Syntax Role for Neural Semantic Role Labeling

Zuchao Li
Shanghai Jiao Tong University
Department of Computer Science and Engineering
charlee@sjtu.edu.cn

Hai Zhao*
Shanghai Jiao Tong University
Department of Computer Science and Engineering
zhaohai@cs.sjtu.edu.cn

Shexia He
Shanghai Jiao Tong University
Department of Computer Science and Engineering
heshexia@sjtu.edu.cn

Jiaxun Cai
Shanghai Jiao Tong University
Department of Computer Science and Engineering
caijiaxun@sjtu.edu.cn

Semantic role labeling (SRL) is dedicated to recognizing the semantic predicate-argument structure of a sentence. Previous studies in terms of traditional models have shown syntactic information can make remarkable contributions to SRL performance; however, the necessity of syntactic information was challenged by a few recent neural SRL studies that demonstrate impressive performance without syntactic backbones and suggest that syntax information becomes much less important for neural semantic role labeling, especially when paired with recent deep neural network and large-scale pre-trained language models. Despite this notion, the neural SRL field still lacks a systematic and full investigation on the relevance of syntactic information in SRL, for

* Corresponding author. This work was supported by the National Key Research and Development Program of China (No. 2017YFB0304100) and the Key Projects of National Natural Science Foundation of China (U1836222 and 61733011).

Submission received: 11 September 2020; revised version received: 11 February 2021; accepted for publication: 5 June 2021.

https://doi.org/10.1162/COLI_a_00408

both dependency and both monolingual and multilingual settings. This paper intends to quantify the importance of syntactic information for neural SRL in the deep learning framework. We introduce three typical SRL frameworks (baselines), sequence-based, tree-based, and graph-based, which are accompanied by two categories of exploiting syntactic information: syntax pruning-based and syntax feature-based. Experiments are conducted on the CoNLL-2005, -2009, and -2012 benchmarks for all languages available, and results show that neural SRL models can still benefit from syntactic information under certain conditions. Furthermore, we show the quantitative significance of syntax to neural SRL models together with a thorough empirical survey using existing models.

1. Introduction

Semantic role labeling (SRL), namely, semantic parsing, is a shallow semantic parsing task that aims to recognize the predicate-argument structure of each predicate in a sentence, such as *who* did *what* to *whom*, *where* and *when*, and so forth. Specifically, SRL seeks to identify arguments and label their semantic roles given a predicate. SRL is an important method for obtaining semantic information that is beneficial to a wide range of natural language processing (NLP) tasks, including machine translation (Shi et al. 2016), question answering (Berant et al. 2013; Yih et al. 2016), discourse relation sense classification (Mihaylov and Frank 2016), and relation extraction (Lin, Liu, and Sun 2017).

SRL can be split into four subtasks: predicate detection, predicate disambiguation, argument identification, and argument classification. For argument annotation, there are two formulations (styles). One is based on constituents (i.e., phrase or span), and the other is based on dependencies. The latter, proposed by the CoNLL-2008 shared task (Surdeanu et al. 2008), is also called semantic dependency parsing and annotates the heads of arguments rather than phrasal arguments. Figure 1 shows example annotations.

In prior SRL work, considerable attention has been paid to feature engineering, which struggles to capture sufficient discriminative information compared to neural network models, which are capable of extracting features automatically. In particular, syntactic information, including syntactic tree features, has been known to be extremely beneficial to SRL since the large scale of empirical verification of Punyakanok, Roth, and Yih (2008). Despite their success, their work suffered from erroneous syntactic input, leading to an unsatisfactory performance.

To alleviate these issues, Marcheggiani, Frolov, and Titov (2017) and He et al. (2017) proposed a simple but effective neural model for SRL without syntactic input. Their work suggested that neural SRL does not have to rely on syntactic features, contradicting the belief that syntax is a necessary prerequisite for SRL, which was believed as early as Gildea and Palmer (2002). This dramatic contradiction motivated us to make a thorough exploration on syntactic contribution to SRL.

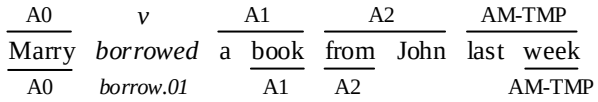


Figure 1
Examples of annotations in span (above) and dependency (below) SRL.

Table 1
A chronicle of related work for span and dependency SRL. SA represents a syntax-aware system (no + indicates a syntax-agnostic system). F_1 is the result of a single model on the official test set.

Span (CoNLL-2005)					Dependency (CoNLL-2009)				
Time	System	SA	Method	$\mathbf{F_1}$	**Time**	System	SA	Method	$\boldsymbol{F_1}$
2008	Punyakanok et al.	+	ILP	**76.3**	**2009**	Zhao et al.	+	ME	***86.2***
2008	Toutanova et al.	+	DP	**79.7**	**2010**	Björkelund et al.	+	global	***86.9***
2015	**FitzGerald et al.**	+	structured	**79.4**			+	structured	***87.3***
2015	Zhou and Xu		deep BiLSTM	**82.8**					
					2016	Roth and Lapata	+	PathLSTM	***87.7***
2017	He et al.		highway BiLSTM	**83.1**	**2017**	Marcheggiani et al.		BiLSTM	***87.7***
					2017	Marcheggiani and Titov	+	GCNs	***88.0***
2018	Tan et al.		self-attention	**84.8**	**2018**	He et al. (b)	+	ELMo	***89.5***
2018	Strubell et al.	+	self-attention	**83.9**	**2018**	Cai et al.		biaffine	***89.6***
2018a	He et al. (a)		ELMo	**87.4**	**2018**	Li et al. (a)	+	ELMo	***89.8***
2019b	Li et al. (b) AAAI		ELMo+biaffine	**87.7**				ELMo+biaffine	***90.4***

As shown in Table 1, span and dependency are effective formal representations for semantics, though it has been unknown for a long time which form, span, or dependency would be better for the convenience and effectiveness of semantic machine learning and later applications. This topic has been roughly discussed in Johansson and Nugues (2008a) and Li et al. (2019a), who both concluded that the (best) dependency SRL system at that time clearly outperformed the span-based (best) system through gold syntactic structure transformation; however, due to the different requirements of downstream task applications, span and dependency both remain focuses of research. Additionally, the two forms of SRL may benefit from each other's joint (rather than separated) development. We, therefore, revisit the role of syntax in SRL on a more solid empirical basis and investigate the role of syntax[1] for the two SRL styles by supplying syntax knowledge of varying quality.

Recent work on syntax contributions has been limited to individual models and the ways in which syntax has been utilized. The conclusions drawn for syntax roles therefore have some limitations. In order to reduce these limitations, we explored three typical and strong baseline models and two categories of syntactic utilization methods. In addition, pre-trained language models, such as ELMo (Peters et al. 2018) and BERT (Devlin et al. 2019), that build contextualized representations, continue to provide gains on NLP benchmarks, and Hewitt and Manning (2019) showed that structure of syntax information emerges in the deep models' word representation spaces. Whether neural SRL models can further benefit from explicit syntax information in addition to this implicit syntax information, however, is another issue we consider.

Besides, most of the SRL literature is dedicated to impressive performance gains on English, while other multiple languages receive relatively little attention. Although human languages have some basic commonalities in syntactic structure and even different levels of grammar, their differences are also very obvious. The study of syntactic roles needs to be examined in the context of multiple languages for verifying its effectiveness and applicability.

1 It is worth noting that the syntax studied in this paper is limited to syntactic knowledge in the narrow sense by using syntactic relationships (via dependency and constituency trees) but does not include lemma, part-of-speech, etc.

In order to quantitatively evaluate the contribution of syntax to SRL, we adopt the ratios between labeled F_1 score for semantic dependencies (Sem-F_1), the labeled attachment score (LAS) for syntactic dependencies, and the F_1 score for syntactic constituents. This ratio was first introduced by CoNLL-2008 (Surdeanu et al. 2008) shared task as an evaluation metric. Because different syntactic parsers contribute different syntactic inputs with varying levels of quality, different syntactically driven SRL systems are based on different syntactic foundations. Therefore, our proposed ratio offers a fairer comparison between different syntactically driven SRL systems, which our empirical study surveys.

2. Background

SRL was first pioneered by Gildea and Jurafsky (2000) based on the FrameNet semantic labeling project (Baker, Fillmore, and Lowe 1998). PropBank (Palmer, Gildea, and Kingsbury 2005) is one of the most commonly used labeling schemes for this task. This involves two variants: span-based labeling (span SRL), where arguments are characterized as word spans (Carreras and Màrquez 2005; Pradhan et al. 2012), and head-based labeling (dependency SRL), which only labels head words and relies on syntactic parse trees (Hajič et al. 2009).

Conventionally, when identifying predicates, span SRL decomposes to two subtasks: argument identification and argument classification. The former identifies the arguments of a predicate, and the latter assigns them semantic role labels, determining the relations between arguments and predicates. PropBank defines a set of semantic roles for labeling arguments. These roles fall into two categories: *core* and *non-core* roles. The core roles (A0-A5 and AA) indicate different semantics in predicate-argument structure, while the non-core roles are modifiers (AM-*adj*), where *adj* specifies the adjunct type, such as in temporal (AM-TMP) and locative (AM-LOC) adjuncts. For the example shown in Figure 1, A0 is a proto-agent, representing the *borrower*.

Slightly different from span SRL in argument annotation, dependency SRL labels the head words[2] of arguments rather than of entire phrases, a practice popularized by the CoNLL-2008 and CoNLL-2009 shared tasks[3] (Surdeanu et al. 2008; Hajič et al. 2009). Furthermore, when no predicate is given, two other indispensable subtasks of dependency SRL are required: predicate identification and predicate disambiguation. The former identifies all predicates in a sentence; and the latter determines the word senses, the specific contextual meanings, of predicates. In the example shown in Figure 1, *01* indicates the first sense from the PropBank sense repository for predicate *borrowed* in the sentence.

Johansson and Nugues (2008c) demonstrated that in conventional SRL models, syntactic trees provide a good form of representation for the assigning of semantic role labels. The successful application of neural networks to SRL (Zhou and Xu 2015; He et al. 2017; Marcheggiani, Frolov, and Titov 2017; Cai et al. 2018) mitigated conventional SRL models' need for comprehensive feature engineering based on syntax trees (Zhao et al. 2009a) and resulted in syntax-agnostic neural SRL models that achieved compet-

2 The head word for a span serves as a dependency relation's modifier for words outside the span and a dependency head for words inside the span. This is different from syntactic heads in head-dependent relationships.

3 CoNLL-2008 is an English-only task, while CoNLL-2009 extends to a multilingual one. Their main difference is that predicates have been indicated beforehand for the latter. Or rather, CoNLL-2009 does not need predicate identification, but it is an indispensable subtask for CoNLL-2008.

itive performance. Recent work has built on this and explored the inclusion of syntax in neural SRL. Including syntax in SRL has three main benefits that have been common motivations for recent work:

- Arguments are often dispersed around the predicates in syntax trees (Xue and Palmer 2004; Zhao and Kit 2008; He et al. 2018b; He, Li, and Zhao 2019).
- Some predicate-argument arcs in semantic dependency graphs are mirrored by head-dependent arcs in their corresponding dependency parse trees, and there is a deterministic mapping between these syntactic relationships and semantic role labels (Surdeanu et al. 2008; Lang and Lapata 2010; Marcheggiani and Titov 2017; Li et al. 2018; Cai and Lapata 2019b; Marcheggiani and Titov 2020).
- Syntax parse trees can strengthen language representations (Johansson and Nugues 2008c; Strubell et al. 2018; Kasai et al. 2019).

In this paper, since the third benefit is a general improvement for downstream tasks and not limited to SRL, we explore the exploitation of the first two benefits for use in neural SRL.

3. Methodology

To fully disclose the predicate-argument structure, typical SRL systems have to perform four subtasks step-by-step or jointly learn and predict the four targets. In order to research the role of syntax, we evaluate our systems in two separate settings: being given the predicate and not being given the predicate. For the first setting, our backbone models all only focus on the identification and labeling of arguments. We use the pre-identified predicate information when the predicate is provided in the corpus and adopt a sequence tagging model to perform predicate disambiguation. In the second condition, we do the work of predicate identification and disambiguation in one sequence tagging model. In summary, we focus on three backbone models for argument identification and disambiguation and feed the predicates into the models as features.

3.1 Factorization and Modeling

We summarize and present three typical baseline models, which are based on the strategies of factoring and modeling of semantic graphs in the SRL: sequence-based, tree-based, and graph-based.

Formalization. Given a sequence of tokens $X = (w_1, w_2, \ldots, w_n)$, a span SRL graph can be defined as a collection of labeled predicate-argument pairs over these tokens: $\mathcal{S} = \{(p, \langle i,j \rangle, r^s), 1 \leq p \leq n, 1 \leq i \leq j \leq n, r^s \in \mathcal{R}^s\}$, where $\mathcal{S}$ represents a labeled predicate-argument pair for predicate p and the argument span located between sentence fence-post positions i and j and with label r^s. A dependency SRL semantic graph for the sentence can be defined as $\mathcal{D} = \{(p, a, r^d), 1 \leq p \leq n, 1 \leq d \leq n, r^d \in \mathcal{R}^d\}$, where (p, a, r^d) consists of a predicate (x_p), an argument (x_a), and the type of the semantic role r^d, which is in label set $\mathcal{R}^d$.

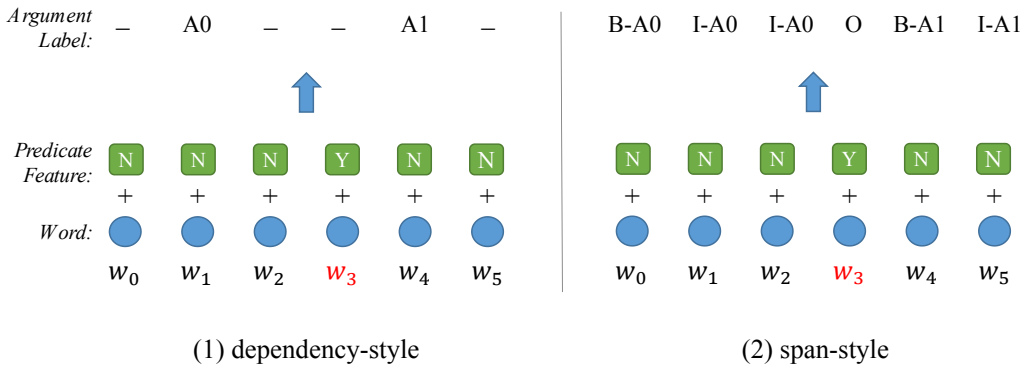


Figure 2
An example of sequence-based factorization.

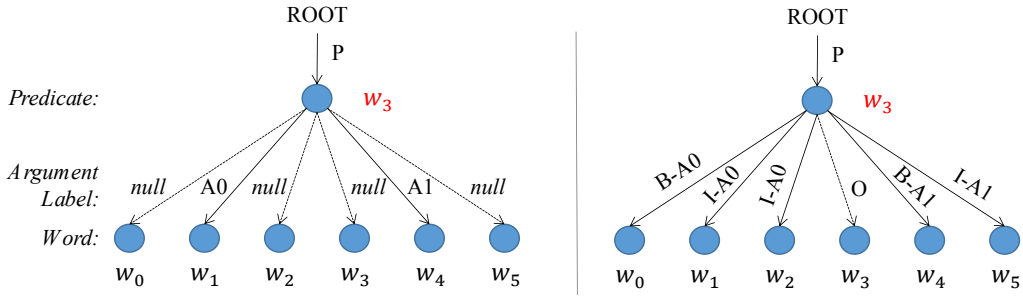
Sequence-based. As shown in Figure 2, the semantic dependency graph of SRL is decomposed by predicates. The arguments for each predicate consist of a sequence of either dependency-style or span-style. Notably, an extra Begin-Inside-Outside (BIO) conversion step is required for span-style argument labels. This decomposition is very simple and efficient. In the baseline model of this factorization, the predicate needs to be input as a source feature, which allows the model to produce different inputs for different target argument sequences. Predicate-specific embeddings are usually used for this reason. In our previous work (He et al. 2018b; Li et al. 2018; Munir, Zhao, and Li 2021), we presented models that recognized and classified arguments as in a sequence labeling task. The predicate-argument pairs were then constructed by performing multiple rounds of sequence labeling according to the number of predicates to obtain a final semantic graph.

In these models, the identification and classification of predicates and the recognition and classification of arguments in the sequence-based modeling are separated into two processes. Formally, the model first identifies the predicate (if not given) and obtains predicate set $\mathcal{P} = \{p_1, p_2, \ldots, p_m\}$. Then, for each $p_i \in \mathcal{P}$ in the predicate set, a sequence labeling model is adopted to predict the argument label of each token:

$$r_1, r_2, \ldots, r_n = \arg\max_{r \in \mathcal{R}}(P(r|w_1, w_2, \ldots, w_n; p_i; \theta)),$$

where θ represents the model parameters, r and $\mathcal{R}$ represent r^s and $\mathcal{R}^s$ in span SRL or r^d and $\mathcal{R}^d$ in dependency SRL, and empty label ϕ ($\phi = null$ in dependency SRL, $\phi = O$ in span SRL) is used to indicate non-arguments. In dependency SRL, a and r^d are obtained after removing the empty labels, while in span SRL, after removing the empty labels, the BIO-converted label should be decoded to get the start and end positions i and j and the span's role label r^s.

Tree-based. Embedding differentiation by relying on the predicate-indicating inputs is only a soft constraint and prompt. This feature may be lost due to forgetting mechanisms such as dropout in the encoder, which potentially limit SRL model performance. For further help in predicate clue integrating, the tree-based method also decomposes the semantic dependency graph to trees with a depth of 2, according to the predicate which is the child node of ROOT; all other nodes are child nodes of the predicate, as



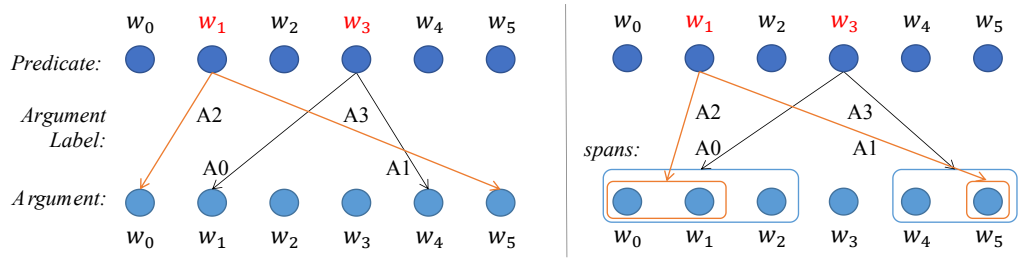
(1) dependency-style (2) span-style

Figure 3
An example of tree-based factorization.

shown in Figure 3. An empty relation $\phi = null$ is set between the non-arguments and the predicate in order to fill the tree. The tree-based factorization can be thought of as an enhanced version of the sequence-based factorization, as the predicate is more prominent and obvious to specify. Also to emphasize a given predicate being handled, predicate-specific embeddings are applied. In our previous work (Cai et al. 2018), the predicate identification and classification and the recognition and classification of arguments are still viewed as two separate processes. Predicate-argument pairs for each identified predicate were scored using the following equation, which follows dependency parsers' head-dependent scoring model rather than scoring the likelihood of a position being an argument:

$$r_1, r_2, \ldots, r_n = \arg\max_{r \in \mathcal{R}}(P(r|\{w_1, w_2, \ldots, w_n\} \otimes p_i; \theta))$$

In tree-based modeling, progressive decoding is performed to output all possible arguments for each predicate.



(1) dependency-style (2) span-style

Figure 4
An example of graph-based factorization. We omit the dashed line between non-predicates and non-arguments, i.e., the empty relation *null*, here.

Graph-based. Sequence-based and tree-based models score the ⟨*argument*, *label*⟩ tuple structure for a determined predicate. The graph-based method further extends this mode; specifically it accommodates undetermined predicates and models the semantic dependency graph directly to output a ⟨*predicate*, *argument*, *label*⟩ triple structure, allowing the model handle to label multiple predicates and arguments at the same time (as shown in Figure 4). This mode not only handles instances without given predicates but also allows instances with given predicates to be enhanced by predicate-specific embeddings. Using dependency-style, the graph-based method is a trivial extension of the tree-based method. Span-style is not so simple because of its argument structure. To account for this, graph-based models enumerate and sort all possible spans, take them as candidate arguments, and score them with the candidate sets of predicates. In our previous work (Li et al. 2019a, 2020), we considered the predicates and arguments jointly and explicitly scored the predicate-argument pairs before classifying their relationships. This modeling objective can be represented as:

$$\{(p, a, r)\} = \underset{p \in \mathcal{P}, a \in \mathcal{A}, r \in \mathcal{R}}{\arg\max} \left(P((p, a, r) | w_1, w_2, \ldots, w_n; \theta)\right)$$

where $\mathcal{P} = \{w_1, w_2, \ldots, w_n\}$ is the set of all predicate candidates, which is used in graph-based modeling instead of relying on the predictions of an additional predicate identification model. In dependency SRL, the argument candidates set $\mathcal{A}$ consists of all words in the sentence, $\mathcal{A} = \{w_1, w_2, \ldots, w_n\}$, and in span SRL, $\mathcal{A}$ consists of all possible spans, $\mathcal{A} = \{(w_i, w_j), 1 \leq i \leq j \leq n\}$.

The above methods cover most mainstream neural SRL models based on semantic dependency graph modeling to the best of our knowledge. There are some modeling approaches, such as transition-based SRL (Choi and Palmer 2011; Fei et al. 2021), that are not based on semantic dependency graphs and hence not the focus of this paper. In the sequence-based and tree-based methods, the BIO conversion is adopted when using span-style, and some works use Conditional Random Fields (CRFs) to model this constraint.

3.2 Baseline Implementation

This subsection presents basic neural SRL models under the three previous aforementioned methods. In order to make fair comparisons with our experiments, we make the architectures of these models as similar as possible.

Word Representation. We produce a predicate-specific word representation e_i for each word w_i in the sequence $w = \{w_1, \cdots, w_n\}$, where i stands for the word position in an input sequence, and n is the length of this sequence, following Marcheggiani, Frolov, and Titov (2017). In this work, word representation $\boldsymbol{e}_i$ is the concatenation of four types of features: a predicate-specific feature and character-level, word-level, and linguistic features. Since previous works demonstrated that the predicate-specific feature is helpful in promoting the role labeling process, we leverage a predicate-specific indicator embedding $\boldsymbol{e}_i^{ie}$ to indicate whether a word is a predicate when predicting and labeling the arguments for each given predicate. At the character level, we exploit a convolutional neural network (CNN) with a bidirectional LSTM (BiLSTM) to learn character embedding $\boldsymbol{e}_i^{ce}$. As shown in Figure 5, the representation calculated by the CNN is fed as input to the BiLSTM. At the word level, we use a randomly initialized word embedding $\boldsymbol{e}_i^{re}$ and a pre-trained word embedding $\boldsymbol{e}_i^{pe}$. For linguistic features,

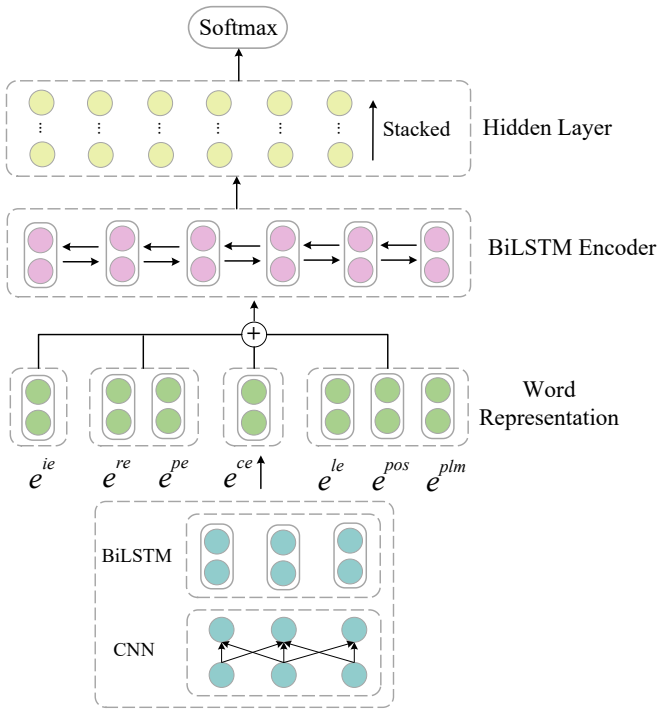


Figure 5
The sequence-based argument labeling baseline model. Notably, the Word Representation and Softmax parts are specific to a single input word/output prediction, and the BiLSTM Encoder and Hidden Layer parts are used across all time steps.

we employ a randomly initialized lemma embedding $\boldsymbol{e}_i^{le}$ and a randomly initialized POS tag embedding $\boldsymbol{e}_i^{pos}$. To further enhance the word representation, we leverage an optimal external representation $\boldsymbol{e}_i^{plm}$ from pre-trained language models. The resulting word representation is concatenated as $\boldsymbol{e}_i = [\boldsymbol{e}_i^{ie}, \boldsymbol{e}_i^{ce}, \boldsymbol{e}_i^{re}, \boldsymbol{e}_i^{pe}, \boldsymbol{e}_i^{le}, \boldsymbol{e}_i^{pos}, \boldsymbol{e}_i^{plm}]$.

Sequence Encoder. As Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber 1997), specifically BiLSTMs, have shown significant representational effectiveness for NLP tasks (Sutskever, Vinyals, and Le 2014; Vinyals et al. 2015), we thus use a BiLSTM as the sentence encoder. Given a sequence of word representations $x = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_n\}$ as input, the i-th hidden state $\boldsymbol{g}_i$ is encoded as follows:

$$\boldsymbol{g}_i^f = LSTM^{\mathcal{F}}\left(\boldsymbol{e}_i, \boldsymbol{g}_{i-1}^f\right), \quad \boldsymbol{g}_i^b = LSTM^{\mathcal{B}}\left(\boldsymbol{e}_i, \boldsymbol{g}_{i+1}^b\right), \quad \boldsymbol{g}_i = \boldsymbol{g}_i^f \oplus \boldsymbol{g}_i^b$$

where $LSTM^{\mathcal{F}}$ denotes the forward LSTM transformation and $LSTM^{\mathcal{B}}$ denotes the backward LSTM transformation. $\boldsymbol{g}_i^f$ and $\boldsymbol{g}_i^b$ are the hidden state vectors of the forward LSTM and backward LSTM, respectively. Specifically, we initialize hidden states g_0 and g_{n+1} as zero tensors.

Scorer in the Sequence-based Model. In the sequence-based model, namely, the sequence tagging model, to get the final predicted semantic roles, stacked multilayer perceptron (MLP) layers on the top of BiLSTM networks are usually exploited, which take as input the hidden representation h_i of all time steps and employ *ReLU* activations between the hidden layers. Finally, a softmax layer is used over the outputs to maximize the likelihood of labels.

Scorer in the Tree-based Model. As in the sequence-based model, to predict and label arguments for a given predicate, a role classifier is employed on top of the BiLSTM encoder. Some work like Marcheggiani, Frolov, and Titov (2017) shows that incorporating the predicate's hidden state in their role classifier enhances the model performance, while we argue that a more natural way to incorporate the syntactic information carried by the predicate is to use the attentional mechanism. We adopt the recently introduced biaffine attention (Dozat and Manning 2017) to enhance our role scorer. Biaffine attention is a natural extension of bilinear attention (Luong, Pham, and Manning 2015), which is widely used in neural machine translation (NMT).

Nonlinear Affine Transformation. Usually, a BiLSTM decoder takes the concatenation $\boldsymbol{g}_i$ of the hidden state vectors as output for each hidden state; however, in the SRL context, the encoder is supposed to distinguish the currently considered predicate from its candidate arguments. As noted in Dozat and Manning (2017), applying an MLP to the recurrent output states before the classifier has the advantage of stripping away irrelevant information for the current decision. Therefore, to distinguish the currently considered predicate from its candidate arguments in an SRL context, we perform two distinct affine transformations with a nonlinear activation on the hidden state $\boldsymbol{g}_i$, mapping it to vectors with smaller dimensionality:

$$\boldsymbol{h}_i^{(pred)} = ReLU\left(\boldsymbol{W}^{(pred)}\boldsymbol{g}_i + \boldsymbol{b}^{(pred)}\right), \quad \boldsymbol{h}_i^{(arg)} = ReLU\left(\boldsymbol{W}^{(arg)}\boldsymbol{g}_i + \boldsymbol{b}^{(arg)}\right)$$

where *ReLU* is the rectilinear activation function (Nair and Hinton 2010), $\boldsymbol{h}_i^{(pred)}$ is the hidden representation for the predicate, and $\boldsymbol{h}_i^{(arg)}$ is the hidden representation for the candidate arguments.

By performing such transformations over the encoder output to feed the scorer, the scorer may benefit from deeper feature extraction. This leads to two benefits. First, instead of keeping both features learned by the two distinct LSTMs, the scorer ideally is now able to learn features composed from both recurrent states with reduced dimensionality. Second, it provides the ability to map the predicates and the arguments into two distinct vector spaces, which is essential for our tasks, since some words can be labeled as predicates and arguments simultaneously. Mapping a word into two different vectors can help the model disambiguate its role in different contexts.

Biaffine Scoring In the standard NMT context, given a target recurrent output vector $h_i^{(t)}$ and a source recurrent output vector $h_j^{(s)}$, a bilinear transformation calculates a score s_{ij} for the alignment:

$$s_{ij} = \boldsymbol{h}_i^{\top(t)}\boldsymbol{W}\boldsymbol{h}_j^{(s)}$$

However, in a traditional classification task, the distribution of classes is often uneven, and the output layer of the model normally includes a bias term designed to capture the prior probability $P(y_i = c)$ of each class, with the rest of the model focusing on learning the likelihood of each class given the data $P(y_i = c|x_i)$. Dozat and Manning (2017) incorporated the bias terms into the bilinear attention to address this uneven problem, resulting in a biaffine transformation, a natural extension of the bilinear transformation and the affine transformation. In the SRL task, the distribution of the role labels is similarly uneven, and the problem worsens after introducing the additional ROOT node and *null* label; directly applying the primitive form of bilinear attention would fail to capture the prior probability $P(y_i = c_k)$ for each class. Thus, introducing the biaffine attention in our model would be extremely helpful for semantic role prediction.

It is worth noting that in our model, the scorer aims to assign a score for each specific semantic role. Besides learning the prior distribution for each label, we wish to further capture the preferences for the label that a specific predicate-argument pair can take. Thus, our biaffine attention contains two distinct bias terms:

$$\boldsymbol{s}_{i,j} = \mathit{Biaffine}(\boldsymbol{h}_j^{(pred)}, \boldsymbol{h}_i^{(arg)}) = \boldsymbol{h}_i^{\top(arg)} \boldsymbol{W}^{(role)} \boldsymbol{h}_j^{(pred)} \quad (1)$$

$$+ \boldsymbol{U}^{(role)} \boldsymbol{h}_i^{(arg)} \oplus \boldsymbol{h}_j^{(pred)} \quad (2)$$

$$+ \boldsymbol{b}^{(role)} \quad (3)$$

where $\boldsymbol{W}^{(role)}$, $\boldsymbol{U}^{(role)}$, and $\boldsymbol{b}^{(role)}$ are parameters that will be updated by gradient descent methods in the learning process. There are several points that should be noted in the above biaffine transformation. First, because our goal is to predict the label for each pair of $\boldsymbol{h}_i^{(arg)}$ and $\boldsymbol{h}_j^{(pred)}$, the output of our biaffine transformation should be a vector of dimensionality N_r instead of a real value, where N_r is the number of all the candidate semantic labels. Thus, the bilinear transformation in Equation (1) maps two input vectors into another vector. This can be accomplished by setting $\boldsymbol{W}^{(role)}$ as a $(d_h \times N_r \times d_h)$ matrix, where d_h is the dimensionality of the hidden state vector. Similarly, the output of the linear transformation in Equation (2) is also a vector by setting $\boldsymbol{U}^{(role)}$ as an $(N_r \times 2d_h)$ matrix. Second, Equation (2) captures the preference of each role (or sense) label and is conditioned on taking the j-th word as a predicate and the i-th word as an argument. Third, the last term $\boldsymbol{b}^{(role)}$ captures the prior probability of each class $P(y_i = c_k)$. Notice that Equations (2) and (3) capture different kinds of bias for the latent distribution of the label set.

Given a sentence of length n, for one of its predicates w_j, the scorer outputs a score vector $\{\boldsymbol{s}_{1,j}, \boldsymbol{s}_{2,j}, \cdots, \boldsymbol{s}_{n,j}\}$. Then, our model picks as its output the label with the highest score from each score vector: $y_{i,j} = \arg\max_{1 \le k \le N_r}(\boldsymbol{s}_{i,j}[k])$, where $\boldsymbol{s}_{i,j}[k]$ denotes the score of the k-th candidate in the semantic label vocabulary with size N_r.

Scorer in the Graph-based Model. As in the scorer of the tree-based model (from the full model shown in Figure 6), the graph-based model (shown in Figure 7) also uses the biaffine scorer to score the predicate-argument structure. Similarly, we also use a nonlinear affine transformation on the top of the BiLSTM encoder. In the sequence-based and tree-based models, dependency- and span-style arguments are converted into a consistent label sequence, while the graph-based model treats arguments as independent graph nodes. In order to unify the two styles of models, we introduce a unified argument representation that can handle both styles of SRL tasks.

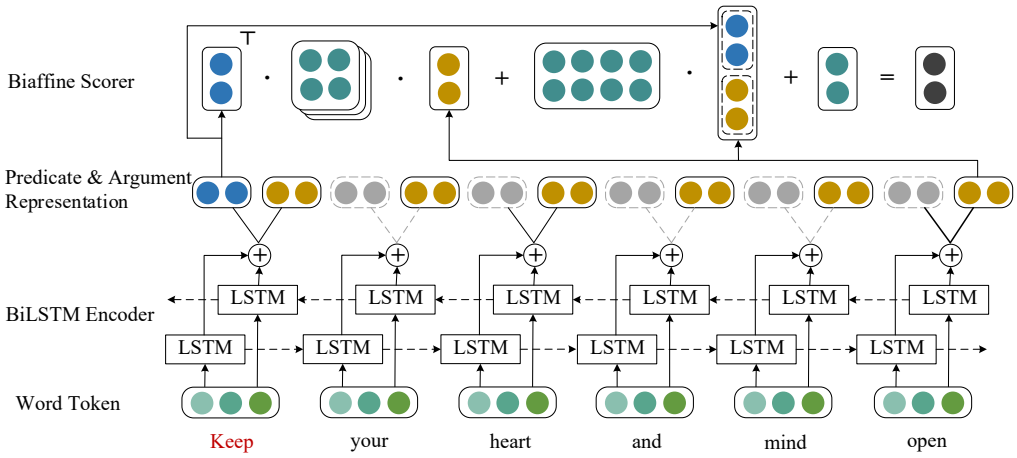


Figure 6
The tree-based argument labeling baseline model.

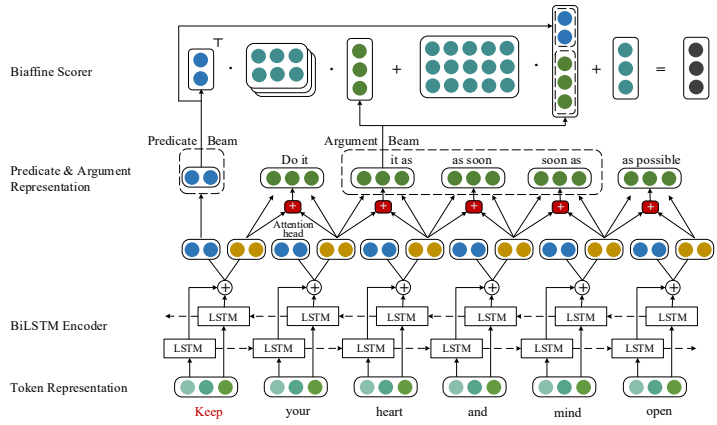


Figure 7
The graph-based argument labeling baseline model.

In the sentence $w_1, w_2, \ldots, w_n$, the model aims to predict a set of predicate-argument-relation tuples $\mathcal{Y} \in \mathcal{P} \times \mathcal{A} \times \mathcal{R}$, where $\mathcal{P} = \{w_1, w_2, \ldots, w_n\}$ is the set of all possible predicate tokens, $\mathcal{A} = \{(w_i, \ldots, w_j) | 1 \leq i \leq j \leq n\}$ includes all the candidate argument spans or dependencies,[4] and $\mathcal{R}$ is the set of the semantic roles. For dependency SRL, we assume single word argument spans and thus limit the length of candidate arguments to be 1, so our model uses h^{arg} to construct the final argument representation $\boldsymbol{h}^{(arg)'}$ directly. For span SRL, we utilize the span representation from Lee et al. (2017). Each candidate span representation $\boldsymbol{h}^{(arg)'}$ is built by

$$\boldsymbol{h}^{(arg)'} = [\boldsymbol{h}^{(arg)}_{START}, \boldsymbol{h}^{(arg)}_{END}, \boldsymbol{h}_{\lambda}, size(\lambda)]$$

4 When $i = j$, span reduces to dependency.

where $\boldsymbol{h}^{arg}_{START}$ and $\boldsymbol{h}^{arg}_{END}$ are boundary representations, λ indicates a span, $size(\lambda)$ is a feature vector encoding the size of span, and h_λ is the specific notion of headedness learned by the attention mechanism (Bahdanau, Cho, and Bengio 2015) over words in each span (where t is the position inside span) as follows:

$$\mu^a_t = \mathbf{w}_{attn} \cdot \mathbf{MLP}_{attn}(\boldsymbol{h}^{(arg)}_t), \quad \nu_t = \frac{\exp(\mu^a_t)}{\sum_{k=START}^{END} \exp(\mu^a_k)}$$

$$h_\lambda = \sum_{t=START}^{END} \nu_t \cdot \boldsymbol{h}^{(arg)}_t$$

Candidate Pruning The number of candidate arguments for a sentence of length l is $O(l^2)$ for span SRL and $O(l)$ for dependency. As the model deals with $O(l)$ possible predicates, the computational complexity is $O(l^3 \cdot |\mathcal{R}|)$ for span and $O(l^2 \cdot |\mathcal{R}|)$ for dependency, both of which are too computationally expensive.

To address this issue, we attempt to prune candidates using two beams for storing the candidate arguments and predicates with size $\beta_p n$ and $\beta_a n$ where β_p and β_a are two manually set thresholds, a method inspired by He et al. (2018a). First, the predicate and argument candidates are ranked according to their predicted scores (ϕ_p and ϕ_a, respectively), and then we reduce the predicate and argument candidates with defined beams. Finally, we take the candidates from the beams for use in label prediction. Such pruning will reduce the overall number of candidate tuples to $O(n^2 \cdot |\mathcal{R}|)$ for both types of tasks. Furthermore, for span SRL, we set the maximum length of candidate arguments to $\mathcal{L}$, which may decrease the number of candidate arguments to $O(n)$. Specifically, for predicates and arguments, we introduce two unary scores based on their candidates for ranking:

$$\phi_p = \mathbf{w}_p \mathbf{MLP}^s_p(g^p), \quad \phi_a = \mathbf{w}_a \mathbf{MLP}^s_a(g^a_f)$$

After pruning, we also adopt the biaffine scorer as in the tree-based models:

$$\Phi_r(p, a) = Biaffine(\boldsymbol{h}^{(pred)}, \boldsymbol{h}^{(arg)'}) \tag{4}$$

4. Syntax Utilization

In this section, we present two types of syntax utilization: syntax-based argument pruning and syntax feature integration.

Algorithm 1 The k-order argument pruning algorithm.

Input: A predicate p, the root node r given a syntactic dependency tree T, the order k
Output: The set of argument candidates S

initialization set p as current node c, $c = p$
for each descendant n_i of c in T **do**
    if $\mathcal{D}(c, n_i) \leq k$ and $n_i \notin S$ **then**
        $S = S + n_i$
    end if
end for
find the syntactic head c_h of c, and let $c = c_h$
if $c = r$ **then**
    $S = S + r$
else
    goto step 2
end if
return argument candidates set S

4.1 Syntax-based Argument Pruning

Hard Pruning.[5] The argument structure for each known predicate will be discovered by our argument labeler using the possible arguments (candidates) set. Most SRL works (Xue and Palmer 2004; Zhao and Kit 2008) in the pre-NN era selected words surrounding the predicate word in a syntactic parse tree and pruned these words. We refer to this strategy as hard pruning. In the NN model, we can also borrow this hard pruning strategy to enhance the SRL baseline, and it is one way of using syntax information. Specifically, before inputting to the model, we use the argument pruning algorithm to get a filtered sequence $w_f = \{w_1, \ldots, w_f\}$ for each predicate. Then, we replace the original sequence with this one and input it to the SRL model.

As noted by Punyakanok, Roth, and Yih (2008), syntactic information is most relevant in identifying the arguments, and the most crucial contribution of full parsing is in the pruning stage. In this paper, we propose a k-order argument hard pruning algorithm inspired by Zhao, Chen, and Kit (2009). First, for node n and its descendant n_d in a syntactic dependency tree, we define the **order** to be the distance between the two nodes, denoted as $\mathcal{D}(n, n_d)$. Then, we define k-order descendants of n as descendants that satisfy $\mathcal{D}(n, n_d) = k$, and we define a k-order traversal as one that visits each node from the given node to its descendant nodes within k-th order. Note that the definition of k-order traversal is somewhat different from a traditional tree traversal in terminology.

A brief description of the proposed k-order pruning algorithm is given as follows. Initially, we set a given predicate as the current node in a syntactic dependency tree. Then, we collect all its argument candidates using a k-order traversal. Afterward, we reset the current node to its syntactic head and repeat the previous step until we reach the root of the tree. Finally, we collect the root and stop. The k-order argument algorithm

5 Notably, "hard pruning" involves the removal of words from a sentence. This does not reflect how pruning techniques have been applied in previous work. Traditionally, pruning in SRL simply meant that an argument span was not considered as a candidate for a given predicate. Sentence structure is typically not affected by this and pruned argument spans are, in fact, still used in the computation of features for other (unpruned) candidates.

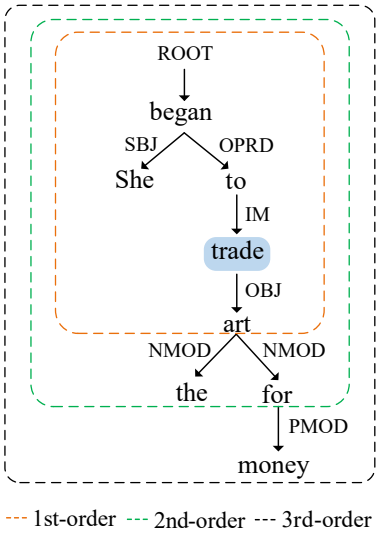


Figure 8
An example of *first*-order, *second*-order, and *third*-order argument pruning. The shaded part indicates the given predicate.

is presented in Algorithm 1 in detail. An example of a syntactic dependency tree for the sentence *She began to trade the art for money* is shown in Figure 8.

The main reasons for applying the extended k-order argument pruning algorithm are two-fold. First, previous standard pruning algorithms may impede the argument coverage too much, even though arguments do usually tend to surround their predicates at a close distance. As a sequence tagging model that has been applied, the algorithm can effectively handle the imbalanced distribution between arguments and non-arguments, which would be poorly handled by early argument classification models that commonly adopt the standard pruning algorithm. Second, the extended pruning algorithm provides a better trade-off between computational cost and performance by carefully tuning k.

Soft Pruning. For word pair classification modeling, one major performance bottleneck is caused by unbalanced data. This is especially pertinent for SRL, where more than 90% of argument candidates are non-arguments. The syntax-based hard pruning methods are thus proposed to alleviate the imbalanced distribution; however, these do not extend well to other baselines and languages and even hinder syntax-agnostic SRL models, as Cai et al. (2018) demonstrated using different k values on English. This hindrance might result because this pruning method breaks up the whole sentence, leading the BiLSTM encoder to take the incomplete sentence as input and fail to learn sentence representation sufficiently.

To alleviate such a drawback from the previous syntax-based pruning methods, we propose a novel pruning rule extraction method based on syntactic parse trees that generally suits diverse baselines at the same time. In detail, we add an argument pruning layer guided by syntactic rules following BiLSTM layers, which can absorb the syntactic clues simply and effectively.

Syntactic Rule. All arguments are specific to a particular predicate. Researchers have found that in syntax trees, the distance between predicates and their arguments generally falls within a certain range for each language; in other words, the arguments of a predicate are typically close to their predicate in their syntactic parse tree (Xue and Palmer 2004; Zhao and Kit 2008; He et al. 2018b; He, Li, and Zhao 2019). Therefore, we introduce a language-specific rule based on syntactic dependency parses to prune some unlikely arguments. We call this rule the **syntactic rule**. Specifically, given a predicate p and its argument a, we define d_p and d_a to be the distance from p and a to their nearest common ancestor node (namely, the root of the minimal subtree that includes p and a), respectively. For example, 0 denotes that a predicate or argument itself is their nearest common ancestor, while 1 represents that their nearest common ancestor is the parent of the predicate or argument. Then, we use the distance tuple (d_p, d_a) as their relative position representation inside the parse tree. Finally, we make a list of all tuples ordered according to how many times each distance tuple occurs in the training data, which is counted for each language independently.

It is worth noting that our syntactic rule is determined by the top-k frequent distance tuples. During training and inference, the syntactic rule takes effect by excluding all candidate arguments whose predicate-argument relative positions in the parse tree are not in the list of top-k frequent tuples.

Figure 9 shows simplified examples of a syntactic dependency tree. Given an English sentence in Figure 9(a), the current predicate is *likes*, whose arguments are *cat* and *fish*. For *likes* and *cat*, the predicate (*likes*) is their common ancestor (denoted as $Root^{arg}$) according to the syntax tree. Therefore, the relative position representation of the predicate and argument is $(0, 1)$, and it is the same for *likes* and *fish*. As for the right side in Figure 9, suppose the marked predicate has two arguments—$arg1$ and $arg2$. The common ancestors of the predicate and arguments are, respectively, $Root^{arg1}$ and $Root^{arg2}$. In this case, the relative position representations are $(0, 1)$ and $(1, 2)$.

Argument Pruning Method. To maintain the sequential inputs through the whole sentence, we propose a novel syntax-based method to softly prune arguments, which is unlike most existing works (Xue and Palmer 2004; Zhao et al. 2009a; He et al. 2018b) with hard pruning strategies that prune argument candidates in the pre-processing stage. Our soft pruning strategy is very straightforward. In the argument pruning layer, our model drops these candidate arguments (more exactly, their BiLSTM representations) that do not comply with the syntactic rule. In other words, only the predicates and

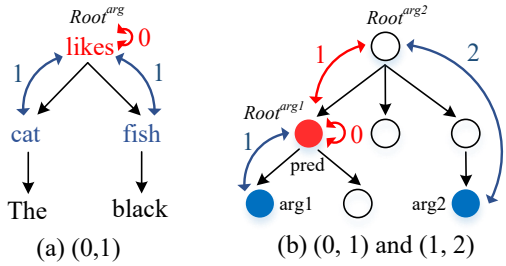


Figure 9
Syntactic parse tree examples (dependency relations are omitted). Red represents the current predicate, and blue indicates its arguments.

arguments that satisfy the syntactic rule will be output to the next layer. Notably, whereas hard pruning removes some of the words from each sentence and tasks the model with processing an incomplete sentence, with soft pruning, the model is given the full original sentence, and by applying a mask instead of discarding part of the inputs. While we do use a "hard" 0/1 binary mask for our "soft" pruning, this step can also be softened to other preset probabilities such as 0.1/0.9 so that the pruned parts can still pass some information. We leave this as an exploration for future work.

Constituent Pruning. In dependency SRL, argument candidates are pruned by a heuristic search over the dependency syntax tree. Constituent syntax trees, which represent the phrasal compositions of sentences, and span SRL have a different relationship than do dependency syntax trees and dependency SRL, which have similarities in their dependency arcs and dependency semantic relations. Since the argument span boundary in span SRL is consistent with that of the phrase in a constituent syntactic tree, we adopt a new constituent-based argument pruning method.

Constituency syntax breaks a sentence into constituents (i.e., phrases or spans), which naturally form a constituency tree in a top–down fashion. In contrast with the dependency syntax tree, words can only be the terminals in a constituency tree, while the non-terminals are phrases with types. In span SRL, each argument corresponds to a constituent in a constituency tree, which can thus be used to generate span argument candidates, given the predicates (Xue and Palmer 2004; Carreras and Màrquez 2005). Punyakanok, Roth, and Yih (2005) showed that constituency trees offer high-quality argument boundaries.

Considering that span SRL models only occasionally violate the syntactic constraints (some candidate arguments may not be constituents), we attempt to prune unlikely arguments based on these constraints, essentially ruling out the likely impossible candidates, albeit at the cost of missing some of the rare violating arguments.

In order to utilize such constituent boundaries in the constituency tree and help decide argument candidates, we extract all boundaries for a constituent c to form a set $\textit{boundaryset} = \{(\textsc{Start}(c), \textsc{End}(c))\}$. We also define an argument pruning layer that drops candidate arguments whose boundaries are not in this set. It is worth noting that because span arguments are converted to BIO labels under the sequence-based and tree-based modeling approaches of span SRL, there is no explicit correspondence between the existing arguments and the constituents, so constituent-based argument pruning is not applicable to the sequence-based and tree-based modeling approaches. We only consider this syntax enhancement when using graph-based modeling.

4.2 Syntax Feature Integration

In addition to guiding argument pruning, another major use of syntax information is serving as a syntax-aware feature in addition to the contextualized representation, thereby enhancing the argument labeler. To integrate the syntactic information into sequential neural networks, we use a syntactic encoder on top of the BiLSTM encoder.

Specifically, given a syntactic dependency tree T, for each node n_k in T, let $C(k)$ denote the syntactic children set of n_k, $H(k)$ denote the syntactic head of n_k, and $L(k, \cdot)$ denote the dependency relation between node n_k and those that have a direct arc from or to n_k. Then, we formulate the syntactic encoder as a transformation f^{τ} over the node n_k, which may take some of $C(k)$, $H(k)$, or $L(k, \cdot)$ as input and compute a syntactic representation v_k for node n_k; namely, $v_k = f^{\tau}(C(k), H(k), L(k, \cdot), x_k)$. When not otherwise specified, x_k denotes the input feature representation of n_k, which may be either the

word representation e_k or the output of BiLSTM h_k. σ denotes the logistic sigmoid function, and $\odot$ denotes the element-wise multiplication.

In practice, the transformation f^{τ} can be any syntax encoding method. In this paper, we will consider three types of syntactic encoders: syntactic graph convolutional network (Syntactic GCN), syntax aware LSTM (SA-LSTM), and tree-structured LSTM (Tree-LSTM).

Syntactic GCN. The GCN (Kipf and Welling 2017) was proposed to induce the representations of nodes in a graph based on the properties of their neighbors. Given its effectiveness, Marcheggiani and Titov (2017) introduced a generalized version for the SRL task, namely, syntactic GCN, and showed that the syntactic GCN is effective in incorporating syntactic information into neural models.

The syntactic GCN captures syntactic information flowing in two directions: one from heads to dependents (along), and the other from dependents to heads (opposite). Additionally, it also models the information flows from a node to itself; that is, it assumes that a syntactic graph contains a self-loop for each node. Thus, the syntactic GCN transformation of a node n_k is defined on its neighborhood $N(k) = C(k) \cup H(k) \cup \{n_k\}$. For each edge that connects n_k and its neighbor n_j, we can compute a vector representation,

$$u_{k,j} = W^{dir(k,j)} x_j + b^{L(k,j)}$$

where $dir(k, j)$ denotes the direction type (along, opposite, or self-loop) of the edge from n_k to n_j, $W^{dir(k,j)}$ is the direction-specific parameter, and $b^{L(k,j)}$ is the label-specific parameter. Considering that syntactic information from all the neighboring nodes may make different contributions to semantic role labeling, the syntactic GCN introduces an additional edge-wise gate for each node pair (n_k, n_j) as

$$g_{k,j} = \sigma(W_g^{dir(k,j)} x_k + b_g^{L(k,j)}).$$

The syntactic representation v_k for a node n_k can be then computed as:

$$v_k = ReLU(\sum_{j \in N(k)} g_{k,j} \odot u_{k,j}).$$

SA-LSTM. The SA-LSTM (Qian et al. 2017) is an extension of the standard BiLSTM architecture, which aims to simultaneously encode the syntactic and contextual information for a given word. On the one hand, the SA-LSTM calculates the hidden state in timestep order as does the standard LSTM,

$$\begin{aligned}
i_g &= \sigma(W^{(i)} x_k + U^{(i)} h_{k-1} + b^{(i)}) \\
f_g &= \sigma(W^{(f)} x_k + U^{(f)} h_{k-1} + b^{(f)}) \\
o_g &= \sigma(W^{(o)} x_k + U^{(o)} h_{k-1} + b^{(o)}) \\
u &= f(W^{(u)} x_k + U^{(u)} h_{k-1} + b^{(u)}) \\
c_k &= i_g \odot u + f_g \odot c_{k-1}
\end{aligned}$$

On the other hand, it further incorporates the syntactic information into the representation of each word by introducing an additional gate,

$$s_g = \sigma(W^{(s)}x_k + U^{(s)}h_{k-1} + b^{(s)})$$
$$h_k = o_g \odot f(c_k) + s_g \odot \tilde{h}_k$$

where $f(\cdot)$ and $\sigma(\cdot)$ represent the tanh and sigmoid activation functions, $\tilde{h}_k = f(\sum_{t_j<t_k} \alpha_j \times h_j)$ is the weighted sum of all hidden state vectors h_j in position t_j (that correspond to previous node [word] n_j) to current timestep t_k, and the weight factor α_j is actually a trainable weight related to the dependency relation $L(k, \cdot)$ when there exists a directed edge from n_j to n_k.

Note that $\tilde{h}_k$ is always the hidden state vector of the syntactic head of n_k according to the definition of α_j. Because a word will be assigned a single syntactic head, such a strict constraint prevents the SA-LSTM from incorporating complex syntactic structures. Inspired by the GCN, we relax the directed constraint of α_j whenever there is an edge between n_j and n_k.

After the SA-LSTM transformation, the outputs of the SA-LSTM layer from both directions are concatenated and taken as the syntactic representation of each word n_k, that is, $v_k = [\overrightarrow{h_k}, \overleftarrow{h_k}]$. Different from the syntactic GCN, SA-LSTM encodes both syntactic and contextual information in a single vector v_k.

Tree-LSTM. The Tree-LSTM (Tai, Socher, and Manning 2015) can be considered an extension of the standard LSTM and aims to model tree-structured topologies. At each timestep, it composes an input vector and the hidden states from arbitrarily many child units. Specifically, the main difference between the Tree-LSTM unit and the standard one is that the memory cell updating and the calculation of gating vectors are dependent on multiple child units. A Tree-LSTM unit can be connected to an arbitrary number of child units, and it assigns a single forget gate for each child unit. This provides Tree-LSTM the flexibility to incorporate or drop the information from each child unit.

Given a syntactic tree, the Tree-LSTM transformation is defined on node n_k and its children set $C(k)$ and is formulated as follows (Tai, Socher, and Manning 2015):

$$\tilde{h}_k = \sum_{j \in C(k)} h_k \tag{5}$$
$$i_g = \sigma(W^{(i)}x_k + U^{(i)}\tilde{h}_k + b^{(i)})$$
$$f_g^{k,j} = \sigma(W^{(f)}x_k + U^{(f)}h_j + b^{(f)}) \tag{6}$$
$$o_g = \sigma(W^{(o)}x_k + U^{(o)}\tilde{h}_k + b^{(o)})$$
$$u = \tanh(W^{(u)}x_k + U^{(u)}\tilde{h}_k + b^{(u)})$$
$$c_k = i_g \odot u + \sum_{j \in C(k)} f_g^{k,j} \odot c_j$$
$$h_k = o_g \odot \tanh(c_k)$$

where $j \in C(k)$, h_j is the hidden state of the j-th child node, c_k is the memory cell of the head node k, and h_k is the hidden state of node k. Note that in Equation (6), a single forget gate $f_g^{k,j}$ is computed for each hidden state h_j.

Note that the primitive form of Tree-LSTM does not take the dependency relations into consideration. Given the importance of dependency relations in the SRL task, we further extend the Tree-LSTM by adding an additional gate r_g and reformulate Equation (5),

$$r_g^{k,j} = \sigma(W^{(r)}x_k + U^{(r)}h_j + b^{L(k,j)})$$
$$\tilde{h}_k = \sum_{j \in C(k)} r_g^{k,j} \odot h_j$$

where $b^{L(k,j)}$ is a relation label-specific bias term. After the Tree-LSTM transformation, the hidden state of each node in the dependency tree is taken as its syntactic representation, that is, $v_k = h_k$.

4.3 Constituent Composition and Decomposition

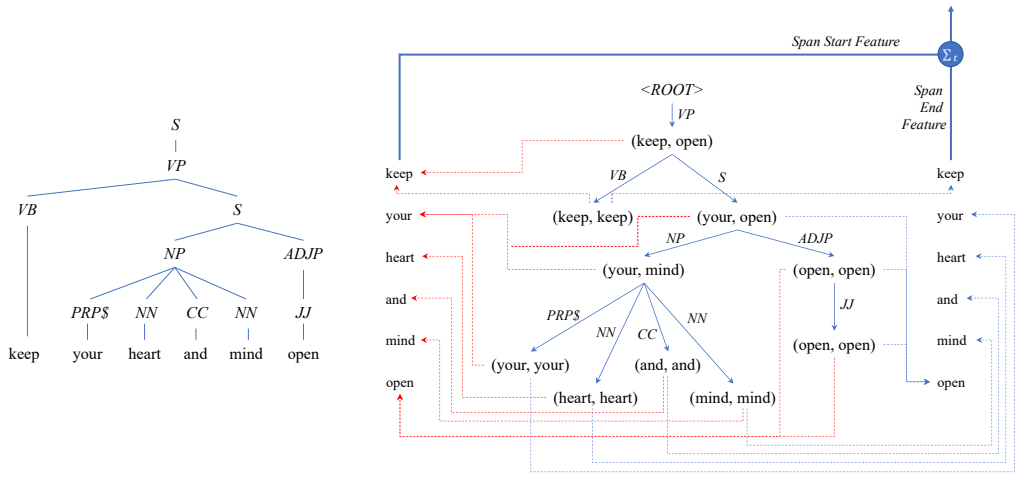
Due to the difference in structure between constituent and dependency syntax trees, tree encoders (GCN, SA-LSTM, Tree-LSTM, etc.) cannot be used to encode the constituent tree directly. In order for the constituent syntax to be encoded into the SRL model as the dependency syntax tree was, inspired by Marcheggiani and Titov (2020), we introduce two processes: constituent tree conversion and feature decomposition.

A constituency tree is composed of terminal nodes and non-terminal nodes, as shown in Figure 10(a). Since the words in a constituent tree all are terminal nodes, if the constituent tree is directly encoded by the tree encoder, the syntax tree structural information cannot be encoded into the words fully. Therefore, we convert the constituent tree to a dependency-like tree, in which the original terminal nodes are removed and the remaining non-terminal nodes are replaced by units consisting of the start and end tokens (words) of the spans they represented. The constituent labels are modified to mimic dependency arcs as in dependency trees, as shown in Figure 10(b).

In a dependency tree, the nodes in the tree are the words in the sentence, so the syntax features output from the tree encoder can be mapped directly to the linear order of the sentences. In our converted constituent tree, the nodes in the tree correspond to the start and end words in the sentence, so we need an additional decomposition process to map this feature back to the word level. As shown in the dashed line in Figure 10(b), every node passes the feature to the first and the last words in their spans, and an extra indicator embedding e^{spos} is appended to distinguish features as starts or ends. Then, these features are input to BiLSTM encoders to obtain the final syntax features for each word.

Specifically, before the tree encoding process, we concatenate the BiLSTM contextualized representations for the start and end positions of the span as the initial node representation; the start and end positions of leaf nodes are both positions themselves. For span (i, j), the initial node representation is:

$$h_{i,j}^n = \text{FFN}([h_i; h_j])$$



(a) Constituent tree (b) Dependency-like Constituent Tree and Feature Decomposition

Figure 10
The structure of an original constituent tree and its dependency-like constituent tree after conversion. The dashed lines map the converted tree encoded features back to the original positions of words in the sentence, i.e., feature decomposition.

where $[\cdot;\cdot]$ represents the concatenation operation, and FFN is a feed-forward layer that is used to keep the model dimension. The syntactic tree is then encoded by the adopted tree encoder[6] to obtain the final tree node representations h^{n*}. We separate the representations of the nodes by splicing different position representations to distinguish whether they come from span starts or ends, and then accumulate all of these representations from span starting or ending, respectively:

$$h_i^t = \text{FFN}([\sum_{i=n.start} [h^{n*}; e^{spos}_{start}]; \sum_{i=n.end} [h^{n*}; e^{spos}_{end}]])$$

where h_i^t represents the syntactic tree features for word w_i and n is a tree node.

5. Experimental Analysis and Syntax Role Study

In this section, we investigate the proposed methods empirically in comparison to the latest SRL models. Moreover, we further explore the syntax role for neural SRL in various architectures. The SRL models are evaluated on the popular CoNLL-2005, CoNLL-2009, and CoNLL-2012 shared tasks following the standard training, development, and test splits. For the SRL task, because the predicate identification subtask is easier than other subtasks, some works only focus on the semantic role prediction with pre-identified predicates, which we name **w/ pred**. There are also many studies that tend to examine settings closer to real-world scenarios, where the predicates are not given and the proposed systems are required to output both the predicates and their corresponding argument. We call this setting **w/o pred**.

6 In the SA-LSTM tree encoder, a linear sequence of tree nodes is required, so we linearize the nodes of the constituent trees with a depth-first algorithm.

The hyperparameters in our model were selected based on the development set. In our experiments, all real vectors are randomly initialized, including 100-dimensional word, lemma, POS tag embeddings, and 16-dimensional predicate-specific indicator embeddings (He et al. 2018b). The pre-trained word embeddings are 100-dimensional GloVe vectors (Pennington, Socher, and Manning 2014) for English and 300-dimensional fastText vectors (Grave et al. 2018) trained on Common Crawl and Wikipedia for other languages. The dimensions of ELMo and BERT word embeddings are of size 1024. Additionally, we use a 3-layer BiLSTM with 400-dimensional hidden states and apply dropout with an 80% keep probability between timesteps and layers. For the biaffine scorer, we use two 300-dimensional affine transformations with the ReLU non-linear activation and also set the dropout probability to 0.2. During training, we use the categorical cross-entropy as the objective and use the Adam optimizer (Kingma and Ba 2015) with an initial learning rate $2e^{-3}$. All models are trained for up to 50 epochs with batch size 64.

In our syntax-based pruning experiments with dependency SRL, we used the predicted syntactic trees given by the CoNLL-2009 data sets (including the multilingual settings) due to the relative insignificance of the syntactic quality in the pruning algorithm. For our syntax feature integration experiments in dependency SRL, because the tree encoder has a minimum syntax tree quality requirement before it can provide any performance improvement, we obtained the predicted dependency syntax tree with the Biaffine Parser (Dozat and Manning 2017) we trained on the golden syntax annotations provided by CoNLL20-09 data sets. In span SRL, following the practice of He et al. (2017), a leading constituency parser (Choe and Charniak 2016) is used to parse the constituent trees. In our experiments, the data sets for the shared task of CoNLL-2009 include predicted POS tags or lemmas, while the data sets for CoNLL-2005 and CoNLL-2012 do not provide them. We remedy this using NLTK to obtain the predicted POS tags and lemmas to keep the input form consistent.

5.1 Data sets

Span-based Data. The CoNLL-2005 shared task focused on verbal predicates for only English. The CoNLL-2005 data set takes sections 2-21 of Wall Street Journal (WSJ) data as the training set; and section 24 as the development set. The test set consists of section 23 of WSJ for in-domain evaluation together with 3 sections from the Brown corpus for out-of-domain evaluation. The larger CoNLL-2012 data set is extracted from OntoNotes v5.0 corpus, which contains both verbal and nominal predicates.

Dependency-based Data. The CoNLL-2009 shared task is focused on dependency-based SRL in multiple languages and merges two treebanks, PropBank and NomBank. NomBank is a complement to PropBank and uses a similar semantic convention for nominal predicate-argument structure annotation. The training, development, and test splits of the English data are identical to those of CoNLL-2005.

5.2 Preprocessing

Hard Pruning. During the pruning of argument candidates, we use the officially predicted syntactic parses provided by CoNLL-2009 shared-task organizers on both English and Chinese. Figure 11 shows changing curves of coverage and reduction following k on the English training set. According to our statistics, the number of non-arguments is ten times more than that of arguments, meaning the data distribution is

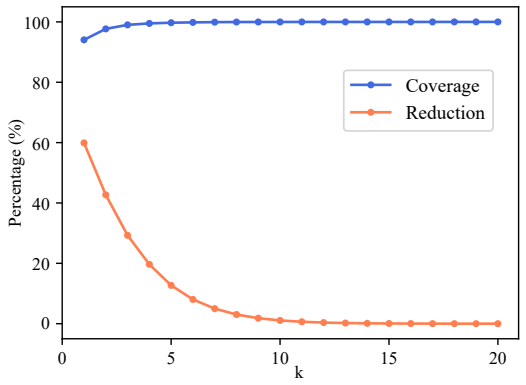


Figure 11
Changing curves of coverage and reduction with different k values on the English training set. The coverage rate is the proportion of true arguments in the pruning output, while the reduction is the proportion of pruned argument candidates in total tokens.

fairly unbalanced; however, a proper pruning strategy could alleviate this problem. Accordingly, the first-order pruning reduces more than 50% of candidates at the cost of missing 5.5% true ones on average, and the second-order prunes about 40% of candidates with nearly 2.0% loss. The coverage of third-order achieves 99%, and it reduces the size of the corpus by approximately one third.

It is worth noting that when k is larger than 19, full coverage is achieved on all argument candidates for the English training set, which allows our high-order pruning algorithm to reduce to a syntax-agnostic setting. In this work, we use tenth-order pruning for best performance.

Soft Pruning. For the syntactic rule used in soft argument pruning, to ensure more than 99% coverage of true arguments in pruning output, we use the top-120 distance tuples on Japanese and top-20 on other languages for a better trade-off between computation and coverage.

Candidate Pruning. In graph-based modeling, the pruning of all predicate-argument pair candidates does not rely on hard pruning or soft pruning based on syntactic trees. Rather, it limits the maximum length of the argument span while also ranking the predicate and argument candidates separately using scores from the neural network scorer and then taking the top-k candidates to reduce the number of candidates. Specifically, we follow the settings of He et al. (2018a), modeling spans up to length $\mathcal{L} = 30$ for span SRL and $\mathcal{L} = 1$ for dependency SRL, using $\beta_p = 0.4$ for pruning predicates and $\beta_a = 0.8$ for pruning arguments.

5.3 Dependency SRL Results

Undoubtedly, dependency SRL offers a number of advantages from a practical perspective, and the efficient dependency parsing algorithms enable SRL models to achieve state-of-the-art results. Therefore, we begin our exploration of syntax roles for neural

SRL with it. In Table 2, we outlined the performance of the current leading dependency SRL models and compared the performance of our three baselines and syntax-enhanced models with different integration approaches on the CoNLL-2009 English in-domain (WSJ) and out-of-domain (Brown) test sets.

In the sequence-based approaches, we utilized another sequence labeling model to tackle the predicate identification and disambiguation subtasks required for the different settings (*w/ pred* and *w/o pred*). The predicate disambiguation model achieves accuracies of 95.01% and 95.58% on the development and test (WSJ) sets for the *w/ pred* setting, respectively, giving a slightly better accuracy than Roth and Lapata (2016),

Table 2
Dependency SRL Results with pre-identified predicates (*w/ pred*) and without pre-identified predicates (*w/o pred*) on the CoNLL-2009 English in-domain (WSJ) and out-of-domain (Brown) test sets. The "PLM" column indicates whether and which pre-trained language model is used, the "SYN" column indicates whether syntax information is employed, and "+E" in the "PLM" column shows that the model leverages ELMo for pre-trained language model features. **[Ens.]** is used to specify the ensemble system, **[Semi.]** indicates semi-supervised training is adopted, and **[Joint]** means joint learning with other tasks.

System	PLM	SYN	*w/ pred*						*w/o pred*					
			WSJ			Brown			WSJ			Brown		
			P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
(Zhao et al. 2009b)		Y	–	–	85.4	–	–	73.3	–	–	–	–	–	–
(Zhao et al. 2009a)		Y	–	–	86.2	–	–	74.6	–	–	–	–	–	–
(Lei et al. 2015)		Y	–	–	86.6	–	–	75.6	–	–	–	–	–	–
(FitzGerald et al. 2015)		Y	–	–	87.3	–	–	75.2	–	–	–	–	–	–
[Ens.] (FitzGerald et al. 2015)		Y	–	–	87.8	–	–	75.5	–	–	–	–	–	–
(Roth and Lapata 2016)		Y	90.0	85.5	87.7	78.6	73.8	76.1	–	–	–	–	–	–
[Ens.] (Roth and Lapata 2016)		Y	90.3	85.7	87.9	79.7	73.6	76.5	–	–	–	–	–	–
(Swayamdipta et al. 2016)		N	–	–	85.0	–	–	–	–	–	80.5	–	–	–
(Marcheggiani and Titov 2017)		Y	89.1	86.8	88.0	78.5	75.9	77.2	–	–	–	–	–	–
[Ens.] (Marcheggiani and Titov 2017)		Y	90.5	87.7	89.1	80.8	77.1	78.9	–	–	–	–	–	–
(Marcheggiani, Frolov, and Titov 2017)		N	88.7	86.8	87.7	79.4	76.2	77.7	–	–	–	–	–	–
(Mulcaire, Swayamdipta, and Smith 2018)		N	–	–	87.2	–	–	–	–	–	–	–	–	–
(Kasai et al. 2019)		Y	89.0	88.2	88.6	78.0	77.2	77.6	–	–	–	–	–	–
	+E	Y	90.3	90.0	90.2	81.0	80.5	80.8	–	–	–	–	–	–
(Cai and Lapata 2019a)		N	91.1	90.4	90.7	82.1	81.3	81.6	–	–	–	–	–	–
[Semi.] (Cai and Lapata 2019a)		N	91.7	90.8	**91.2**	83.2	81.9	82.5	–	–	–	–	–	–
(Zhang, Wang, and Si 2019)		Y	89.6	86.0	87.7	–	–	–	–	–	–	–	–	–
(Lyu, Cohen, and Titov 2019)	+E	N	–	–	91.0	–	–	82.2	–	–	–	–	–	–
(Chen, Lyu, and Titov 2019)	+E	N	90.7	91.4	91.1	82.7	82.8	82.7	–	–	–	–	–	–
[Joint] (Zhou, Li, and Zhao 2020)		N	88.7	89.8	89.3	82.5	83.2	82.8	84.2	87.6	85.9	76.5	78.5	77.5
	+E	N	89.7	90.9	90.3	83.9	85.0	**84.5**	85.2	88.2	**86.7**	78.6	80.8	**79.7**
Sequence-based (2018b; 2018)	+E	N	89.5	87.9	88.7	81.7	76.1	78.8	83.5	82.4	82.9	71.5	70.9	71.2
+K-order Hard Pruning (2018b)	+E	Y	89.7	89.3	89.5	81.9	76.9	79.3	83.9	82.7	83.3	71.5	71.3	71.4
+SynRule Soft Pruning	+E	Y	89.9	89.1	89.5	78.8	81.2	**80.0**	82.9	84.3	83.6	70.9	72.1	71.5
+GCN Syntax Encoder (2018)	+E	Y	90.3	89.3	**89.8**	80.6	79.0	79.8	85.3	82.5	83.9	71.9	71.5	**71.7**
+SA-LSTM Syntax Encoder (2018)	+E	Y	90.8	88.6	89.7	81.0	78.2	79.6	85.3	82.6	**84.0**	71.8	71.6	**71.7**
+Tree-LSTM Syntax Encoder (2018)	+E	Y	90.0	88.8	89.4	80.4	78.7	79.5	83.1	83.7	83.4	70.9	72.1	71.5
Tree-based (2018)	+E	N	89.2	90.4	89.8	80.0	78.6	79.3	84.8	85.4	85.1	72.4	74.0	73.2
+K-order Hard Pruning	+E	Y	90.3	89.5	89.9	80.0	79.0	79.5	83.9	86.5	85.2	73.6	72.8	73.2
+SynRule Soft Pruning (2019)	+E	Y	90.0	90.7	90.3	79.6	80.4	80.0	84.9	85.9	85.4	72.7	74.3	73.5
+GCN Syntax Encoder	+E	Y	90.9	90.1	**90.5**	81.4	78.8	80.1	86.1	84.9	**85.5**	73.5	73.7	**73.6**
+SA-LSTM Syntax Encoder	+E	Y	91.1	89.9	**90.5**	80.9	79.5	**80.2**	85.3	85.0	85.2	72.9	73.5	73.2
+Tree-LSTM Syntax Encoder	+E	Y	89.8	90.6	90.2	80.0	79.8	79.9	85.3	85.3	85.3	73.9	73.1	73.5
Graph-based (2019a)	+E	N	89.6	91.2	90.4	81.7	81.4	81.5	85.6	85.0	85.3	73.0	74.0	73.5
+K-order Hard Pruning	+E	Y	90.3	89.7	90.0	80.7	81.9	81.3	84.6	85.8	85.2	73.7	73.3	73.5
+SynRule Soft Pruning	+E	Y	89.8	90.6	90.2	80.8	82.4	81.6	85.0	86.0	85.5	72.8	74.4	73.6
+GCN Syntax Encoder	+E	Y	90.5	91.7	**91.1**	83.3	80.9	**82.1**	86.2	86.0	**86.1**	73.8	74.6	**74.2**
+SA-LSTM Syntax Encoder	+E	Y	91.0	90.4	90.7	82.4	81.6	82.0	86.3	85.5	85.9	75.4	72.8	74.1
+Tree-LSTM Syntax Encoder	+E	Y	90.7	90.3	90.5	80.2	83.4	81.8	86.9	84.3	85.6	74.1	73.7	73.9

which had 94.77% and 95.47% accuracy on development and test sets, respectively. As for the *w/o pred* setting, the F_1 score of our predicate labeling model is 90.11% and 90.53% on development and test (WSJ) sets, respectively. With the help of the ELMo pretrained language model, our sequence-based baseline model has achieved competitive results when compared to the other leading SRL models. Compared to the closest work (Marcheggiani, Frolov, and Titov 2017), ELMo brought a 1.0% improvement for our baseline model, which verifies it is a strong baseline. In this case, the syntax enhancement gave us a performance improvement of 0.8%–1.1% (*w/ pred*, in-domain test set), demonstrating that both hard/soft pruning and syntax encoders effectively exploit syntax for sequence-based neural SRL models.

In the tree-based approaches, the predicate disambiguation subtask is unifiedly tackled with argument labeling by making predictions on the ROOT node of the factorized tree. The disambiguation precision is 95.0% in the *w/ pred* setting, whereas in the *w/o pred* setting, we first attach all the words in the sentence to the ROOT node and label the word that is not a predicate with the *null* role label. It should be noted that in the *w/o pred* setting, we just attach the predicates to the ROOT node, since we do not need to distinguish the predicate from other words. The training scheme remains the same as in the *w/o pred* setting, whereas in the inference phase, an additional procedure is performed to find out all the predicates of a given sentence. The F_1 score on predicate identification and labeling of this process is 89.43%. Based on the tree-based baseline, the hard pruning syntax enhancement fails to improve on the baseline despite the hard pruning method's ability to alleviate the imbalanced label distribution caused by the *null* role labels. We suspect the possible reason is the use of a biaffine attention structure, which already alleviates imbalanced label distribution issue. This is problematic because both the biaffine attention and hard pruning work to balance the label distribution, and after the biaffine attention balances it to some degree, hard pruning is much more likely to incorrectly prune true labels, which potentially even leads to a decrease in performance. Compared with hard pruning, soft pruning can greatly reduce the incorrect pruning of true arguments, which serve as clues in the model. Because of this, the soft pruning algorithm applied to the tree-based model can obtain performance improvement similar to that of the tree-based baseline. In addition, the performance improvements of syntax encoders in the tree-based model are similar to those of the sequence-based model.

In the graph-based approaches, because of the introduction of candidate pruning, argument pruning is directly controlled by the neural network scorer, and both syntax-based hard and soft pruning methods lose the effects they provided in the sequence-based and tree-based models. The syntax encoder, however, can provide quite stable performance improvement as it does in sequence-based and tree-based models.

Though most SRL literature is dedicated to impressive performance gains on the English benchmark, exploring syntax's enhancing effects on diverse languages is also important for examining the role of syntax in SRL. Table 3 presents all in-domain test results on seven languages of CoNLL-2009 data sets. Compared with previous methods, our baseline yields strong performance on all data sets. Nevertheless, applying the syntax information to the strong syntax-agnostic baseline can still boost the model performance in general, which demonstrates the effectiveness of syntax information. On the other hand, the similar performance impact of hard/soft argument pruning on the baseline models indicates that syntax is generally beneficial to multiple languages and can enhance multilingual SRL performance with effective syntactic integration.

Based on the above results and analysis, we conclude that the syntax-based pruning algorithms proposed before the era of neural network can still play a role under certain

Table 3
Dependency SRL results on the CoNLL-2009 multilingual in-domain test sets with pre-identified predicates (*w/ pred*) setting. The first row is the best result of the CoNLL-2009 shared task (Hajič et al. 2009). The "PLM" column indicates whether and which pre-trained language model is used, the "SYN" column indicates whether syntax information is employed, "+E" in the "PLM" column indicates the model leverages pre-trained ELMo features for all languages, and "+E†" in the "PLM" column indicates ELMo is only used for English.

System	PLM	SYN	CA	CS	DE	EN	ES	JA	ZH	Avg.
CoNLL-2009 best		Y	80.3	86.5	79.7	86.2	80.5	78.3	78.6	81.4
(Zhao et al. 2009a)		Y	80.3	85.2	76.0	85.4	80.5	78.2	77.7	80.5
(Roth and Lapata 2016)		Y	–	–	80.1	87.7	80.2	–	79.4	–
(Marcheggiani and Titov 2017)		Y	–	–	–	88.0	–	–	82.5	–
(Marcheggiani, Frolov, and Titov 2017)		N	–	86.0	–	87.7	80.3	–	81.2	–
(Mulcaire, Swayamdipta, and Smith 2018)		N	79.5	85.1	70.0	87.2	77.3	76.0	81.9	79.6
(Kasai et al. 2019)	+E	Y	–	–	–	90.2	**83.0**	–	–	
(Cai and Lapata 2019a)		N	–	–	83.3	90.7	82.1	–	84.6	–
[Semi.] (Cai and Lapata 2019a)		N	–	–	**83.8**	**91.2**	82.9	–	**85.0**	–
(Zhang, Wang, and Si 2019)		Y	–	–	–	87.7	–	–	84.2	–
(Lyu, Cohen, and Titov 2019)	+E†	N	80.9	87.6	75.9	91.0	80.5	**82.5**	83.3	**83.1**
(Chen, Lyu, and Titov 2019)	+E†	N	**81.7**	**88.1**	76.4	91.1	81.3	81.3	81.7	**83.1**
Sequence-based (2018b; 2018)	+E	N	84.0	87.8	76.8	88.7	82.9	82.8	83.1	83.7
+K-order Hard Pruning (2018b)	+E	Y	84.5	88.3	77.3	89.5	83.3	82.9	82.8	84.1
+SynRule Soft Pruning	+E	Y	84.4	88.2	**77.5**	89.5	83.2	83.0	83.3	84.2
+GCN Syntax Encoder (2018)	+E	Y	**84.6**	**88.5**	77.2	**89.8**	**83.6**	**83.2**	**83.8**	84.4
+SA-LSTM Syntax Encoder (2018)	+E	Y	84.3	**88.5**	77.0	89.7	83.5	83.1	83.5	84.2
+Tree-LSTM Syntax Encoder (2018)	+E	Y	84.1	88.3	76.9	89.4	83.2	82.9	83.4	84.0
Tree-based (2018)	+E	N	84.1	88.4	78.4	89.9	83.5	83.0	84.0	84.5
+K-order Hard Pruning	+E	Y	84.2	88.5	78.4	89.9	83.4	82.8	84.2	84.5
+SynRule Soft Pruning (2019)	+E	Y	84.4	88.8	78.5	90.0	83.7	83.1	84.6	84.7
+GCN Syntax Encoder	+E	Y	**84.8**	**89.3**	78.1	**90.2**	**84.0**	**83.3**	**85.0**	**85.0**
+SA-LSTM Syntax Encoder	+E	Y	84.6	89.0	**78.8**	90.0	83.7	83.1	84.8	84.9
+Tree-LSTM Syntax Encoder	+E	Y	84.4	88.9	78.6	89.9	83.6	83.0	84.5	84.7
Graph-based (2019a)	+E	N	85.0	90.2	76.0	90.0	83.8	82.7	85.7	84.8
+K-order Hard Pruning	+E	Y	84.9	90.2	75.7	89.8	83.5	82.8	85.8	84.7
+SynRule Soft Pruning	+E	Y	85.2	90.3	76.2	90.1	84.0	82.9	85.8	84.9
+GCN Syntax Encoder	+E	Y	**85.5**	**90.5**	**76.6**	**90.4**	**84.3**	**83.2**	**86.1**	**85.2**
+SA-LSTM Syntax Encoder	+E	Y	85.2	**90.5**	76.4	90.3	84.1	**83.2**	86.0	85.1
+Tree-LSTM Syntax Encoder	+E	Y	85.0	90.3	76.2	90.3	84.0	83.0	85.8	84.9

conditions in the neural network era; however, when there are neural structures whose motivation is consistent with the original intention of these syntax-based pruning algorithms, the effects of these algorithms are quite limited, and they can even have negative effects. Despite this limitation, the neural syntax encoder is beneficial in that it delegates the decisions of how to include syntax and what kind of syntax to use in neural networks, which reduces the number of manually defined features. This is an important result of the transition from the pre-neural network era to the neural network era. Handcrafted features may be advantageous when compared to poorly designed neural networks, but well-designed neural networks can easily outperform models relying on handcrafted features. In addition, neural models can also benefit from handcrafted features, so the two do not necessarily have to be directly compared, even though neural networks reduce the need for handcrafted features.

5.4 Span SRL Results

Apart from the dependency SRL experiments, we also conducted experiments to compare different syntax utilization on span SRL models. Table 4 shows results on the

Table 4
Span SRL results with pre-identified predicates on the CoNLL-2005 and CoNLL-2012 test sets. The "PLM" column indicates whether and which pre-trained language model is used, the "SYN" column indicates whether syntax information is employed, and "+E" in the "PLM" column shows that the model leverages the ELMo for pre-trained language model features. **[Ens.]** is used to specify the ensemble system and **[Joint]** means joint learning with other tasks.

System	PLM	SYN	CoNLL05 WSJ			CoNLL05 Brown			CoNLL12		
			P	R	F_1	P	R	F_1	P	R	F_1
[Ens.] (Punyakanok, Roth, and Yih 2008)		Y	82.3	76.8	79.4	73.4	62.9	67.8	—	—	—
(Toutanova, Haghighi, and Manning 2008)		Y	—	—	79.7	—	—	67.8	—	—	—
[Ens.] (Toutanova, Haghighi, and Manning 2008)		Y	81.9	78.8	80.3	—	—	68.8	—	—	—
(Pradhan et al. 2013)*		Y	—	—	—	—	—	—	78.5	76.6	77.5
(Täckström, Ganchev, and Das 2015)		Y	82.3	77.6	79.9	74.3	68.6	71.3	80.6	78.2	79.4
(Zhou and Xu 2015)		N	82.9	82.8	82.8	70.7	68.2	69.4	—	—	81.3
(FitzGerald et al. 2015)		Y	81.8	77.3	79.4	73.8	68.8	71.2	80.9	78.4	79.6
[Ens.] (FitzGerald et al. 2015)		Y	82.5	78.2	80.3	74.5	70.0	72.2	81.2	79.0	80.1
(He et al. 2017)		Y	83.1	83.0	83.1	72.9	71.4	72.1	81.7	81.6	81.7
[Ens.] (He et al. 2017)		Y	85.0	84.3	84.6	74.9	72.4	73.6	83.5	83.3	83.4
(Yang and Mitchell 2017)		N	—	—	81.9	—	—	72.0	—	—	—
(Tan et al. 2018)		N	84.5	85.2	84.8	73.5	74.6	74.1	81.9	83.6	82.7
[Ens.] (Tan et al. 2018)		N	85.9	86.3	86.1	74.6	75.0	74.8	83.3	84.5	83.9
(Peters et al. 2018)		N	—	—	—	—	—	—	—	—	81.4
	+E	N	—	—	—	—	—	—	—	—	84.6
(He et al. 2018a)		N	—	—	83.9	—	—	73.7	—	—	82.1
	+E	N	—	—	87.4	—	—	**80.4**	—	—	85.5
(Strubell et al. 2018)		N	84.7	84.2	84.5	73.9	72.4	73.1	—	—	—
		Y	84.6	84.6	84.6	74.8	74.3	74.6	—	—	—
(Ouchi, Shindo, and Matsumoto 2018)		N	84.7	82.3	83.5	76.0	70.4	73.1	84.4	81.7	83.0
	+E	N	88.2	87.0	87.6	79.9	77.5	78.7	87.1	85.3	86.2
(Wang et al. 2019)	+E	N	—	—	87.7	—	—	78.1	—	—	85.8
	+E	Y	—	—	**88.2**	—	—	79.3	—	—	**86.4**
(Marcheggiani and Titov 2020)		Y	85.8	85.1	85.4	76.2	74.7	75.5	84.5	84.3	84.4
[Joint] (Zhou, Li, and Zhao 2020)		N	85.9	85.8	85.8	76.9	74.6	75.7	—	—	—
	+E	N	87.8	88.3	88.0	79.6	78.6	79.1	—	—	—
Sequence-based	+E	N	87.4	85.6	86.5	80.0	78.1	79.0	84.2	85.6	84.9
+GCN Syntax Encoder	+E	Y	87.2	86.8	**87.0**	78.6	80.2	**79.4**	85.3	85.7	**85.5**
+SA-LSTM Syntax Encoder	+E	Y	87.1	86.5	86.8	79.3	78.9	79.1	85.9	84.3	85.1
+Tree-LSTM Syntax Encoder	+E	Y	87.1	85.9	86.5	78.8	79.2	79.0	85.2	84.2	84.7
Tree-based	+E	N	88.8	86.0	87.4	79.9	79.5	79.7	86.6	84.8	85.7
+GCN Syntax Encoder	+E	Y	87.7	88.3	**88.0**	81.1	79.9	**80.5**	86.9	85.5	**86.2**
+SA-LSTM Syntax Encoder	+E	Y	87.5	87.6	87.6	80.4	79.8	80.1	86.3	85.3	85.8
+Tree-LSTM Syntax Encoder	+E	Y	87.0	87.6	87.3	81.0	79.1	80.0	86.0	85.6	85.8
Graph-based (2019a)	+E	N	87.9	87.5	87.7	80.6	80.4	80.5	85.7	86.3	86.0
+Constituent Soft Pruning	+E	Y	88.4	87.4	87.9	80.9	80.3	80.6	85.5	86.9	86.2
+GCN Syntax Encoder	+E	Y	89.0	88.2	**88.6**	80.8	81.2	81.0	87.2	86.2	**86.7**
+SA-LSTM Syntax Encoder	+E	Y	88.6	87.8	88.2	81.0	81.2	**81.1**	87.0	85.8	86.4
+Tree-LSTM Syntax Encoder	+E	Y	86.9	89.1	88.0	81.5	80.3	80.9	86.6	86.0	86.3

CoNLL-2005 in-domain (WSJ) and out-of-domain (Brown) test sets, as well as the CoNLL-2012 test set (OntoNotes). The first block of the table presents results from previous work. These results demonstrate that with the development of neural networks, in particular the emergence of pre-trained language models, SRL achieved a large performance increase of more than 8.0%, and syntax further enhanced the effect of these strong baselines, enabling syntax+pre-trained language models to achieve the state-of-the-art results (Wang et al. 2019). This indicates that the effect of SRL can still be improved as long as the syntax is used properly under current circumstances.

Additionally, comparing the results of Strubell et al. (2018) and He et al. (2018a), it can be found that the feature extraction ability of self-attention is stronger than that of RNN, and the self-attention baseline obviously outperforms the RNN-based one, but when syntactic information or a pre-trained language model is used to enhance performance, the performance margin becomes smaller. Therefore, we can speculate

that self-attention implicitly and partially functions as the syntax information, as do pre-trained language models.

By comparing our full model to state-of-the-art SRL systems, we show that our model genuinely benefits from incorporating syntactic information and other modeling factorization. Although our use of constituent syntax requires the composition and decomposition processes, which contrasts the simple and intuitive dependency syntax, we achieved consistent improvements compared to dependency SRL on all three baselines: sequence-based, tree-based, and graph-based. This shows that these syntax encoders are general for syntax choice and can encode syntax effectively.

Constituent syntax information is usually used in span SRL, while the dependency tree is adopted for the argument pruning algorithm. Additionally, in the sequence-based and tree-based factorizations of span SRL, argument spans are linearized with multiple B-, I-, and O- tags, which alleviates some label imbalance problems and hence lessens the need for argument pruning. Constituent syntax mainly provides the boundary information of span for the model to guide the model to predict the correct argument span when it is used for pruning. In graph-based models, because the argument span exists alone, the boundary set obtained by the constituent tree can be used to prune candidate arguments. As shown in the results, constituent-based soft pruning can still improve performance on the graph-based baseline, but the improvement is smaller than that of syntax encoders, indicating that the syntax encoder can extract more information than just span boundaries.

We report the experimental results on the CoNLL-2005 and -2012 data sets without pre-identified predicates in Table 5. Overall, our syntax-enhanced model using ELMo

Table 5
Span SRL results without pre-identified predicates on the CoNLL-2005 and CoNLL-2012 data sets. The "PLM" column indicates whether and which pre-trained language model is used, the "SYN" column indicates whether syntax information is employed, and "+E" in the "PLM" column shows that the model leverages the ELMo for pre-trained language model features. **[Ens.]** is used to specify the ensemble system and **[Joint]** means joint learning with other tasks.

System	PLM	SYN	CoNLL05 WSJ			CoNLL05 Brown			CoNLL12		
			P	R	F_1	P	R	F_1	P	R	F_1
(He et al. 2017)		N	80.2	82.3	81.2	67.6	69.6	68.5	78.6	75.1	76.8
[Ens.] (He et al. 2017)		N	82.0	83.4	82.7	69.7	70.5	70.1	80.2	76.6	78.4
(He et al. 2018a)	+E	N	84.8	87.2	86.0	73.9	78.4	76.1	81.9	84.0	82.9
(Strubell et al. 2018)		Y	84.0	83.2	83.6	73.3	70.6	71.9	81.9	79.6	80.7
	+E	Y	86.7	86.4	**86.6**	79.0	77.2	**78.1**	84.0	82.3	**83.1**
[Joint] (Zhou, Li, and Zhao 2020)		N	83.7	85.5	84.6	72.0	73.1	72.6	–	–	–
	+E	N	85.3	87.7	86.5	76.1	78.3	77.2	–	–	–
Sequence-based	+E	N	84.4	83.6	84.0	76.5	73.9	75.2	81.7	82.9	82.3
+GCN Syntax Encoder	+E	Y	85.5	84.3	**84.9**	78.8	73.4	**76.0**	83.1	82.5	**82.8**
+SA-LSTM Syntax Encoder	+E	Y	85.0	84.2	84.6	74.9	76.7	75.8	83.1	81.9	82.5
+Tree-LSTM Syntax Encoder	+E	Y	84.7	84.1	84.4	76.2	75.2	75.7	82.7	81.9	82.3
Tree-based	+E	N	85.4	83.6	84.5	76.1	75.1	75.6	83.3	81.9	82.6
+GCN Syntax Encoder	+E	Y	84.5	85.9	**85.2**	76.7	75.9	**76.3**	82.9	83.3	**83.1**
+SA-LSTM Syntax Encoder	+E	Y	85.0	85.0	85.0	77.2	75.0	76.1	83.5	82.7	**83.1**
+Tree-LSTM Syntax Encoder	+E	Y	85.7	84.1	84.9	75.5	75.9	75.7	83.0	82.4	82.7
Graph-based (2019a)	+E	N	85.2	87.5	86.3	74.7	78.1	76.4	84.9	81.4	83.1
+Constituent Soft Pruning	+E	Y	87.1	85.7	86.4	77.0	76.2	76.6	83.4	83.2	83.3
+GCN Syntax Encoder	+E	Y	86.9	86.5	**86.7**	77.5	76.3	**76.9**	84.4	83.0	**83.7**
+SA-LSTM Syntax Encoder	+E	Y	87.3	85.7	86.5	76.0	77.2	76.6	83.8	83.2	83.5
+Tree-LSTM Syntax Encoder	+E	Y	85.8	86.6	86.2	76.9	76.1	76.5	83.6	82.8	83.2

achieved the best F_1 scores on the CoNLL-2005 in-domain and CoNLL-2012 test sets. In comparison with the three baselines, our syntax utilization approaches consistently yielded better F_1 scores regardless of the factorization. Although the performance difference is small when using the constituent soft pruning on the graph-based model, the improvement seems natural because the constituent syntax for SRL has more to be explored than just boundary information.

5.5 Dependency vs. Span

It is very hard to say which style of semantic formal representation, dependency or span, would be more convenient for machine learning as they adopt incomparable evaluation metrics. Recent research (Peng et al. 2018) has proposed to learn semantic parsers from multiple data sets in FrameNet style semantics, while our goal is to compare the quality of different models in span and dependency SRL for Propbank style semantics. Following Johansson and Nugues (2008a), we choose to directly compare their performance in terms of dependency-style metric through transformation. Using the head-finding algorithm in Johansson and Nugues (2008a) which used gold-standard syntax, we may determine a set of head nodes for each span. This process will output an upper bound performance measure about the span conversion due to the use of gold syntax.

Based on our syntax-agnostic graph-based baseline, we do not train new models for the conversion and the resultant comparison. Instead, we use the span-style CoNLL-2005 test set and the dependency-style CoNLL-2009 test set (WSJ and Brown), considering these two test sets share the same text content. As the former only contains verbal predicate-argument structures, for the latter, we discard all nominal predicate-argument related results and predicate disambiguation results during performance statistics. CoNLL-2005 and CoNLL-2009 have inconsistent tokenization standards, but there exists a straightforward conversion that does not skew results. Because CoNLL-2009 tokenizes words into more fine-grained segments, in order to avoid the need to modify the dependency tree annotations, we re-tokenize the data of CoNLL-2005 according to the tokenization standard adopted in CoNLL-2009. For the constituent tree and span SRL annotations, this further tokenization will not affect the boundary of the spans, so the constituent tree and span SRL annotations are not affected. The detailed conversion process can be found in our paper (Li et al. 2019b).

Table 6 shows the comparison. On a more strict setting basis, the results from our same model for span and dependency SRL verify the same conclusion of Johansson

Table 6
Comparing our performance Dependency SRL with Span-converted Dependency SRL and on Span SRL with Dependency-converted Span SRL using the CoNLL-2005 and CoNLL-2009 test sets, respectively.

		Dep F_1	Span-converted F_1	Δ F_1
WSJ	J & N	85.93	84.32	1.61
	Our system	90.41	89.20	1.21
WSJ+ Brown	J & N	84.29	83.45	0.84
	Our system	88.91	88.23	0.68
		Span F_1	Dep-converted F_1	Δ F_1
WSJ	Our system	87.70	87.23	0.47
WSJ+Brown		86.03	85.62	0.41

and Nugues (2008a), namely, dependency form is more favorable in machine learning for SRL, even compared to the conversion upper bound of the span form. While we used golden dependency trees for the conversion from span SRL to dependency SRL, some argument spans and constituent spans differ, meaning an argument span can have multiple head words. This means the conversion is not fully accurate and thus produces errors. To prevent these errors from impacting our evaluation, we also added the conversion from dependency SRL to span SRL. From the comparison, we found that although there are errors in both directions of conversion, the difference between the result of dependency-converted SRL and the original span SRL system is less than that of span-converted and the original dependency SRL system. With this observation, we can roughly conclude that dependency is a more computationally friendly form (Johansson and Nugues 2008b). This suggests that if we can only perform one type of SRL training but need two types of SRL outputs, we can prioritize selecting the training dependency SRL and then getting the span SRL based on the conversion to achieve the best overall performance.

5.6 Syntax Role under Different Pre-trained Language Models

Language modeling, as an unsupervised natural language training technique, can produce a pre-trained language model by training generally on a large amount of text before further training on a more specific one out of a variety of natural language processing tasks. The downstream tasks then use the obtained pre-trained models for further enhancement. After the introduction of pre-trained language models, they quickly and greatly dominated the performance of downstream tasks. Typical language models are ELMo (Peters et al. 2018), BERT (Devlin et al. 2019), RoBERTa (Liu et al. 2019), XLNet (Yang et al. 2019), and ALBERT (Lan et al. 2020), and so on. Therefore, in order to further explore the role of syntax in SRL on the strong basis of pre-trained language models, we evaluated the performance of these typical pre-trained language models on the best-performing combination of the graph-based baseline and GCN syntax encoder. The experimental results are shown in Table 7.

From the results in the table, all systems using the pre-trained language model have been greatly improved compared with the baseline models, especially in the out-of-domain test set. This indicates that since the pre-trained language model is trained on a very large scale corpus, the performance decrease caused by the domain inconsistency between the training and the test data is reduced, and the generalization ability for the SRL model is enhanced.

For the role of syntax, we found that the improvement from syntax enhancement in the baseline is greater than the improvement that systems gained from using the pre-trained language models. When the ability of pre-trained language models is strengthened, the performance growth brought by syntax gradually declines, and in some very strong language models, syntax even brings no further improvements for SRL. We suspect the reason is that the pre-trained language model can learn some syntactic information implicitly through unsupervised language modeling training (Hewitt and Manning 2019; Clark et al. 2019), thus superimposing explicit syntactic information will not bring as much improvement as it will on the baseline model. We also found that the implicit features for pre-trained language models do not fully maximize syntactic information, as providing explicit syntactic information can still lead to improvements, though this depends on the quality of syntactic information and the ability to extract explicit syntactic information. When explicit syntax is accurate enough and the syntax feature encoder is strong enough, the syntax information can further enhance the ac-

Table 7
SRL results with different pre-trained language models on CoNLL-2005, CoNLL-2009, and CoNLL-2012 test sets. The results listed in the table are evaluated while given pre-identified predicates. The baseline is our proposed graph-based model because of its good performance. The "SYN" column indicates whether syntax information is employed, and the GCN syntax encoder is adopted for all models that are enhanced with syntax information. In this table, "△" in brackets represents relative improvements using syntax information compared to the syntax-agnostic model, while the "↑" indicates an absolute improvement using a pre-trained language model compared to the pure baseline model without any syntax information or pre-trained language model enhancement.

System	SYN	CoNLL09 WSJ	CoNLL09 Brown	CoNLL05 WSJ	CoNLL05 Brown	CoNLL12
Baseline	N	87.8	79.2	84.5	74.8	83.5
	Y	89.2 (△ +1.4)	80.1 (△ +0.9)	85.6 (△ +1.1)	76.2 (△ +1.4)	84.7 (△ +1.2)
ELMo	N	90.4 (↑ +2.6)	81.5 (↑ +2.3)	87.7 (↑ +3.2)	80.5 (↑ +5.7)	86.0 (↑ +2.5)
	Y	91.1 (△ +0.7)	82.1 (△ +0.6)	88.6 (△ +0.9)	81.0 (△ +0.5)	86.7 (△ +0.7)
BERT	N	91.4 (↑ +3.6)	82.8 (↑ +3.6)	89.0 (↑ +4.5)	82.3 (↑ +7.5)	87.4 (↑ +3.9)
	Y	91.8 (△ +0.4)	83.2 (△ +0.4)	89.6 (△ +0.6)	82.8 (△ +0.5)	87.9 (△ +0.5)
RoBERTa	N	91.4 (↑ +3.6)	83.1 (↑ +3.9)	89.3 (↑ +4.8)	82.7 (↑ +7.9)	87.9 (↑ +4.4)
	Y	91.7 (△ +0.3)	83.2 (△ +0.1)	89.7 (△ +0.4)	83.4 (△ +0.7)	88.0 (△ +0.1)
XLNet	N	91.5 (↑ +3.7)	84.1 (↑ +4.9)	89.8 (↑ +5.3)	85.2 (↑ +10.4)	88.2 (↑ +4.7)
	Y	91.6 (△ +0.1)	84.2 (△ +0.1)	89.8 (△ +0.0)	85.4 (△ +0.2)	88.3 (△ +0.1)
ALBERT	N	91.6 (↑ +3.8)	84.0 (↑ +4.8)	90.0 (↑ +5.5)	84.9 (↑ +10.1)	88.5 (↑ +5.0)
	Y	91.6 (△ +0.0)	84.3 (△ +0.3)	90.1 (△ +0.1)	85.1 (△ +0.2)	88.7 (△ +0.2)

curacy of SRL; however, if these conditions are not satisfied, then syntax is no longer the first option for promoting SRL performance in the neural network era. Besides, due to the uncontrollability of the implicit features of the pre-trained models, syntax as an auxiliary tool for SRL will exist for a long time, and neural SRL models can see even higher accuracy gains by leveraging syntactic information rather than ignoring it until neural networks' black box is fully revealed.

5.7 Syntactic Contribution

Syntactic information plays an informative role in semantic role labeling; however, few studies have been done to quantitatively evaluate the contribution of syntax to SRL. In dependency SRL, we observe that most of the neural SRL systems compared above used the syntactic parser of Björkelund et al. (2010) for syntactic inputs instead of the one from the CoNLL-2009 shared task, which adopted a much weaker syntactic parser. In particular, Marcheggiani and Titov (2017) adopted an external syntactic parser with even higher parsing accuracy. Contrarily, our SRL model is based on the automatically predicted parse with moderate performance provided by the CoNLL-2009 shared task, but we still manage to outperform their models. In span SRL, He et al. (2017) injected syntax as a decoding constraint without having to retrain the model and compared the auto and gold syntax information for SRL. Strubell et al. (2018) presented a neural network model in which syntax is incorporated by training one attention head to attend to syntactic parents for each token and demonstrated that the SRL models benefit from injecting state-of-the-art predicted parses. Because different types of syntax and syntactic parsers are used in different works, the results are not directly comparable. Thus, this motivates us to explore how much syntax contributes to dependency-based SRL in the deep learning framework and how to effectively evaluate the relative performance of syntax-based SRL. To this end, we conduct experiments for empirical analysis with different syntactic inputs.

Syntactic Input. In dependency SRL, four types of syntactic input are used to explore the role of syntax:

1. The automatically predicted parse provided by CoNLL-2009 shared task.
2. The parsing results of the CoNLL-2009 data by state-of-the-art syntactic parser, the Biaffine Parser (Dozat and Manning 2017).
3. Corresponding results from another parser, the BIST Parser (Kiperwasser and Goldberg 2016), which is also adopted by Marcheggiani and Titov (2017).
4. The gold syntax available from the official data set.

Besides, to obtain flexible syntactic inputs for research, we design a faulty syntactic tree generator (referred to as STG hereafter) that is able to produce random errors in the output dependency tree like a real parser does. To simplify implementation, we construct a new syntactic tree based on the gold standard parse tree. Given an input error probability distribution estimated from a true parser output, our algorithm presented in Algorithm 2 stochastically modifies the syntactic heads of nodes on the premise of a valid tree. Notably, the "random error" in the STG parse tree is just a simulation of the actual error of a real parser. It does not fully reflect the actual error of the parsers but is used to roughly explore the effect of the relative accuracy of the syntax on the SRL model. In span SRL, because the data set only provides a golden constituent syntax, we compared the auto syntax from the Choe&Charniak Parser (Choe and Charniak 2016), Kitaev&Klein Parser (Kitaev and Klein 2018), and the gold syntax from the data set for SRL models.

Evaluation Measure. For the SRL task, the primary evaluation measure is the semantic labeled F_1 score; however, this score is influenced by the quality of syntactic input to

Algorithm 2 Faulty syntactic tree generator.

Input: A gold standard syntactic tree GT, the specific error probability p
Output: The new generative syntactic tree NT

```
N denotes the number of nodes in GT
for each node n ∈ GT do
    r = random(0, 1), a random number
    if r < p then
        h = random(0, N), a random integer
        find the syntactic head n_h of n in GT
        modify n_h = h, and get a new tree NT
        if NT is a valid tree then
            break
        else
            goto step 5
        end if
    end if
end for
return the new generative tree NT
```

Table 8
Dependency SRL results on the CoNLL 2009 English WSJ test set, in terms of labeled attachment score for syntactic dependencies (LAS), semantic precision (P), semantic recall (R), semantic labeled F_1 score (Sem-F_1), and the ratio Sem-F_1/LAS. A superscript "*" indicates LAS results from our personal communication with the authors.

System		PLM	LAS	P	R	Sem-F_1	Sem-F_1/LAS
(Zhao et al. 2009b)			86.0	–	–	85.4	99.30
(Zhao et al. 2009a)			89.2	–	–	86.2	96.64
(Björkelund et al. 2010)			89.8	87.1	84.5	85.8	95.55
(Lei et al. 2015)			90.4	–	–	86.6	95.80
(FitzGerald et al. 2015)			90.4	–	–	86.7	95.90
(Roth and Lapata 2016)			89.8	88.1	85.3	86.7	96.5
(Marcheggiani and Titov 2017)			90.34*	89.1	86.8	88.0	97.41
Sequence-based + K-order hard pruning	CoNLL09 predicted	+E	86.0	89.7	89.3	89.5	104.07
	STG Auto syntax	+E	90.0	90.5	89.3	89.9	99.89
	Gold syntax	+E	100.0	91.0	89.7	90.3	90.30
Sequence-based + Syntax GCN encoder	CoNLL09 predicted	+E	86.0	90.5	88.5	89.5	104.07
	Biaffine Parser	+E	90.22	90.3	89.3	89.8	99.53
	BIST Parser	+E	90.05	90.3	89.1	89.7	99.61
	Gold syntax	+E	100.0	91.0	90.0	90.5	90.50

some extent, leading to an unfaithful reflection of the competence of a syntax-based SRL system. Namely, this is not the outcome of a true and fair quantitative comparison for these types of SRL models. To normalize the semantic score relative to syntactic parse, we take into account an additional evaluation measure to estimate the actual overall performance of SRL. Here, we use the ratio between labeled F_1 score for semantic dependencies (Sem-F_1) and the LAS for syntactic dependencies proposed by Surdeanu et al. (2008) as the evaluation metric in dependency SRL.[7] The benefits of this measure are two-fold: It quantitatively evaluates syntactic contribution to SRL and impartially estimates the true performance of SRL, independent of the performance of the input syntactic parser. In addition, we further extended this evaluation metric for span SRL and used the ratio of Sem-F_1 and constituent syntax F_1 score (Syn-F_1) to measure the pure contribution of the proposed SRL model to clearly show the source of SRL performance improvement from the model's contribution rather than the improvement due to syntax accuracy. It is worth noting that the ratio of Sem-F1/LAS or Sem-F1/Syn-F1 can only be used as a rough estimate under regular circumstances; that is, it can be used in SRL systems that use sufficiently accurate syntax parses, but if the system uses low-quality syntactic trees with extremely low LAS or Syn-F1, there will be a numerical explosion that renders the value meaningless.

Table 8 reports the dependency SRL performance of existing models[8] in terms of Sem-F_1/LAS ratio on the CoNLL-2009 English test set. Interestingly, even though our system has significantly lower scores than others by 3.8% LAS in syntactic components, we obtain the highest results on both Sem-F_1 and Sem-F_1/LAS ratio. These results show that our SRL component is relatively much stronger. Moreover, the ratio comparison in Table 8 also shows that since the CoNLL-2009 shared task, most SRL works actually benefit from the enhanced syntactic component rather than the improved SRL

7 The idea of the ratio score in Surdeanu et al. (2008) actually was from an author of this paper, Hai Zhao; this was indicated in the acknowledgment part of Surdeanu et al. (2008).

8 Note that several SRL systems that do not provide syntactic information are not listed in the table.

component itself. No post-CoNLL SRL systems, neither traditional nor neural types, exceeded the top systems of the CoNLL-2009 shared task (Zhao et al. 2009b [SRL-only track using the provided predicted syntax] and Zhao et al. 2009a [Joint track using self-developed parser]). We believe that this work, for the first time, reports both a higher Sem-F_1 and a higher Sem-F_1/LAS ratio since the CoNLL-2009 shared task. We also presented the comprehensive results of our sequence-based SRL model with a syntax encoder instead of pruning on the aforementioned syntactic inputs of different quality and compare these with previous SRL models. A number of observations can be made from these results. First, the model with the GCN syntax encoder gives quite stable SRL performance no matter the syntactic input quality, which varies in a broad range, and it obtains overall higher scores compared to the previous states-of-the-art. Second, it is interesting to note that the Sem-F_1/LAS score of our model becomes relatively smaller as the syntactic input becomes better. Not surprisingly though, these results show that our SRL component is relatively even stronger. Third, when we adopt a syntactic parser with higher parsing accuracy, our SRL system achieves a better performance. Notably, our model yields a Sem-F_1 of 90.53% taking gold syntax as input. This suggests that high-quality syntactic parse may indeed enhance SRL, which is consistent with the conclusion in He et al. (2017).

We further evaluated the syntactic contribution in span SRL, as shown in Table 9. For Wang et al. (2019) and our results, when syntax and pre-trained language model were kept the same, our model obtained better Sem-F_1, which is also reflected in the Sem-F_1/Syn-F_1 ratio, indicating that our model has stronger syntactic utilization ability. In addition, by comparing the results of whether a pre-trained language model is used, it can be found that those using the pre-trained language model have a higher ratio of Sem-F_1/Syn-F_1, which shows that the features offered by the pre-trained language model potentially increase the syntactic information, resulting in a higher ratio of syntax contribution.

Additionally, to show how SRL performance varies with syntax accuracy, we also test our sequence-based dependency SRL model with *k*-order hard pruning in first and tenth orders using different erroneous syntactic inputs generated from STG and evaluate their performance using the Sem-F_1/LAS ratio. Figure 12 shows Sem-F_1 scores at different qualities of syntactic parse inputs on the English test set, whose LAS varies from 85% to 100%, compared to previous states-of-the-art (Marcheggiani and Titov 2017). Our tenth-order pruning model gives quite stable SRL performance no matter

Table 9
Span SRL results on the CoNLL 2005 English WSJ test set, in terms of constituent syntax F_1 score (Syn-F_1), semantic precision (P), semantic recall (R), semantic labeled F_1 score (Sem-F_1), and the ratio Sem-F_1/Syn-F_1.

System		PLM	Syn-F_1	P	R	Sem-F_1	Sem-F_1/Syn-F_1
(He et al. 2017)	Choe&Charniak Parser		93.8	–	–	84.8	90.41
	Gold syntax		100.0	–	–	87.0	87.00
(Wang et al. 2019)	Kitaev&Klein Parser	+E	95.4	–	–	88.2	92.45
	Gold syntax	+E	100.0	–	–	92.2	92.20
(Marcheggiani and Titov 2020)	Kitaev&Klein Parser		95.4	85.8	85.1	85.4	89.52
Sequence-based + Syntax GCN encoder	Choe&Charniak Parser		93.8	86.4	84.8	85.6	91.26
	Choe&Charniak Parser	+E	93.8	87.2	86.8	87.0	92.75
	Kitaev&Klein Parser	+E	95.4	88.3	89.1	88.5	92.77
	Gold syntax	+E	100.0	93.2	92.4	92.6	92.60

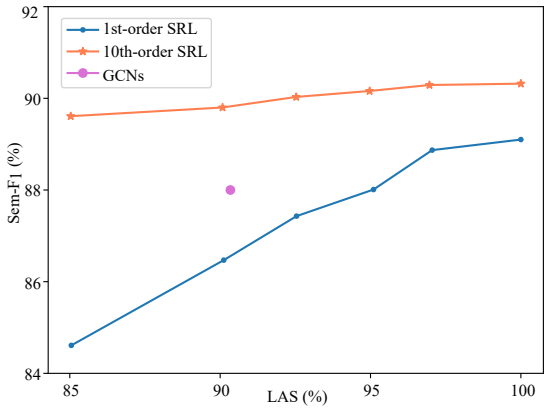


Figure 12
The Sem-F_1 scores of our models with different quality of syntactic inputs vs. original GCNs (Marcheggiani and Titov 2017) on CoNLL-2009 WSJ test set.

the syntactic input quality, which varies in a broad range, while our first-order pruning model yields overall lower results (1–5% F_1 drop), owing to missing too many true arguments. These results show that high-quality syntactic parses may indeed enhance dependency SRL. Furthermore, they indicate that our model with syntactic input as accurate as Marcheggiani and Titov (2017), namely, 90% LAS, will give a Sem-F_1 exceeding 90%.

5.8 Improvement Source for Syntactic Pruning

In this paper, we explore two benefits of using syntax information in SRL. The first is the proximal distribution of arguments in relation to predicates in dependency trees, which allows us to prune unlikely argument candidates. We hypothesize that the pruning process is so effective because of the imbalanced distribution of semantic role labels, which is compounded by the fact that non-arguments are usually used as special arguments in modeling. Pruning primarily eliminates obvious non-arguments and hence reduces this label imbalance to a certain degree. To verify this hypothesis, we compare the F_1 scores for argument labeling using our tree-based models with and without the SynRule Soft

Table 10
The frequency and P / R / F_1 scores for typical argument role labels on the CoNLL-2009 English test set. Baseline refers to the tree-based model, and +Pruning refers to the SynRule Soft Pruning method. Due to the width constraint, we omit the Precision (P) and Recall (R) scores of the nominal pred.

	verbal pred			nominal pred		
Role	FREQ	Baseline	+Pruning	FREQ	Baseline	+Pruning
A0	15%	93.1 / 91.9 / 92.5	93.2 / 92.3 / 92.7	10%	83.3	83.6
A1	21%	93.9 / 93.1 / 93.5	93.6 / 93.4 / 93.5	16%	87.2	87.4
A2	5%	84.3 / 81.8 / 83.0	85.1 / 83.1 / 84.1	7%	81.0	82.8
AM-*	16%	82.2 / 80.2 / 81.2	82.4 / 80.6 / 81.5	5%	75.4	76.3

Table 11
The accuracy on mirror arcs in the dependency trees and semantic dependency graphs predicted using the CoNLL-2009 English test set.

	Sequence-based	Tree-based	Graph-based
Baseline	88.4%	88.7%	89.0%
+Syntax GCN encoder	89.7%	89.9%	89.5%

Pruning. We chose the typical argument labels: A0, A1, A2, and AM-* (such as AM-TMP, AM-LOC) to count their respective frequencies and F_1 according to nominal and verbal predicates.

The quantitative results are shown in Table 10. From the label frequency, we found that the distribution of role labels was uneven. Due to the introduction of the pruning approach, the gain on the low-frequency label is higher than that on high-frequency ones, which suggests that the pruning method reduced the imbalance in the distribution of role labels, which is consistent with our hypothesis.

5.9 Improvement Source of Syntactic Feature

For the second motivation in improving SRL in this paper, we assume that because there are mirrored arcs[9] in the dependency tree and the semantic dependency graph, the use of syntax would increase the accuracy of role labeling compared to that of the baseline model. In order to verify this hypothesis, on the CoNLL-2009 English test set, we computed additional statistics on the predictions of three baselines and these models with the GCN Syntax encoder. In these statistics, we compute the ratio of the number of correctly predicted mirrored arcs to the total number of mirror arcs and exclude arc direction to focus solely on how syntax aids in the predictions on mirror arcs.

The statistical results are presented in Table 11. Comparing the correct ratio of the mirror arcs in the prediction between the baseline and the baseline with an added syntactic encoder, we found that the syntactic information consistently improved these mirrored semantic dependency arcs on the three baselines, which fits our hypothesis. Because these mirrored structures are greatly enhanced in a number of syntactic enhancement models, we can conclude that the GCN encoder effectively encodes syntactic information, and that this encoded syntactic feature is useful for improving the mirrored semantic structures. It is also worth noting that our exploration of the source of the improvement of syntax for SRL does not mean that the syntax has only this effect. Actually, there may be many other factors, which deserve more exploration in the future.

5.10 Effects of Better Syntax on SRL when using Pre-trained Language Models

The results shown in the previous experiments show that with the addition of strong pre-trained language models, the use of syntax information loses its dominance; however, there are two factors that affect the final performance enhancement of syntax on SRL: one is the quality of the syntax parse, and the other is the method of integration. The previous experiments mainly discussed effective integration and corresponding

9 When a dependency edge i→j exists in both the syntactic tree and the semantic graph, the semantic dependency arc i→j is referred to as a mirror arc.

Table 12
The effect of different quality syntactic parse trees on the graph-based SRL model with the enhancement of pre-trained language models. The results are evaluated on the CoNLL-2009 English test set. Notably, the Biaffine Parser was trained by us. Due to the difference in implementation, the parsing performance listed is lower than that of Dozat and Manning (2017).

Parser		w/ ELMo			w/ BERT		
	LAS	P	R	Sem-F_1	P	R	Sem-F_1
N/A	–	89.6	91.2	90.4	92.2	90.7	91.4
CoNLL09 predicted	86.00	91.0	89.5	90.2	91.8	91.0	91.5
Biaffine Parser	90.22	90.5	91.7	91.1	92.1	91.6	91.8
HPSG Parser	94.34	91.7	91.4	91.5	92.1	91.9	92.0

effects but did not pay enough attention to the quality of the syntax. Therefore, in this section, we study how varying the quality of syntactic inputs impacts SRL systems with pre-trained language models. In addition, we use the HPSG Parser, which was trained on the golden syntactic annotations of CoNLL-2009, to provide better quality real parse trees in this experiment. The HPSG parser is a state-of-the-art parser proposed by Zhou and Zhao (2019) and was based on a head-driven phrase structure grammar (HPSG) and the XLNet pre-trained language model. For this experiment, the base model is the graph-based model with the syntax GCN encoder, and the evaluation results are listed in Table 12.

The results show that on one hand, the pre-trained language model greatly improves the performance of SRL, but on the other hand, even with the pre-trained language model, a stronger syntactic parser can still bring performance gains. Using poor quality syntax, however, is unlikely to improve performance and can even have a negative impact on a strong enough SRL baseline.

5.11 Role of Other Syntactic Information

In addition to the syntax tree, part-of-speech tags and lemmas are also viewed as a kind of syntactic knowledge, and they can affect SRL. Since these are typically only used as additional features to enhance representation (which is not the main focus of this work), we conduct a simple experimental exploration on the role of POS tags and lemmas in this section. We picked the sequence-based model[10] with a syntax GCN encoder, conducted experiments on the CoNLL-2009 English data set, and used two pre-trained language models, ELMo and BERT, to investigate their roles under different conditions.

The experimental results are shown in Table 13. From the comparison of the experimental results, we found that with the help of ELMo, both POS tags and lemmas show enhancement effects. POS tags had a greater improvement, which may be because lemmatization produces a more shallow syntactic information. When switching the pre-trained language model from ELMo to the stronger BERT, we found that lemmas almost provide no additional improvement, and the usefulness of POS tags is also very limited. This shows that stronger pre-trained language models may cover shallow syntactic information, and stacking such information will therefore not have additional enhancement effects.

10 The reason for choosing the sequence-based model instead of the graph-based model as in the previous experiment is that POS and Lemma have less obvious influence on graph-based models.

Table 13
Performance comparison of POS tag and lemma on the CoNLL-2009 English test set. Full Model refers to the sequence-based model with syntax GCN encoder, which leverages POS tag, lemma, syntax tree.

System	w/ ELMo			w/ BERT		
	P	R	Sem-F_1	P	R	Sem-F_1
Full Model	90.3	89.3	89.8	91.1	89.5	90.3
-POS	89.8	89.2	89.5	91.0	89.4	90.2
-Lemma	90.1	89.1	89.6	91.1	89.5	90.3

6. Related Work

The origins of semantic role labeling can date back several decades to when Fillmore (1968) first theorized the existence of deep semantic relations between the predicate and other sentential constituents. Over several years, various linguistic formalisms and their related predicate-argument structure inventories have extended Fillmore's seminal intuition (Dowty 1991; Levin 1993). Gildea and Jurafsky (2000) pioneered the task of semantic role labeling as a shallow semantic parsing. In dependency SRL, most traditional SRL models rely heavily on feature templates (Pradhan et al. 2005; Zhao, Chen, and Kit 2009; Björkelund, Hafdell, and Nugues 2009). Among them, Pradhan et al. (2005) combined features derived from different syntactic parses based on SVM classifier, while Zhao, Chen, and Kit (2009) and Zhao, Zhang, and Kit (2013) presented an integrative approach for dependency SRL via a greedy feature selection algorithm. Later, Collobert et al. (2011) proposed a convolutional neural network model that induced word embeddings rather than relied on handcrafted features, which was a breakthrough for the SRL task.

Foland and Martin (2015) presented a dependency semantic role labeler using convolutional and time-domain neural networks, while FitzGerald et al. (2015) exploited neural networks to jointly embed arguments and semantic roles, akin to the work (Lei et al. 2015) that induced a compact feature representation by applying a tensor-based approach. Recently, researchers have considered multiple ways to effectively integrate syntax into SRL learning. Roth and Lapata (2016) introduced dependency path embedding to model syntactic information and exhibited notable success. Marcheggiani and Titov (2017) leveraged the graph convolutional network to incorporate syntax into neural models. Differently, Marcheggiani, Frolov, and Titov (2017) proposed a syntax-agnostic for dependency SRL that used effective word representation, which for the first time achieved performance comparable to state-of-the-art syntax-aware SRL models.

Most neural SRL works, however, often pay less attention to the impact of input syntactic quality on SRL performance since they usually only use a fixed syntactic quality input. This work is thus more than proposing a high performance SRL model through reviewing the highlights of previous models; it also presents an effective syntax tree-based method for argument pruning. Our work is also closely related to Punyakanok, Roth, and Yih (2008). Under traditional methods, Punyakanok, Roth, and Yih (2008) investigated the significance of syntax to SRL systems and showed syntactic information was most crucial in the pruning stage. There are two important differences between Punyakanok, Roth, and Yih (2008) and our work. First, in our paper, we summarize the current dependency and span SRL and consider them under multiple baseline models

and syntax integration approaches to reduce deviations resulting from the model structure and syntax integration approach. Second, the development of pre-trained language models has dramatically changed the basis of SRL models, which motivated us to revisit the role of syntax based on new situations.

As researchers have constantly explored new approaches to further improve SRL, the exploitation of syntactic features has emerged as a natural option. Kasai et al. (2019) extracted the supertags from dependency parses as an additional feature to enhance the SRL model. Cai and Lapata (2019b) presented a system that jointly trained with two syntactic auxiliary tasks: predicting the dependency label of a word and predicting there exists an arc linking said word to the predicate. They also suggested that syntax could help SRL because a significant portion of the predicate-argument relations in the semantic dependency graph mirror the arcs that appear in the syntactic dependency graph, and there is often a deterministic mapping between syntactic and semantic roles. Compared with our research, these works only focus on using syntax with a specific model to improve SRL, whereas instead we performed more systematic and comprehensive research.

In the other span SRL research lines, Moschitti, Pighin, and Basili (2008) applied tree kernels as encoders to extract constituency tree features for SRL, while Naradowsky, Riedel, and Smith (2012) used graphical models to model the tree structures. Socher et al. (2013) and Tai, Socher, and Manning (2015) proposed recursive neural networks for incorporating syntax information in SRL that recursively encoded constituency trees to constituent representations. He et al. (2017) presented an extensive error analysis with deep learning models for span SRL and included a discussion of how constituent syntactic parsers could be used to improve SRL performance. With the recent advent of self-attention, syntax can be used not only for pruning or encoding to provide auxiliary features, but also for guiding the structural learning of models. Strubell et al. (2018) modified the dependency tree structure to train one attention head to attend to syntactic parents for each token. In addition, compared with the application of dependency syntax, the difficult step is mapping the constituent features back to the word level. Wang et al. (2019) extended the syntax linearization approaches of Gómez-Rodríguez and Vilares (2018) and incorporated this information as a word-level feature in a SRL model; Marcheggiani and Titov (2020) introduced a novel neural architecture, SpanGCN, for encoding constituency syntax at the word level.

7. Conclusion

This paper explores the role of syntax for the semantic role labeling task. We presented a systematic survey based on our recent works on SRL and a recently popular pre-trained language modeling. Through experiments on both the dependency and span formalisms, and the sequence-based, tree-based, and graph-based modeling approaches, we conclude that although the effects of syntax on SRL seem like a never-ending topic of research, with the help of current pre-trained language models, the syntax improvement provided to SRL model performance seems to be gradually reaching its upper limit. Beyond presenting approaches that lead to improved SRL performance, we performed a detailed and fair experimental comparison between span and dependency SRL formalisms to show which is more fit for machine learning. In addition, we have studied a variety of methods of syntax integration and have shown that there is unacclimation for the hard pruning in the deep learning model which was very popular in the pre-NN era.

For the exact role of syntax in serving the state-of-the-art SRL models, we have a cautious conclusion: Syntax may still provide downstream task enhancement in the time of deep learning (even with the time of powerful pre-trained language models); however, its returns are diminishing. As we can see from our experimental results, syntactic clues contribute less and less when used with more and more powerful baseline SRL models, and we saw that for our best settings, absolute performance improvement may be just around 0.5%. This is a vast difference from pre-NN times, when the inclusion of syntax could bring an absolute performance improvement of as high as 5% to 10%. Syntax, however, may have alternative methods of integration that do not consist of simply including a pre-trained syntax parser. Some works have demonstrated that the current deep learning models in NLP naturally incorporate syntax using neural models' powerful representation learning. Actually, our recent work (Zhou, Li, and Zhao 2020) showed that syntactic parsers and semantic parsers can help each other, which suggests that in general, neural NLP models, including SRL models, may comprehensively and latently learn linguistic knowledge that covers both syntax and semantics. This may effectively explain why neural SRL models do not rely on syntactic clues as much as their non-NN SRL counterparts. When using even more powerful pre-trained language models for enhancement, we indeed saw even less SRL performance improvement, which is also explainable considering that pre-trained language models have been known to contain syntax in specific hidden layers, as shown in Hewitt and Manning (2019) and Clark et al. (2019). When pre-trained language models offered a strong syntactic signal, including syntax information from a third-party, the pre-trained syntactic parser will certainly lead to weaker improvement, as the added syntax information is somewhat redundant.

Overall, we summarize that there is no certainty that specific syntactic information may surely benefit specific deep SRL models, when the latter may more or less automatically capture syntax from its own task-specific learning. Syntax may still be helpful for deep models in a general way, however, the significance which makes such a help depends on the way of integrating the syntax, but not on the type of models or syntax.

Syntax is an amazing discovery from theoretical linguistics, as it is a complete linguistic concept, but it is not really an observable linguistic phenomena. As Chomsky (1965) addressed, though it was believed that there exists a universal grammar in the human brain, according to his and later inspired research, as to our best knowledge, there is not clear enough evidence from cognitive or electrophysiological outcomes to confirm the existence of such syntax or grammar represented by some kind of human brain structure or cognitive mechanism. While lexical (word) linguistics can be identified using writing control (boundaries between words), and semantics can be noted by mapping words and their corresponding entities in the real world, syntax cannot be experienced like either of the above ways. It is well known that native speakers do not really speak following some perceptive "syntax," yet linguists believe syntax is always there. We speculate that if this is the way humans use syntax when processing language, then, in the future, this may also be how neural NLP models adopt syntax.

Acknowledgments

We thank Kevin Parnow (parnow@sjtu.edu.cn), from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, for his kind help in proofreading when we were working on this paper. Many thanks to the editor and anonymous reviewers for their valuable suggestions and comments.

References

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align

and translate. In *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*, San Diego, CA.

Baker, Collin F., Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90, Montreal.

Berant, Jonathan, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, WA.

Björkelund, Anders, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Coling 2010: Demonstrations*, pages 33–36, Beijing, China.

Björkelund, Anders, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 43–48, Boulder, CO.

Cai, Jiaxun, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntactic-agnostic over syntactic-aware? In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2753–2765, Santa Fe, NM.

Cai, Rui and Mirella Lapata. 2019a. Semi-supervised semantic role labeling with cross-view training. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1018–1027, Hong Kong.

Cai, Rui and Mirella Lapata. 2019b. Syntax-aware semantic role labeling without parsing. *Transactions of the Association for Computational Linguistics*, 7:343–356.

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

Chen, Xinchi, Chunchuan Lyu, and Ivan Titov. 2019. Capturing argument interaction in semantic role labeling with capsule networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5415–5425, Hong Kong.

Choe, Do Kook and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, TX.

Choi, Jinho D. and Martha Palmer. 2011. Transition-based semantic role labeling using predicate argument clustering. In *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*, pages 37–45, Portland, OR.

Chomsky, Noam. 1965. *Aspects of the Theory of Syntax*, volume 11.

Clark, Kevin, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? An analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence.

Collobert, Ronan, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

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

Dowty, David. 1991. Thematic proto-roles and argument selection. *Language*, 67(3):547–619. `https://doi.org/10.2307/415037`, `https://doi.org/10.1353/lan.1991.0021`

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

Fei, Hao, Meishan Zhang, Bobo Li, and Donghong Ji. 2021. End-to-end semantic role labeling with neural transition-based model. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial*

Intelligence, EAAI 2021, Virtual Event, pages 12803–12811, AAAI Press.

Fillmore, C. J. 1968. The case for case. *Universals in Linguistic Theory*. pages 1–9.

FitzGerald, Nicholas, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 960–970, Lisbon.

Foland, William and James Martin. 2015. Dependency-based semantic role labeling using convolutional neural networks. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 279–288, Denver, CO.

Gildea, Daniel and Daniel Jurafsky. 2000. Automatic labeling of semantic roles. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 512–520, Hong Kong.

Gildea, Daniel and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 239–246, Philadelphia, PA.

Gómez-Rodríguez, Carlos and David Vilares. 2018. Constituent parsing as sequence labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1314–1324, Brussels.

Grave, Edouard, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki.

Hajič, Jan, 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 Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, CO.

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

He, Luheng, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver.

He, Shexia, Zuchao Li, and Hai Zhao. 2019. Syntax-aware multilingual semantic role labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5350–5359, Hong Kong.

He, Shexia, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018b. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2061–2071, Melbourne.

Hewitt, John and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, MN.

Hochreiter, Sepp and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780. `https://doi.org/10.1162/neco.1997.9.8.1735`

Johansson, Richard and Pierre Nugues. 2008a. Dependency-based semantic role labeling of PropBank. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 69–78, Honolulu, HI.

Johansson, Richard and Pierre Nugues. 2008b. Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 183–187, Manchester.

Johansson, Richard and Pierre Nugues. 2008c. The effect of syntactic representation on semantic role labeling. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 393–400, Manchester, UK.

Kasai, Jungo, Dan Friedman, Robert Frank, Dragomir Radev, and Owen Rambow.

2019. Syntax-aware neural semantic role labeling with supertags. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 701–709, Minneapolis, MN.

Kingma, Diederik P. and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*, San Diego, CA.

Kiperwasser, Eliyahu and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327. https://doi.org/10.1162/tacl_a_00101

Kipf, Thomas N. and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings*, Toulon.

Kitaev, Nikita and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne.

Lan, Zhenzhong, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020*, Addis Ababa.

Lang, Joel and Mirella Lapata. 2010. Unsupervised induction of semantic roles. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 939–947, Los Angeles, CA.

Lee, Kenton, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen.

Lei, Tao, Yuan Zhang, Lluís Màrquez, Alessandro Moschitti, and Regina Barzilay. 2015. High-order low-rank tensors for semantic role labeling. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1150–1160, Denver, CO.

Levin, Beth. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*, University of Chicago Press.

Li, Zuchao, 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 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2401–2411, Brussels.

Li, Zuchao, Shexia He, Hai Zhao, Yiqing Zhang, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2019a. Dependency or span, end-to-end uniform semantic role labeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6730–6737.

Li, Zuchao, Shexia He, Junru Zhou, Hai Zhao, Kevin Parnow, and Rui Wang. 2019b. Dependency and span, cross-style semantic role labeling on PropBank and NomBank. *arXiv preprint arXiv:1911.02851*.

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

Lin, Yankai, Zhiyuan Liu, and Maosong Sun. 2017. Neural relation extraction with multi-lingual attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 34–43, Vancouver.

Liu, Yinhan, 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. *arXiv preprint arXiv:1907.11692*.

Luong, Thang, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon.

Lyu, Chunchuan, Shay B. Cohen, and Ivan Titov. 2019. Semantic role labeling with iterative structure refinement. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1071–1082, Hong Kong.

Marcheggiani, Diego, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for

dependency-based semantic role labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 411–420, Vancouver.

Marcheggiani, Diego and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, Copenhagen.

Marcheggiani, Diego and Ivan Titov. 2020. Graph convolutions over constituent trees for syntax-aware semantic role labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3915–3928, Online.

Mihaylov, Todor and Anette Frank. 2016. Discourse relation sense classification using cross-argument semantic similarity based on word embeddings. In *Proceedings of the CoNLL-16 Shared Task*, pages 100–107, Berlin.

Moschitti, Alessandro, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224. https://doi.org/10.1162/coli.2008.34.2.193

Mulcaire, Phoebe, Swabha Swayamdipta, and Noah A. Smith. 2018. Polyglot semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 667–672, Melbourne.

Munir, Kashif, Hai Zhao, and Zuchao Li. 2021. Adaptive convolution for semantic role labeling. *IEEE ACM Transactions on Audio, Speech, and Language Processing*, 29:782–791. https://doi.org/10.1109/TASLP.2020.3048665

Nair, Vinod and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, Haifa.

Naradowsky, Jason, Sebastian Riedel, and David Smith. 2012. Improving NLP through marginalization of hidden syntactic structure. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 810–820, Jeju Island.

Ouchi, Hiroki, Hiroyuki Shindo, and Yuji Matsumoto. 2018. A span selection model for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1630–1642, Brussels.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106. https://doi.org/10.1162/0891201053630264

Peng, Hao, Sam Thomson, Swabha Swayamdipta, and Noah A. Smith. 2018. Learning joint semantic parsers from disjoint data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1492–1502, New Orleans, LA.

Pennington, Jeffrey, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha.

Peters, Matthew, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, LA.

Pradhan, Sameer, 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 Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria.

Pradhan, Sameer, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island.

Pradhan, Sameer, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 581–588, Ann Arbor, MI.

Punyakanok, Vasin, Dan Roth, and Wen-tau Yih. 2005. The necessity of syntactic

parsing for semantic role labeling. In *IJCAI*, volume 5, pages 1117–1123. https://doi.org/10.1162/coli.2008.34.2.257

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

Qian, Feng, Lei Sha, Baobao Chang, Lu-chen Liu, and Ming Zhang. 2017. Syntax aware LSTM model for semantic role labeling. In *Proceedings of the 2nd Workshop on Structured Prediction for Natural Language Processing*, pages 27–32, Copenhagen.

Roth, Michael and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1192–1202, Berlin.

Shi, Chen, 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 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2245–2254, Berlin.

Socher, Richard, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, WA.

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

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

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 3104–3112, Montreal.

Swayamdipta, Swabha, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Greedy, joint syntactic-semantic parsing with stack LSTMs. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 187–197, Berlin.

Täckström, Oscar, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41. https://doi.org/10.1162/tacl_a_00120

Tai, Kai Sheng, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing.

Tan, Zhixing, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 4929–4936, AAAI Press.

Toutanova, Kristina, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191. https://doi.org/10.1162/coli.2008.34.2.161

Vinyals, Oriol, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, pages 2773–2781, Montreal.

Wang, Yufei, Mark Johnson, Stephen Wan, Yifang Sun, and Wei Wang. 2019. How to best use syntax in semantic role labelling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5338–5343, Florence.

Xue, Nianwen and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 88–94, Barcelona.

Yang, Bishan and Tom Mitchell. 2017. A joint sequential and relational model for frame-semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, Copenhagen.

Yang, Zhilin, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 5754–5764, Vancouver.

Yih, Wen tau, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206, Berlin.

Zhang, Yue, Rui Wang, and Luo Si. 2019. Syntax-enhanced self-attention-based semantic role labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 616–626, Hong Kong.

Zhao, Hai, 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 Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 61–66, Boulder, CO.

Zhao, Hai, Wenliang Chen, and Chunyu Kit. 2009. Semantic dependency parsing of NomBank and PropBank: An efficient integrated approach via a large-scale feature selection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 30–39, Singapore.

Zhao, Hai, Wenliang Chen, Chunyu Kit, and Guodong Zhou. 2009b. Multilingual dependency learning: A huge feature engineering method to semantic dependency parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 55–60, Boulder, CO.

Zhao, Hai and Chunyu Kit. 2008. Parsing syntactic and semantic dependencies with two single-stage maximum entropy models. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 203–207, Manchester, UK.

Zhao, Hai, Xiaotian Zhang, and Chunyu Kit. 2013. Integrative semantic dependency parsing via efficient large-scale feature selection. *Journal of Artificial Intelligence Research*, 46:203–233.

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

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

Zhou, Junru and Hai Zhao. 2019. Head-Driven Phrase Structure Grammar parsing on Penn Treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence.