constituency and (4) dependency structures for (2) SRL, based on (3) an example sentence.

et al., 2008; Hajič et al., 2009). By adopting various neural network methods, two types of SRL have achieved significant performances in recent years (FitzGerald et al., 2015; He et al., 2017; Fei et al., 2021a)

Syntactic features have been extensively verified to be highly effective for SRL (Pradhan et al., 2005; Punyakanok et al., 2008; Marcheggiani and Titov, 2017; Strubell et al., 2018; Zhang et al., 2019). In particular, syntactic dependency features have gained a majority of attention, especially for the dependency-based SRL, considering their close relevance with the dependency structure (Roth and Lapata, 2016; He et al., 2018; Xia et al., 2019; Fei et al., 2021b). Most existing works focus on designing various methods for modeling the dependency representations into the SRL learning, such as TreeLSTM (Li et al., 2018; Xia et al., 2019) and graph convolutional networks (GCN) (Marcheggiani and Titov, 2017; Li et al., 2018). On the other hand, some efforts try to encode the constituency representations for facilitating the span-based SRL (Wang et al., 2019; Marcheggiani and Titov, 2020).

Yet almost all the syntax-based SRL methods use one standalone syntactic tree, i.e., either dependency or constituency tree. Constituent and dependency syntax actually depict the syntactic structure from different perspectives, and integrat-

[*] Corresponding author.

ing these two heterogeneous representations can intuitively bring complementary advantages (Farkas et al., 2011; Yoshikawa et al., 2017; Zhou and Zhao, 2019). As exemplified in Figure 1, the dependency edges represent the inter-relations between arguments and predicates, while the constituency structure[1] locates more about phrase boundaries of argument spans, and then directs the paths to the predicate globally. Interacting these two structures can better guide the system to focus on the most proper granularity of phrasal spans (as circled by the dotted box), while also ensuring the route consistency between predicate-argument pairs. Unfortunately, we find that there are very limited explorations of the heterogeneous syntax integration in SRL. For instance, Li et al. (2010) manually craft two types of discrete syntax features for statistical model, and recently Fei et al. (2020a) implicitly distill two heterogeneous syntactic representations into one unified neural model.

In this paper, we present two innovative neural methods for explicitly integrating two kinds of syntactic features for SRL. As shown in Figure 2, in our framework, the syntactic constituent and dependency encoders are built jointly as a unified block (i.e., Heterogeneous Syntax Fuser, namely `HeSyFu`), and work closely with each other. In the first architecture of `HeSyFu` (cf. Figure 3), we take two separate TreeLSTMs as the structure encoders for two syntactic trees. Based on our framework, we try to answer the following questions:

► **Q1**. *Whether the combination of constituent and dependency syntax can really improve SRL*?

► **Q2**. If yes, *how much will such improvements be for the dependency- and span-based SRL*?

We further propose *Const* GCN and *Dep* GCN encoders to enhance the syntax encoding in `HeSyFu`, where the *syntactic labels* (i.e., dependent arc types and constituency node types) are modeled in a unified manner within the label-aware GCN, as illustrated in Figure 4. With this, we can dig deeper:

► **Q3**. *How different will the results be by employing the TreeLSTM or GCN encoder*?

► **Q4**. *Can SRL be further improved by leveraging syntactic labels?*

► **Q5**. *What kind of associations can be discovered between SRL structures and these heterogeneous syntactic structures*?

[1]Following Marcheggiani and Titov (2020), we strip off the nodes of POS tags from the constituency tree for brevity.

To find the answers, we conduct extensive experiments on both span- and dependency-based SRL benchmarks (i.e., CoNLL05/12 and CoNLL09). The results and analyses show that,

►**A1**. *combining two types of syntax information is more helpful than just using either one of them*;

►**A2**. *the improvement for span-based SRL is more obvious than dependency-based one*;

►**A3**. *GCN performs better than TreeLSTM*;

►**A4**. *syntactic labels are quite helpful for SRL*;

►**A5**. *SRL and both kinds of syntactic structures have strong associations and should be exploited for mutual benefits*.

In our experiments, our SRL framework with two proposed `HeSyFu` encoders achieves better results than current best-performing systems, and yield more explainable task improvements.

2 Related Work

The SRL task, uncovering the shallow semantic structure (i.e. '*who did what to whom where and when*') is pioneered by Gildea and Jurafsky (2000), and popularized from PropBank (Palmer et al., 2005) and FrameNet (Baker et al., 1998). SRL is typically divided into the span-based one and dependency-based one on the basis of the granularity of arguments (e.g., phrasal spans or dependency heads). Earlier efforts focus on designing hand-crafted features with machine learning methods (Pradhan et al., 2005; Punyakanok et al., 2008; Zhao et al., 2009b,a). Later, SRL works mostly employ neural networks with distributed features for the task improvements (FitzGerald et al., 2015; Roth and Lapata, 2016; Marcheggiani and Titov, 2017; Strubell et al., 2018). Most high-performing systems model the task as a sequence labeling problem with *BIO* tagging scheme for both two types of SRL (He et al., 2017; Ouchi et al., 2018; Fei et al., 2020c,b).

On the other hand, syntactic features are a highly effective SRL performance enhancer, according to numbers of empirical verification in prior works (Marcheggiani et al., 2017; He et al., 2018; Swayamdipta et al., 2018; Zhang et al., 2019), as intuitively SRL shares much underlying structure with syntax. Basically, the syntactic dependent feature is more preferred to be injected into the dependency-based SRL (Roth and Lapata, 2016; Marcheggiani and Titov, 2017; He et al., 2018; Kasai et al., 2019), while other consider the constituent syntax for the span-based SRL (Wang et al., 2019; Marcheggiani and Titov, 2020).

Actually, the constituent and dependency syntax depict the structural features from different angles, while they can share close linguistic relevance. Related works have revealed the mutual benefits on integrating these two heterogeneous syntactic representations for various NLP tasks (Collins, 1997; Charniak, 2000; Charniak and Johnson, 2005; Farkas et al., 2011; Yoshikawa et al., 2017; Zhou and Zhao, 2019; Strzyz et al., 2019; Kato and Matsubara, 2019). Unfortunately, there are very limited explorations for SRL. For example, Li et al. (2010) construct discrete heterogeneous syntactic features for SRL. More recent work in Fei et al. (2020a) leverage knowledge distillation method to inject the heterogeneous syntax representations from various tree encoders into one model for enhancing the span-based SRL. In this work, we consider an explicit integration of these two syntactic structures via two neural solutions. To our knowledge, we are the first attempt performing thorough investigations on the impacts of the heterogeneous syntax combination to the SRL task.

Various neural models have been proposed for encoding the syntactic structures, such as attention mechanism (Strubell et al., 2018; Zhang et al., 2019), TreeLSTM (Li et al., 2018; Xia et al., 2019), GCN (Marcheggiani and Titov, 2017; Li et al., 2018; Marcheggiani and Titov, 2020), etc. In this work, we take the advantages of the TreeLSTM and GCN models for encoding the constituent and dependency trees, as two solutions of our `HeSyFu` encoders. It is worth noticing that prior works using GCN to encode dependency (Marcheggiani and Titov, 2017) and constituent (Marcheggiani and Titov, 2020), where however the syntactic labels are not managed in a unified manner. We thus consider enhancing the syntax GCN by simultaneously modeling the syntactic labels within the structure.

3 SRL Model

3.1 Task Modeling

Following prior works (Tan et al., 2018; Marcheggiani and Titov, 2020), our system aims to identify and classify the arguments of a predicate into semantic roles, such as *A0*, *A1*, *AM-LOC*, etc. We denote the complete role set as $\mathcal{R}$. We adopt the *BIO* tagging scheme. And given a sentence s=$\{w_1$,$\cdots$,$w_n\}$ and a predicate w_p, the model assigns each word w_i a label $\hat{y} \in \mathcal{Y}$,

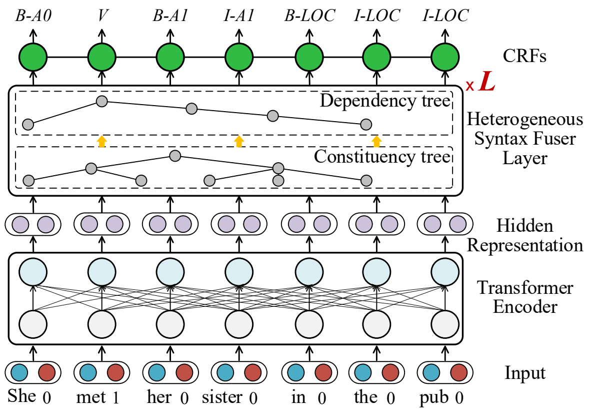


Figure 2: Overview of our SRL framework.

where $\mathcal{Y}$=$(\{B, I\}\times\mathcal{R}) \cup \{O\}$.[2] Note that each semantic argument corresponds to a word span of $\{w_j$,$\cdots$,$w_k\}$ $(1\leq j\leq k\leq n)$.[3]

3.2 Framework

As illustrated in Figure 2, our SRL framework consists of four components, including input representations, Transformer encoder, heterogeneous syntax fuser layer and CRFs decoding layer.

Given an input sentence s and a predicate word w_p (p is the position), the input representations $\boldsymbol{x}_i$ are the concatenation ($\oplus$) of word embeddings $\boldsymbol{x}_{w_i}$ and predicate binary embeddings $\boldsymbol{x}_{(i==p)}$ indicating the presence or absence of w_p:

$$\boldsymbol{x}_i = \boldsymbol{x}_{w_i} \oplus \boldsymbol{x}_{(i==p)} . \tag{1}$$

Afterwards, we adopt Transformer (Vaswani et al., 2017) as our base encoder for yielding contextualized word representations. Transformer (*Trm*) works with multi-head self-attention mechanism:

$$\text{Softmax}(\frac{\boldsymbol{Q} \cdot \boldsymbol{K}^{\mathrm{T}}}{\sqrt{d_k}}) \cdot \boldsymbol{V} , \tag{2}$$

where $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ are the linear projections from the input representation $\boldsymbol{x}_i$. We simplify the flow:

$$\{\boldsymbol{r}_1, \cdots, \boldsymbol{r}_n\} = \text{Trm}(\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}) . \tag{3}$$

Next, based on the hidden representation $\boldsymbol{r}_i$, our heterogeneous syntax fuser (`HeSyFu`) layer, which will be elaborated in Section §4, integrates the constituency and dependency syntax, and yields the syntax-aware hidden representation:

$$\{\boldsymbol{s}_1, \cdots, \boldsymbol{s}_n\} = \text{HeSyFu}(\{\boldsymbol{r}_1, \cdots, \boldsymbol{r}_n\}) . \tag{4}$$

[2]This work focuses on the pipeline-style SRL which is under the assumption that predicates are pre-identified.

[3]When $j = k$, the span-based SRL model shifts into the dependency-based one.

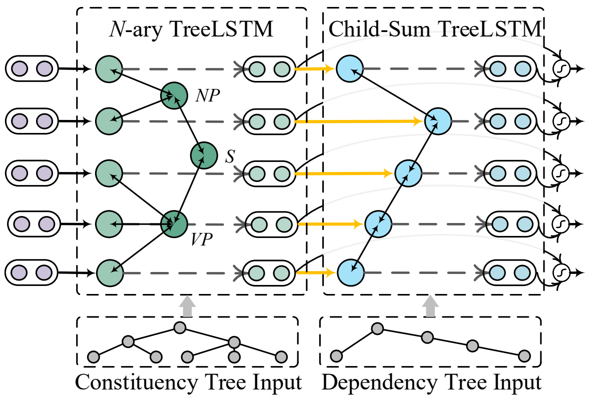


Figure 3: TreeLSTM-based HeSyFu layer.

Based on the syntax-aware hidden representation $\boldsymbol{s}_i$, we use CRFs (Lafferty et al., 2001) to compute the probability of each candidate output $\boldsymbol{y} = \{y_1, \cdots, y_n\}$:

$$p(y|s) = \frac{\exp\{\sum_i (\boldsymbol{W}\boldsymbol{s}_n + \boldsymbol{T}_{y_{i-1},y_i})\}}{Z}, \quad (5)$$

where $\boldsymbol{W}$ and $\boldsymbol{T}$ are the parameters and Z is a normalization factor. The Viterbi algorithm is used to search for the highest-scoring tag sequence $\hat{\boldsymbol{y}}$.

4 Integration of Syntactic Constituency and Dependency Structure

We present two neural heterogeneous syntax fusers (a.k.a., HeSyFu), including a TreeLSTM-based HeSyFu (cf. Figure 3), and a label-aware GCN-based HeSyFu (cf. Figure 4). HeSyFu is stacked with total L layers for a full syntax interaction. We design the architecture with the constituency (denoted as *const.*) encoding in front of the dependency (denoted as *dep.*) encoding, based on the intuition that the boundary recognition helped by *const.* syntax should go before the semantic relation determination aided by *dep.* syntax.

4.1 TreeLSTM Heterogeneous Syntax Fuser

Our TreeLSTM-based HeSyFu (Tr-HeSyFu) is comprised of the N-ary TreeLSTM for *const.* trees and the Child-Sum TreeLSTM for *dep.* trees motivated by Tai et al. (2015).

Constituency tree encoding The flow in TreeLSTM is bidirectional, i.e., bottom-up and top-down, for a full information interaction. For each node u in the tree, we denote the hidden state and memory cell of its v-th ($v \in [1, M]$) branching child as $\boldsymbol{h}^{\uparrow}_{uv}$ and $\boldsymbol{c}_{uv}$. The bottom-up one computes the representation $\boldsymbol{h}^{\uparrow}_u$ from its children hierarchically:

$$\begin{aligned}
\boldsymbol{i}_u &= \sigma(\boldsymbol{W}^{(i)}\boldsymbol{r}_u + \textstyle\sum_{v=1}^{M}\boldsymbol{U}^{(i)}_v\boldsymbol{h}^{\uparrow}_{uv} + \boldsymbol{b}^{(i)}),\\
\boldsymbol{f}_{uk} &= \sigma(\boldsymbol{W}^{(f)}\boldsymbol{r}_u + \textstyle\sum_{v=1}^{M}\boldsymbol{U}^{(f)}_{kv}\boldsymbol{h}^{\uparrow}_{uv} + \boldsymbol{b}^{(f)}),\\
\boldsymbol{o}_u &= \sigma(\boldsymbol{W}^{(o)}\boldsymbol{r}_u + \textstyle\sum_{v=1}^{M}\boldsymbol{U}^{(o)}_v\boldsymbol{h}^{\uparrow}_{uv} + \boldsymbol{b}^{(o)}),\\
\boldsymbol{u}_u &= \mathrm{Tanh}(\boldsymbol{W}^{(u)}\boldsymbol{r}_u + \textstyle\sum_{v=1}^{M}\boldsymbol{U}^{(u)}_v\boldsymbol{h}^{\uparrow}_{uv} + \boldsymbol{b}^{(u)}),\\
\boldsymbol{c}_u &= \boldsymbol{i}_u \odot \boldsymbol{u}_u + \textstyle\sum_{k=1}^{M}\boldsymbol{f}_{uk} \odot \boldsymbol{c}_{uk},\\
\boldsymbol{h}^{\uparrow}_u &= \boldsymbol{o}_u \odot \tanh(\boldsymbol{c}_u),
\end{aligned} \quad (6)$$

where $\boldsymbol{W}$, $\boldsymbol{U}$ and $\boldsymbol{b}$ are parameters. $\boldsymbol{r}_u, \boldsymbol{i}_u, \boldsymbol{o}_u$ and $\boldsymbol{f}_{uv}$ are the input token representation, input gate, output gate and forget gate. Analogously, the top-down N-ary TreeLSTM calculates the representation $\boldsymbol{h}^{\downarrow}_u$ the same way. We concatenate the representations of two directions: $\boldsymbol{h}^{const}_u = \boldsymbol{h}^{\uparrow}_u \oplus \boldsymbol{h}^{\downarrow}_u$. Note that the constituent tree nodes include terminal word nodes and non-terminal constituent nodes, and we only take the representations (i.e., $\boldsymbol{h}^{const}_i$) corresponding to the word node w_i for any usage.

Dependency tree encoding Slightly different from N-ary TreeLSTM for *const.* tree, the non-terminal nodes in *dep.* tree encoded by Child-Sum TreeLSTM are all the word nodes. We also consider the bidirectional calculation here. The bottom-up TreeLSTM obtains $\boldsymbol{h}^{\uparrow}_i$ of the word w_i via:

$$\begin{aligned}
\overline{\boldsymbol{h}}^{\uparrow}_i &= \textstyle\sum_{j\in C(i)}\boldsymbol{h}^{\uparrow}_j,\\
\boldsymbol{i}_i &= \sigma(\boldsymbol{W}^{(i)}\boldsymbol{r}^{'}_i + \boldsymbol{U}^{(i)}\overline{\boldsymbol{h}}^{\uparrow}_i + b^{(i)}),\\
\boldsymbol{f}_{ij} &= \sigma(\boldsymbol{W}^{(f)}\boldsymbol{r}^{'}_i + \boldsymbol{U}^{(f)}\overline{\boldsymbol{h}}^{\uparrow}_j + b^{(f)}),\\
\boldsymbol{o}_i &= \sigma(\boldsymbol{W}^{(o)}\boldsymbol{r}^{'}_i + \boldsymbol{U}^{(o)}\overline{\boldsymbol{h}}^{\uparrow}_i + b^{(o)}),\\
\boldsymbol{u}_i &= \mathrm{Tanh}(\boldsymbol{W}^{(u)}\boldsymbol{r}^{'}_i + \boldsymbol{U}^{(u)}\overline{\boldsymbol{h}}^{\uparrow}_i + b^{(u)}),\\
\boldsymbol{c}_i &= \boldsymbol{i}_i \odot \boldsymbol{u}_i + \textstyle\sum_{j\in C(i)}\boldsymbol{f}_{ij} \odot \boldsymbol{c}_j,\\
\boldsymbol{h}^{\uparrow}_i &= \boldsymbol{o}_i \odot \tanh(\boldsymbol{c}_i),
\end{aligned} \quad (7)$$

where $C(i)$ is the set of child nodes of w_i. $\boldsymbol{r}^{'}_i$ is the input token representation consulting the foregoing constituent output representation: $\boldsymbol{r}^{'}_i = \boldsymbol{r}_i + \boldsymbol{h}^{const}_i$. The top-down one yields $\boldsymbol{h}^{\downarrow}_i$, which is concatenated with the bottom-up one: $\boldsymbol{h}^{dep}_i = \boldsymbol{h}^{\uparrow}_i \oplus \boldsymbol{h}^{\downarrow}_i$.

Integration To fully make use of the heterogeneous syntactic knowledge, we fuse these two resulting syntactic representations. We apply a fusion gate to flexibly coordinate their contributions:

$$\begin{aligned}
\boldsymbol{g}_i &= \sigma(\boldsymbol{W}^{(g_1)}\boldsymbol{h}^{const}_i + \boldsymbol{W}^{(g_2)}\boldsymbol{h}^{dep}_i + b^{(g)}),\\
\boldsymbol{s}_i &= \boldsymbol{g}_i \odot \boldsymbol{h}^{const}_i + (1 - \boldsymbol{g}_i) \odot \boldsymbol{h}^{dep}_i.
\end{aligned} \quad (8)$$

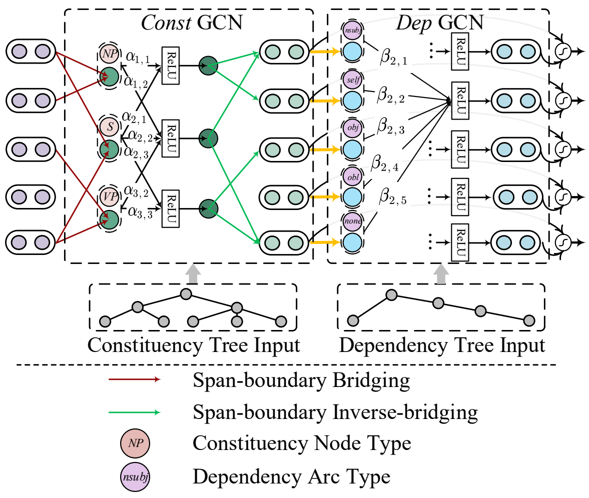


Figure 4: Label-aware GCN-based HeSyFu layer.

4.2 Label-aware GCN-based Heterogeneous Syntax Fuser

Compared with TreeLSTM, GCN is more computationally efficient on performing the structural propagation among nodes, i.e., with O(1) complexity. On the other hand, it is also crucial to leverage the syntactic labels (i.e., dependent arc types, and constituent phrasal types) into the SRL learning. For example, within the dependency tree, the information from the neighboring nodes under distinct types of arcs can contribute in different degrees. However we note that current popular syntax GCNs (Marcheggiani and Titov, 2017, 2020) do not encode the dependent or constituent labels with the nodes in a unified manner, which could be inaccurate to describe the syntactic connecting attributes between the neighbor nodes. Based on their syntax GCNs, we newly propose label-aware constituency and dependency GCNs which are able to explicitly formalize the structure edges with syntactic labels simultaneously, and normalize them unitedly.[4] As illustrated in Figure 4, our label-aware GCN-based HeSyFu (denoted as LG-HeSyFu) has a similar assembling architecture to TreeLSTM-based HeSyFu, and will finally be navigated via the gate mechanism as in Eq. (8).

Constituency tree encoding The constituent tree is modeled as a graph $G^{(c)}$=$(U^{(c)}, E^{(c)})$, where $U^{(c)}$ is the node set and $E^{(c)}$ is the edge set. We denote $e_{uv}^{(c)}$=1 if there is an edge between node u and node v, and e_{uv}=0 vice versa. We enable the edges to be bidirectional. μ_u represents the constituent label of node u, such as *S*, *NP* and *VP*, etc. We take the vectorial embedding $\boldsymbol{v}_u^{(c)}$ for the node label μ_u. Our constituent GCN (denoted as *Const* GCN) yields the node representations $\boldsymbol{h}_u^{(c)}$:

$$\boldsymbol{h}_u^{(c)} = \text{ReLU}\{\textstyle\sum_{v=1}^{M} \alpha_{uv}(\boldsymbol{W}^{(c_1)} \cdot \boldsymbol{r}_v^b + \boldsymbol{W}^{(c_2)} \cdot \boldsymbol{v}_v^{(c)} + b^{(c)})\}\,, \tag{9}$$

where $\boldsymbol{r}_v^b$ is the initial node representation of the node v via **span-boundary bridging operation**, i.e., adding the start and end token representation of the phrasal span, $\boldsymbol{r}_v^b = \boldsymbol{r}_{start} + \boldsymbol{r}_{end}$. And α_{uv} is the constituent connecting distribution:

$$\alpha_{uv} = \frac{e_{uv}^{(c)} \cdot \exp\{(\boldsymbol{z}_u^{(c)})^T \cdot \boldsymbol{z}_v^{(c)}\}}{\sum_{v'=1}^{M} e_{uv'}^{(c)} \cdot \exp\{(\boldsymbol{z}_u^{(c)})^T \cdot \boldsymbol{z}_{v'}^{(c)}\}}\,, \tag{10}$$

where $\boldsymbol{z}_u^{(c)} = \boldsymbol{v}_u + \boldsymbol{v}_u^{(c)}$. This distribution α_{uv} encodes both the syntactic edge and label information, and thus comprehensively reflects the connecting strengths between neighbors. We then perform **span-boundary inverse-bridging** to restore the token node representation $\boldsymbol{h}_i^{const}$ for each word w_i, i.e., $\boldsymbol{h}_i^{const} = \boldsymbol{h}_{u'}^{(c)} + \boldsymbol{h}_{v'}^{(c)}$.

Dependency tree encoding Likewise, the dependent tree is modeled as a graph $G^{(d)}$=$(U^{(d)}, E^{(d)})$. $e_{ij}^{(d)}$=1/0 denotes the dependency arc existence. $\pi_{ij}^{\leftrightarrow}$ represents the edge label between w_i and w_j, which is also bidirectional. Besides of the pre-defined dependency labels, we additionally add a '*self*' label as the self-loop edge $\pi_{ii}^{\leftrightarrow}$, and a '*none*' label representing no edge between w_i and w_j. We use the embedding form $\boldsymbol{v}_{ij}^{(d)}$ for $\pi_{ij}^{\leftrightarrow}$. The update in dependent GCN (denoted as *Dep* GCN) is written as:

$$\boldsymbol{h}_i^{(d)} = \text{ReLU}(\textstyle\sum_{j=1}^{n} \beta_{ij}(\boldsymbol{W}^{(d_1)} \cdot \boldsymbol{r}_j^{'} + \boldsymbol{W}^{(d_2)} \cdot \boldsymbol{v}_{ij}^{(d)} + b^{(d)}))\,, \tag{11}$$

where $\boldsymbol{r}_j^{'} = \boldsymbol{r}_j + \boldsymbol{h}_i^{const}$. β_{ij} is the neighbor connecting-strength distribution:

$$\beta_{ij} = \frac{e_{ij}^{(d)} \cdot \exp\{(\boldsymbol{z}_i^{(d)})^T \cdot \boldsymbol{z}_j^{(d)}\}}{\sum_{j'=1}^{n} e_{ij'}^{(d)} \cdot \exp\{(\boldsymbol{z}_i^{(d)})^T \cdot \boldsymbol{z}_{j'}^{(d)}\}}\,, \tag{12}$$

where $\boldsymbol{z}_i^{(d)} = \boldsymbol{r}_i^{'} + \boldsymbol{v}_{ij}^{(d)}$. Here $\boldsymbol{h}_i^{(d)}$ also can be denoted as $\boldsymbol{h}_i^{dep}$, which navigates the dependent arc and label information in a more unified way.

5 Experiments

5.1 Setups

We conduct experiments on the span-based SRL datasets (CoNLL05 & CoNLL12), and

[4]We note that the constituent labels are attached onto nodes, while dependent labels are attached onto edges.

	CoNLL05 WSJ			CoNLL05 Brown			CoNLL12 OntoNotes		
	P	R	F1	P	R	F1	P	R	F1
Without Syntax									
He et al. (2017)	85.00	84.30	84.60	74.90	72.40	73.60	83.50	83.30	83.40
Tan et al. (2018)	84.50	85.20	84.80	73.50	74.60	74.10	81.90	83.60	82.70
Li et al. (2020a)+RoBERTa	88.05	88.00	88.03	80.04	79.56	79.80	86.40	86.83	86.61
Trm+RoBERTa†	87.41	87.72	87.60	79.78	79.86	79.82	86.28	86.67	86.40
With Dependency Syntax									
Strubell et al. (2018)	84.70	84.24	84.47	73.89	72.39	73.13	83.30	81.38	82.33
Xia et al. (2020)	85.12	85.00	85.06	76.30	75.42	75.86	-	-	-
Child-Sum TreeLSTM†	84.94	85.80	85.40	74.60	74.10	74.36	83.42	83.56	83.47
Dep GCN†	86.03	86.52	86.22	75.38	75.89	75.62	84.32	84.88	84.61
Dep GCN+RoBERTa†	88.21	87.82	88.07	80.73	79.82	80.13	86.58	86.99	86.82
With Constituency Syntax									
Wang et al. (2019)*	85.40	85.02	85.23	75.48	75.23	75.36	84.35	84.11	84.21
Marcheggiani and Titov (2020)	85.80	85.10	85.40	76.20	74.70	75.50	84.50	84.30	84.40
Marcheggiani and Titov (2020)+RoBERTa	87.70	88.10	87.90	80.50	80.70	80.60	86.50	87.10	86.80
N-ary TreeLSTM†	85.91	85.27	85.58	75.22	75.06	75.12	84.12	83.85	84.02
Const GCN†	86.68	86.38	86.52	76.54	76.21	76.36	85.51	84.96	85.25
Const GCN+RoBERTa†	88.71	88.94	88.81	81.52	81.05	81.27	87.33	87.42	87.35
With Dependency & Constituency Syntax									
Fei et al. (2020a)*	86.82	86.50	86.72	76.67	76.35	76.48	85.86	85.30	85.50
Tr-HeSyFu†	86.27	86.52	86.64	76.95	76.50	76.87	85.91	85.48	85.66
LG-HeSyFu†	87.16	87.63	87.32	78.72	77.35	78.12	86.51	85.92	86.20
LG-HeSyFu w/o Syn.Label†	86.93	87.21	86.98	77.61	76.85	77.48	85.93	85.68	85.79
LG-HeSyFu+RoBERTa†	**88.86**	**89.28**	**89.04**	**83.52**	**83.75**	**83.67**	**88.09**	**88.83**	**88.59**

Table 1: Results on span-based SRL datasets. Values with $*$ are from our re-implementations, while others are retrieved from the raw papers. Scores with $\dagger$ are presented after significant test ($p \leq 0.05$).

dependency-based SRL dataset (CoNLL09). Each dataset has its own training, development, and test sets. We convert the constituency syntax annotations in CoNLL05&12 into dependency annotations by following the standard of Stanford Typed Dependency (v3.3.0).[5] We obtain the constituency annotations for CoNLL09 from the PTB data. We adopt the CoNLL05 evaluation scripts[6] to evaluate the performances, with precision (P), recall (R) and F1 score as the metrics. We conduct significance tests via Dan Bikel's evaluation comparer.[7] The Transformer hidden size is 768. The hidden sizes in TreeLSTM and GCN encoders are in [250,300,350]. We adopt the Adam optimizer with an initial learning rate of 2e-5. We train the model[8] by mini-batch size in [16,24,32] with early-stop strategy. We also load the pre-trained parameters[9] from the RoBERTa language model (Liu et al., 2019) to our Transformer encoder for boosting the performance. The environment is with Intel i9 CPU and NVIDIA RTX 3090Ti GPU.

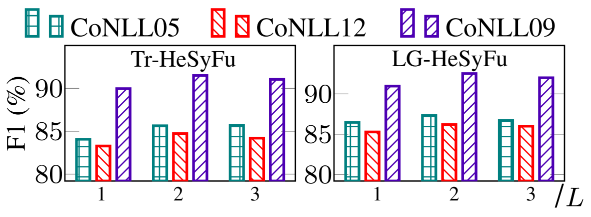


Figure 5: HeSyFu with different layers.

5.2 Development Experiments

We first perform preliminary experiments based on the development sets.

Layer of syntax encoder From Figure 5 we see that either too larger or fewer layers of HeSyFu does no benefits to the overall performances. When L=2 for Tr-/LG-HeSyFu, the performances be-

[5] https://nlp.stanford.edu/software/lex-parser.html

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

[7] http://www.cis.upenn.edu/~dbikel/software.html#comparator

[8] Codes: https://github.com/scofield7419/HeSyFu

[9] https://github.com/pytorch/fairseq/tree/master/examples/RoBERTa

	CoNLL05		CoNLL12		CoNLL09	
	F1	Δ	F1	Δ	F1	Δ
Tr-HeSyFu						
Dep.→Const.	84.5	-1.2	83.3	-1.0	91.2	-0.3
Const.→Dep.	85.7		84.3		91.5	
LG-HeSyFu						
Dep.→Const.	85.3	-1.4	84.3	-0.8	92.1	-0.4
Const.→Dep.	86.7		85.1		92.5	

Table 2: Influences of the syntax encoding order.

come universally the best.

Order of the heterogeneous syntax encoding We design the architecture with constituency encoding before dependency encoding, as described earlier. If we exchange this encoding order, we see from Table 2 that the drops come out. Also the drops are more severe on the span-based SRL data. This verifies the correctness of our model design.

5.3 Main Results

Our aim is to answer the research questions as listed in Section §1, based on the main experimental results in Table 1 and Table 3. **[★ Answer to Q1]** Our first observation is that leveraging syntax knowledge, e.g. either the dependency or constituency, benefits both the span-based and dependency-based SRL, while the integration of two heterogeneous syntax contributes the most, more than any one of the standalone syntax.

However we see that the improvements from this syntax integration is slightly different between span-based and dependency-based SRL. **[★ Answer to Q2]** In particular, the improvements for span-based SRL are more notable than dependency-based SRL, which can be learned by the comparisons between 'Trm+RoBERTa' and 'LG-HeSyFu+RoBERTa' on two tables. Our conjecture is that the the constituent structure knowledge will additionally help the span boundary detection of span-based SRL, compared with dependency-based SRL. Also we find that using only constituency syntax contributes more span-based SRL, while the dependency-based SRL benefits more from dependency syntax.

Looking into the specific results, within the scope of heterogeneous syntax integration methods, our systems (both Tr-HeSyFu and LG-HeSyFu) outperform Fei et al. (2020a), demonstrating the advances of our heterogeneous syntax integrating methods. Overall, our LG-HeSyFu model wins the new state-of-the-art performances on the used datasets, and with the help of the RoBERTa language model, the superiority is still maintained.

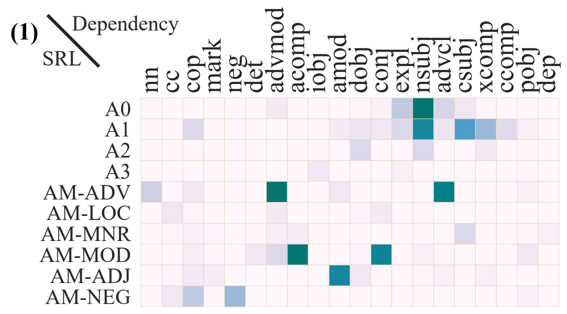
[★ Answer to Q3] Also we show that our LG-HeSyFu based system consistently outperforms Tr-HeSyFu based one. Even LG-HeSyFu without using the syntax label features can still keep better. It is also clear that the GCN based encoders show consistently higher scores than the TreeLSTM based ones, verifying the effectiveness of leveraging GCN encoding syntax (Marcheggiani and Titov, 2017; Li et al., 2018). **[★ Answer to Q4]** Meanwhile, the ablation of syntax label information reveals the importance of its leverage for the SRL learning.

	P	R	F1
Without Syntax			
He et al. (2018)	89.50	87.90	88.70
Li et al. (2020b)	-	-	90.26
Trm+RoBERTa†	91.34	91.12	91.25
With Dependency Syntax			
Li et al. (2018)	90.30	89.30	89.80
He et al. (2019)	89.96	89.96	89.96
Child-Sum TreeLSTM†	90.67	90.60	90.63
Dep GCN†	90.98	90.85	90.91
Dep GCN+RoBERTa†	92.45	92.05	92.23
With Constituency Syntax			
N-ary TreeLSTM†	89.56	89.21	89.42
Const GCN†	90.48	90.19	90.35
Const GCN+RoBERTa†	91.33	91.87	91.65
With Dependency & Constituency Syntax			
Fei et al. (2020a)*	90.78	90.92	90.88
Tr-HeSyFu†	91.02	91.22	91.10
LG-HeSyFu†	92.24	92.53	92.45
LG-HeSyFu w/o Syn. Label†	91.85	92.15	92.05
LG-HeSyFu+RoBERTa†	**92.89**	**92.80**	**92.83**

Table 3: Results on dependency-based SRL CoNLL09 dataset.

5.4 Analysis and Discussion

Correlations between SRL and syntax structures We explore the correlations between the SRL structure and the two syntax structures. We reach this by analyzing the SRL prediction with the neighbor connecting weights, i.e., α_{uv} of *Const* GCN and β_{ij} of *Dep* GCN. We visualize the results (on CoNLL05) in Figure 6. **[★ Answer to Q5]** We learn that our framework indeed has captured the underlying inter-dependency between the SRL structures and the syntactic structure from the diversified visualizations. By accurately modeling



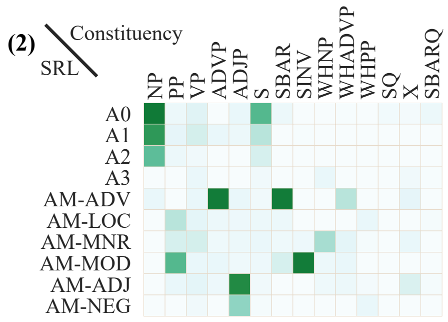


Figure 6: Discovered correlations of (1) SRL vs. dependent structure, (2) SRL vs. constituent structure.

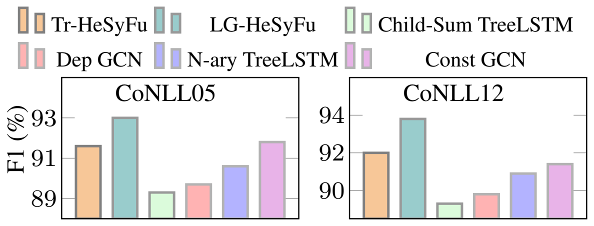


Figure 7: F1 scores for span boundary detection.

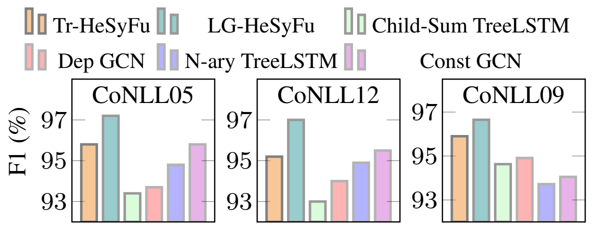


Figure 8: Results on the argument role label prediction.

such correlations, our LG-`HeSyFu` system naturally yields prominent meanwhile explainable SRL performances. Also some interesting patterns can be observed. Actually, not all the syntactic elements contribute the SRL learning. For example, the semantic roles `A0`, `A1` and `A2` relates more to the dependent edge *nsubj* and *csubj*, and more to the constituent phrase *NP*. We believe this can lay a crucial foundation for the direction of unsupervised semantic role labeling that relies on the syntactic structures.

Span boundary detection We now investigate the influences of the heterogeneous syntax integration to the span boundary match[10] on span-based SRL, i.e., on CoNLL05/12 data. From Figure 7 we learn that the heterogeneous syntax integration can improve the boundary detection over any standalone syntax leverage, while actually the constituency syntax contributes more significantly than dependency feature. And our LG-`HeSyFu` shows the best helpfulness than Tr-`HeSyFu`.

Label prediction We next evaluate the role label prediction. We only measure the correctly extracted arguments on whether its label further matches the gold annotation. We show the F1 score in Figure 8. Similar to the span boundary identification, the heterogeneous syntax integration can contribute the most than that with any single syntax usage. Interestingly, the standalone dependency syntax shows more improvements on the dependency-based SRL, while the phrasal constituency features benefit more the span-based SRL.

Error breakdown To analyze which error types different syntax-aided SRL models tend to make, we follow prior works (He et al., 2017; Strubell et al., 2018), manually fixing the errors by applying oracle transformations incrementally based on CoNLL05.[11] The analysis is shown in Figure 9. Specifically, constituency syntax methods perform better than dependency-aided methods, w.r.t. the span boundary errors (‘Merge Spans’, ‘Split Spans’ and ‘Fix Span Boundary’). Most importantly, it is quite clear that our heterogeneous syntax integrated systems (Tr-`HeSyFu` and LG-`HeSyFu`) makes fewer errors than baseline standalone syntax-aware methods, demonstrating the necessity to combine both two types of syntax.

Syntax distribution By observing the gate values $\boldsymbol{g}_i$ (in Eq. 8) we can analyze the distributions of dependency and constituency features required by span-based and dependency-based SRL. From Figure 10 we see that span-SRL relies more on constituency feature, while dependency-SRL needs more dependency-aware feature. Such finding quite coincides with the foregoing quantitative analysis, as well as our intuition.

[10]A correct match means both the start and end boundary of an argument span is correct, regardless of its label.

[11]The bigger the correction error improves, the more the model makes on it.

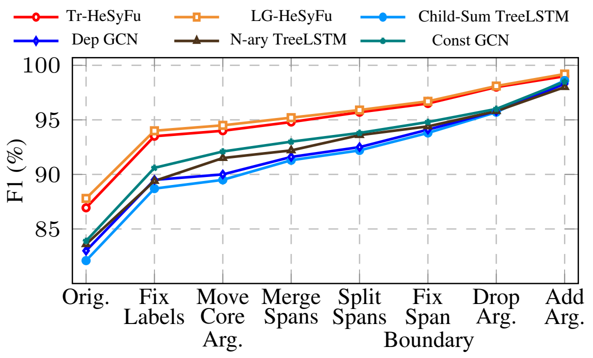


Figure 9: Performances after error corrections.

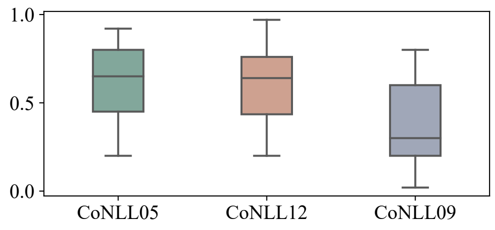


Figure 10: Heterogeneous syntax distribution. Level ≥0.5 means more reliance upon constituent syntax, otherwise for dependency

6 Conclusion and Future Work

We investigated the integration of constituency and dependency syntax for the SRL task. We first introduced TreeLSTM-based heterogeneous syntax fusing encoders, and further proposed innovative label-aware syntax GCN encoders for the integration. Experimental results showed that combining the heterogeneous syntax brought better results on both span-based and dependency-based SRL, than any one standalone syntax knowledge. As future work, we investigate other kinds of structural knowledge integration besides syntax, such as Semantic Dependency Structure, Abstract Meaning Representation (AMR), and explore the possibility of extending our model to incorporating such structured information. Besides, integrating the heterogeneous syntax knowledge into pre-training language models will be a promising direction.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61772378), the National Key Research and Development Program of China (No. 2017YFC1200500), the Research Foundation of Ministry of Education of China (No. 18JZD015).

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the COLING*, pages 86–90.

Emanuele Bastianelli, Giuseppe Castellucci, Danilo Croce, and Roberto Basili. 2013. Textual inference and meaning representation in human robot interaction. In *Proceedings of the Joint Symposium on Semantic Processing*, pages 65–69.

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

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the NAACL*, pages 132–139.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the ACL*, pages 173–180.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the ACL*, pages 16–23.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the EMNLP*, pages 1535–1545.

Richárd Farkas, Bernd Bohnet, and Helmut Schmid. 2011. Features for phrase-structure reranking from dependency parses. In *Proceedings of the ICPT*, pages 209–214.

Hao Fei, Fei Li, Bobo Li, and Donghong Ji. 2021a. Encoder-decoder based unified semantic role labeling with label-aware syntax. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1479–1488.

Hao Fei, Yafeng Ren, and Donghong Ji. 2020a. Mimic and conquer: Heterogeneous tree structure distillation for syntactic NLP. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 183–193.

Hao Fei, Meishan Zhang, and Donghong Ji. 2020b. Cross-lingual semantic role labeling with high-quality translated training corpus. In *Proceedings of the ACL*, pages 7014–7026.

Hao Fei, Meishan Zhang, Bobo Li, and Donghong Ji. 2021b. End-to-end semantic role labeling with neural transition-based model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 566–575.

Hao Fei, Meishan Zhang, Fei Li, and Donghong Ji. 2020c. Cross-lingual semantic role labeling with model transfer. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2427–2437.

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

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

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 the CoNLL*, pages 1–18.

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

Shexia He, Zuchao Li, and Hai Zhao. 2019. Syntax-aware multilingual semantic role labeling. In *Proceedings of the EMNLP*, pages 5350–5359.

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

Jungo Kasai, Dan Friedman, Robert Frank, Dragomir Radev, and Owen Rambow. 2019. Syntax-aware neural semantic role labeling with supertags. In *Proceedings of the NAACL*, pages 701–709.

Yoshihide Kato and Shigeki Matsubara. 2019. PTB graph parsing with tree approximation. In *Proceedings of the ACL*, pages 5344–5349.

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

Shiqi Li, Qin Lu, Tiejun Zhao, Pengyuan Liu, and Hanjing Li. 2010. Combining constituent and dependency syntactic views for Chinese semantic role labeling. In *Proceedings of the COLING*, pages 665–673.

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

Zuchao Li, Shexia He, Jiaxun Cai, Zhuosheng Zhang, Hai Zhao, Gongshen Liu, Linlin Li, and Luo Si. 2018. A unified syntax-aware framework for semantic role labeling. In *Proceedings of the EMNLP*, pages 2401–2411.

Zuchao Li, Hai Zhao, Rui Wang, and Kevin Parnow. 2020b. High-order semantic role labeling. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1134–1151.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Umar Maqsud, Sebastian Arnold, Michael Hülfenhaus, and Alan Akbik. 2014. Nerdle: Topic-specific question answering using wikia seeds. In *Proceedings of the COLING*, pages 81–85.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the CoNLL*, pages 411–420.

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

Diego Marcheggiani and Ivan Titov. 2020. Graph convolutions over constituent trees for syntax-aware semantic role labeling. In *Proceedings of the EMNLP*, pages 3915–3928.

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

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

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

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of the ACL*, pages 581–588.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

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

Chen Shi, Shujie Liu, Shuo Ren, Shi Feng, Mu Li, Ming Zhou, Xu Sun, and Houfeng Wang. 2016. Knowledge-based semantic embedding for machine translation. In *Proceedings of the ACL*, pages 2245–2254.

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

Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. 2019. Sequence labeling parsing by learning across representations. In *Proceedings of the ACL*, pages 5350–5357.

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

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

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the ACL*, pages 1556–1566.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the NeurIPS*, pages 5998–6008.

Yufei Wang, Mark Johnson, Stephen Wan, Yifang Sun, and Wei Wang. 2019. How to best use syntax in semantic role labelling. In *Proceedings of the ACL*, pages 5338–5343.

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

Qingrong Xia, Rui Wang, Zhenghua Li, Yue Zhang, and Min Zhang. 2020. Semantic role labeling with heterogeneous syntactic knowledge. In *Proceedings of the COLING*, pages 2979–2990.

Deyi Xiong, Min Zhang, and Haizhou Li. 2012. Modeling the translation of predicate-argument structure for SMT. In *Proceedings of the ACL*, pages 902–911.

Kun Xu, Haochen Tan, Linfeng Song, Han Wu, Haisong Zhang, Linqi Song, and Dong Yu. 2020. Semantic Role Labeling Guided Multi-turn Dialogue ReWriter. In *Proceedings of the EMNLP*, pages 6632–6639.

Masashi Yoshikawa, Hiroshi Noji, and Yuji Matsumoto. 2017. A* CCG parsing with a supertag and dependency factored model. In *Proceedings of the ACL*, pages 277–287.

Yue Zhang, Rui Wang, and Luo Si. 2019. Syntax-enhanced self-attention-based semantic role labeling. In *Proceedings of the EMNLP*, pages 616–626.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009a. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the CoNLL*, pages 61–66.

Hai Zhao, Wenliang Chen, and Chunyu Kit. 2009b. Semantic dependency parsing of NomBank and PropBank: An efficient integrated approach via a large-scale feature selection. In *Proceedings of the EMNLP*, pages 30–39.

Junru Zhou and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on Penn treebank. In *Proceedings of the ACL*, pages 2396–2408.