 Parsing All: Syntax and Semantics, Dependencies and Spans

Junru Zhou\textsuperscript{1,2,3}, Zuchao Li\textsuperscript{1,2,3}, Hai Zhao\textsuperscript{1,2,3}\textsuperscript{*}

\textsuperscript{1}Department of Computer Science and Engineering, Shanghai Jiao Tong University
\textsuperscript{2}Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, China
\textsuperscript{3}MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University

\{zhoujunru, charlee\}@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn

Abstract

Both syntactic and semantic structures are key linguistic contextual clues, in which parsing the latter has been well shown beneficial from parsing the former. However, few works ever made an attempt to let semantic parsing help syntactic parsing. As linguistic representation formalisms, both syntax and semantics may be represented in either span (constituent/phrase) or dependency, on both of which joint learning was also seldom explored. In this paper, we propose a novel joint model of syntactic and semantic parsing on both span and dependency representations, which incorporates syntactic information effectively in the encoder of neural network and benefits from two representation formalisms in a uniform way. The experiments show that semantics and syntax can benefit each other by optimizing joint objectives. Our single model achieves new state-of-the-art or competitive results on both span and dependency semantic parsing on Propbank benchmarks and both dependency and constituent syntactic parsing on Penn Treebank.

1 Introduction

Second, as semantics is usually considered as a higher layer of linguistics over syntax, most previous studies focus on how the latter helps the former. Though there comes a trend that syntactic clues show less impact on enhancing semantic parsing since neural models were introduced (Marcheggiani and Titov, 2017; Li et al., 2018). In fact, recent works (He et al., 2017; Marcheggiani et al., 2017) propose syntax-agnostic models for semantic parsing and achieve competitive and even state-of-the-art results. However, semantics may not only benefit from syntax which has been well known, but syntax may also benefit from semantics, which is an obvious gap in explicit linguistic structure parsing and few attempts were ever reported. To our best knowledge, only (Shi et al., 2016) ever made a brief attempt on Chinese Semantic Treebank to show the mutual benefits between dependency syntax and semantic roles.

To fill such a gap, in this work, we further exploit both strengths of the span and dependency representation of both semantic role labeling (SRL) (Lewis et al., 2015; Strubell et al., 2018) and syntax, and propose a joint model with multi-task learning (Caruana) in a balanced mode which improves both semantic and syntactic parsing. Moreover, in our model, semantics is learned in an end-to-end way with a uniform representation and syntactic parsing is represented as a \textit{joint span} structure (Zhou and Zhao, 2019) relating to head-driven phrase structure grammar (HPSG) (Pollard and Sag, 1994) which can incorporate both head and phrase information of dependency and constituent syntactic parsing.

We verify the effectiveness and applicability of the proposed model on Propbank semantic parsing \textsuperscript{1} in both span style (CoNLL-2005) (Carreras and Márquez, 2005) and dependency style, (CoNLL-...

\textsuperscript{*} Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100) and Key Projects of National Natural Science Foundation of China (No. U1830222 and No. 61733011).

\textsuperscript{1}It is also called semantic role labeling (SRL) for the semantic parsing task over the Propbank.
Hajić et al., 2009) and Penn Treebank (PTB) (Marcus et al., 1993) for both constituent and dependency syntactic parsing. Our empirical results show that semantics and syntax can indeed benefit each other, and our single model reaches new state-of-the-art or competitive performance for all four tasks: span and dependency SRL, constituent and dependency syntactic parsing.

2 Structure Representation

In this section, we introduce a preprocessing method to handle span and dependency representation, which have strong inherent linguistic relation for both syntax and semantics.

For syntactic representation, we use a formal structure called joint span following (Zhou and Zhao, 2019) to cover both constituent and head information of syntactic tree based on HPSG which is a highly lexicalized, constraint-based grammar (Pollard and Sag, 1994). For semantic (SRL) representation, we propose a unified structure to simplify the training process and employ SRL constraints for span arguments to enforce exact inference.

2.1 Syntactic Representation

The joint span structure which is related to the HEAD FEATURE PRINCIPLE (HFP) of HPSG (Pollard and Sag, 1994) consists of all its children phrases in the constituent tree and all dependency arcs between the head and children in the dependency tree.

For example, in the constituent tree of Figure 1(a), Federal Paper Board is a phrase (1,3) assigned with category NP and in dependency tree, Board is parent of Federal and Paper, thus in our joint span structure, the head of phrase (1,3) is Board. The node $S_H(1,9)$ in Figure 1(b) as a joint span is: $S_H(1,9) = \{ S_H(1,3), S_H(4,8), S_H(9,9), l(1,9,<S>), d(Board,sells), d(.,sells) \}$, where $l(i,j,<S>)$ denotes category of span $i$, $d(r,h)$ indicates the dependency between the word $r$ and its parent $h$. At last, the entire syntactic tree $T$ being a joint span can be represented as:

$$S_H(T) = \{S_H(1,9), d(sells,root)\}^2.$$  

For dependency label of each word, we train a separated multi-class classifier simultaneously with the parser by optimizing the sum of their objectives.

![Figure 1: Constituent, dependency, and joint span structures from (Zhou and Zhao, 2019), which is indexed from 1 to 9 and assigned interval range for each node. The dotted box represents the same part. The special category # is assigned to divide the phrase with multiple heads. Joint span structure contains constitute phrase and dependency arc. Categ in each node represents the category of each constituent, and HEAD indicates the head word.]

Moreover, to simplify the syntactic parsing algorithm, we add a special empty category $\emptyset$ to spans to binarize the n-ary nodes and apply a unary atomic category to deal with the nodes of the unary chain, which is popularly adopted in constituent syntactic parsing (Stern et al., 2017; Gaddy et al., 2018).
2.2 Semantic Representation

Similar to the semantic representation of (Li et al., 2019), we use 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 and dependencies, and \( \mathcal{R} \) is the set of the semantic roles and employ a null label \( \epsilon \) to indicate no relation between predicate-argument pair candidate. The difference from that of (Li et al., 2019) is that in our model, we predict the span and dependency arguments at the same time which needs to distinguish the single word span arguments and dependency arguments. Thus, we represent all the span arguments \( \mathcal{A} = \{(w_i, \ldots, w_j) | 1 \leq i \leq j \leq n\} \) as span \( S(i-1, j) \) and all the dependency arguments \( \mathcal{A} = \{(w_i) | 1 \leq i \leq n\} \) as span \( S(i, i) \). We set a special start token at the beginning of sentence.

3 Our Model

3.1 Overview

As shown in Figure 2, our model includes four modules: token representation, self-attention encoder, scorer module, and two decoders. Using an encoder-decoder backbone, we apply self-attention encoder (Vaswani et al.) that is modified by position partition (Kitaev and Klein, 2018a). We take multi-task learning (MTL) approach (Caruana) sharing the parameters of token representation and self-attention encoder. Since we convert two syntactic representations as joint span structure and apply uniform semantic representation, we only need two decoders, one for syntactic tree based on joint span syntactic parsing algorithm (Zhou and Zhao, 2019), another for uniform SRL.

3.2 Token Representation

In our model, token representation \( x_i \) is composed of characters, words, and part-of-speech (POS) representation. For character-level representation, we use CharLSTM (Ling et al., 2015). For word-level representation, we concatenate randomly initialized and pre-trained word embeddings. We concatenate character representation and word representation as our token representation \( x_i = [x_{\text{char}}; x_{\text{word}}; x_{\text{POS}}] \).

In addition, we also augment our model with ELMo (Peters et al., 2018), BERT (Devlin et al., 2018) or XLNet (Yang et al., 2019) as the sole token representation to compare with other pre-training models. Since BERT and XLNet are based on sub-word, we only take the last sub-word vector of the word in the last layer of BERT or XLNet as our sole token representation \( x_i \).

3.3 Self-Attention Encoder

The encoder in our model is adapted from (Vaswani et al.) and factor explicit content and position information in the self-attention process. The input matrices \( X = [x_1, x_2, \ldots, x_n] \) in which \( x_i \) is concatenated with position embedding are transformed by a self-attention encoder. We factor the model between content and position information both in self-attention sub-layer and feed-forward network, whose setting details follow (Kitaev and Klein, 2018a).

3.4 Scorer Module

Since span and dependency SRL share uniform representation, we only need three types of scores:
syntactic constituent span, syntactic dependency head, and semantic role scores.

We first introduce the span representation $s_{ij}$ for both constituent span and semantic role scores. We define the left end-point vector as concatenation of the adjacent token $\hat{p}_{\ell} = [\hat{y}_i; \hat{y}_{i+1}]$, which $\hat{y}_i$ is constructed by splitting in half the outputs from the self-attention encoder. Similarly, the right end-point vector is $\bar{p}_{\ell} = [\bar{y}_{i+1}; \bar{y}_i]$. Then, the span representation $s_{ij}$ is the differences of the left and right end-point vectors $s_{ij} = [\bar{p}_{\ell} - \hat{p}_{\ell}]$.

**Constituent Span Score** We follow the constituent syntactic parsing (Zhou and Zhao, 2019; Kitaev and Klein, 2018a; Gaddy et al., 2018) to obtain the tree by adding all scores of span $(i, j) = \max_\ell \sum_s S(i, j, \ell)$. The individual score of category $\ell$ is denoted by $S_{\text{categ}}(i, j, \ell) = [S(i, j)]_{\ell}$, where $[\cdot]_{\ell}$ indicates the value of corresponding the l-th element of the score vector. The score $s(T)$ of the constituent parse tree $T$ is obtained by adding all scores of span $(i, j)$ with category $\ell$: $s(T) = \sum_{(i, j, \ell) \in T} S_{\text{categ}}(i, j, \ell)$.

The goal of constituent syntactic parsing is to find the tree with the highest score: $T = \arg\max_T s(T)$. We use CKY-style algorithm (Gaddy et al., 2018) to obtain the tree $T$ in $O(n^3)$ time complexity. This structured prediction problem is handled with satisfying the margin constraint:

$s(T^*) \geq s(T) + \Delta(T, T^*)$,

where $T^*$ denotes correct parse tree, and $\Delta$ is the Hamming loss on constituent spans with a slight modification during the dynamic programming search. The objective function is the hinge loss,

$J_1(\theta) = \max(0, \max_T [s(T) + \Delta(T, T^*)] - s(T^*))$.

**Dependency Head Score** We predict a distribution over the possible head for each word and use the biaffine attention mechanism (Dozat and Manning, 2017) to calculate the score as follow:

$\alpha_{ij} = h_i^T W g_j + U^T h_i + V^T g_j + b$,

where $\alpha_{ij}$ indicates the child-parent score, $W$ denotes the weight matrix of the bi-linear term, $U$ and $V$ are the weight vectors of the linear term, and $b$ is the bias item, $h_i$ and $g_i$ are calculated by a distinct one-layer perceptron network.

We minimize the negative log-likelihood of the golden dependency tree $Y$, which is implemented as a cross-entropy loss:

$J_2(\theta) = -\lambda \log P_\theta(h_i|x_i) + \lambda \log P_\theta(l_i|x_i, h_i)$,

where $P_\theta(h_i|x_i)$ is the probability of correct parent node $h_i$ for $x_i$, and $P_\theta(l_i|x_i, h_i)$ is the probability of the correct dependency label $l_i$ for the child-parent pair ($x_i, h_i$).

**Semantic Role Score** To distinguish the currently considered predicate from its candidate arguments in the context, we employ one-layer perceptron to contextualized representation for argument $a_{ij}$ candidates:

$a_{ij} = g(W_3 s_{ij} + b_1)$,

where $g$ is the Rectified Linear Unit nonlinearity and $s_{ij}$ denotes span representation.

And predicate candidates $p_k$ is simply represented by the outputs from the self-attention encoder: $p_k = y_k$.

For semantic role, different from (Li et al., 2019), we simply adopt concatenation of predicates and arguments representations, and one-layer feedforward networks to generate semantic role score:

$\Phi_r(p, a) = W_5 g(LN(W_4[p_k; a_{ij}] + b_4)) + b_5$,

and the individual score of semantic role label $r$ is denoted by: $\Phi_r(p, a, r) = \Phi_r(p, a) | r$.

Since the total of predicate-argument pairs are $O(n^3)$, which is computationally impractical. We apply candidates pruning method in (Li et al., 2019; He et al., 2018a). First of all, we train separated scorers ($\phi_p$ and $\phi_a$) for predicates and arguments by two one-layer feedforward networks. Then, the predicate and argument candidates are ranked according to their predicted score ($\phi_p$ and $\phi_a$), and we select the top $n_p$ and $n_a$ predicate and argument candidates, respectively:

$n_p = \min(\lambda_p n_p, m_p)$, $n_a = \min(\lambda_a n_a, m_a)$,

where $\lambda_p$ and $\lambda_a$ are pruning rate, and $m_p$ and $m_a$ are maximal numbers of candidates.

Finally, the semantic role scorer is trained to optimize the probability $P_\theta(y|s)$ of the predicate-argument-relation tuples $y_{(p,a,r)} \in Y$ given the
Algorithm 1 Joint span syntactic parsing algorithm

**Input:** sentence length \( n \), span and dependency score \( s(i, j, \ell) \), \( d(r, h) \), \( 1 \leq i \leq j \leq n \), \( r, h, \ell \)

**Output:** maximum value \( S_H(T) \) of tree \( T \)

**Initialization:**
\[
s_t[i][j][h] = s[i][j][h] = 0, \forall i, j, h
\]

**for** \( \text{len} = 1 \) to \( n \) \n
**for** \( i = 1 \) to \( n - \text{len} \) \n
\[ j = i + \text{len} - 1 \]

**if** \( \text{len} = 1 \) \n
\[ s_t[i][j][i] = s_t[i][j][i] = \max s(i, j, \ell) \]

**else**

**for** \( h = i \) to \( j \) \n
\[ \text{split}_t = \max_{i \leq r < h} \{ \max_{r < k < h} \{ s_t[i][k][r] + \}
\]

\[ s_t[k + 1][j][h] \} + d(r, h) \}

\[ \text{split}_r = \max_{k \leq r \leq j} \{ \max_{h \leq k < r} \{ s_t[i][k][h] + \}
\]

\[ s_t[k + 1][j][h] \} + d(r, h) \}

\[ s_t[i][j][h] = \max \{ \text{split}_t, \text{split}_r \} + \max_{\ell \in \mathbb{R}} s(i, j, \ell) \]

**end for**

**end if**

**end for**

\[ S_H(T) = \max_{1 \leq h \leq n} \{ s_t[1][n][h] + d(h, \text{root}) \} \]

---

Sentence \( s \), which can be factorized as:

\[
J_3(\theta) = \sum_{p, a, r \in A, r \in R} -\log P_\theta(y(p, a, r) | s)
\]

\[
= \sum_{p, a, r \in A, r \in R} -\log \exp \phi(p, a, r) / \sum_{\hat{r} \in R} \exp \phi(p, a, \hat{r})
\]

where \( \theta \) represents the model parameters, and \( \phi(p, a, r) = \phi_p + \phi_a + \Phi_r(p, a, r) \) is the score by the predicate-argument-relation tuple including predicate score \( \phi_p \), argument score \( \phi_a \) and semantic role label score \( \Phi_r(p, a, r) \). In addition, we fix the score of null label \( \phi(p, a, e) = 0 \).

At last, we train our scorer for simply minimizing the overall loss:

\[ J_{\text{overall}}(\theta) = J_1(\theta) + J_2(\theta) + J_3(\theta). \]

### 3.5 Decoder Module

**Decoder for Joint Span Syntax**

As the joint span is defined in a recursive way, to score the root joint span has been equally scoring all spans and dependencies in syntactic tree.

During testing, we apply the *joint span CKY-style algorithm* (Zhou and Zhao, 2019), as shown in Algorithm 1 to explicitly find the globally highest score \( S_H(T) \) of our joint span syntactic tree \( T \).

Also, to control the effect of combining span and dependency scores, we apply a weight \( \lambda_H \)

\[
s(i, j, \ell) = \lambda_H S_{\text{catag}}(i, j, \ell), d(i, j) = (1-\lambda_H) \alpha_{ij},
\]

where \( \lambda_H \) in the range of 0 to 1. In addition, we can merely generate constituent or dependency syntactic parsing tree by setting \( \lambda_H \) to 1 or 0, respectively.

**Decoder for Uniform Semantic Role** Since we apply uniform span for both dependency and span semantic role, we use a single dynamic programming decoder to generate two semantic forms following the non-overlapping constraints: span semantic arguments for the same predicate do not overlap (Punyakanok et al., 2008).

### 4 Experiments

We evaluate our model on CoNLL-2009 shared task (Hajič et al., 2009) for dependency-style SRL, CoNLL-2005 shared task (Carreras and Márquez, 2005) for span-style SRL both using the Propbank convention (Palmer et al., 2005), and English Penn Treebank (PTB) (Marcus et al., 1993) for constituent syntactic parsing, Stanford basic dependencies (SD) representation (Marneffé et al., 2006) converted by the Stanford parser for dependency syntactic parsing. We follow standard data splitting: semantic (SRL) and syntactic parsing take section 2-21 of Wall Street Journal (WSJ) data as training set, SRL takes section 24 as development set while syntactic parsing takes section 22 as development set, SRL takes section 23 of WSJ together with 3 sections from Brown corpus as test set while syntactic parsing only takes section 23. POS tags are predicted using the Stanford tagger (Toutanova et al., 2003). In addition, we use two SRL setups: end-to-end and pre-identified predicates.

For the predicate disambiguation task in dependency SRL, we follow (Marcheggiani and Titov, 2017) and use the off-the-shelf disambiguator

\[ \text{http://nlp.stanford.edu/software/lex-parser.html} \]

\[ \text{http://nlp.stanford.edu/software/lex-parser.html} \]

\[ \text{http://nlp.stanford.edu/software/lex-parser.html} \]
| Model                  | F1  | UAS | LAS |
|------------------------|-----|-----|-----|
| separate constituent   | 93.98 | 95.38 | 94.06 |
| converted dependency   | —   | 95.80 | 94.40 |
| separate dependency    | 95.38 | 94.06 | —   |
| joint span \(\lambda_H = 1.0\) | 93.89 | —   | —   |
| joint span \(\lambda_H = 0.0\) | —   | 95.90 | 94.50 |
| joint span \(\lambda_H = 0.8\) | 93.98 | 95.99 | 94.53 |
| converted dependency   | —   | 95.70 | 94.60 |

Table 1: PTB dev set performance of joint span syntactic parsing. The converted means the corresponding dependency syntactic parsing results are from the corresponding constituent parse tree using head rules.

From (Roth and Lapata, 2016). For constituent syntactic parsing, we use the standard evalb tool to evaluate the F1 score. For dependency syntactic parsing, following previous work (Dozat and Manning, 2017), we report the results without punctuations of the labeled and unlabeled attachment scores (LAS, UAS).

4.1 Setup

**Hyperparameters** In our experiments, we use 100D GloVe (Pennington et al., 2014) pre-train embeddings. For the self-attention encoder, we set 12 self-attention layers and use the same other hyperparameters settings as (Kitaev and Klein, 2018a). For semantic role scorer, we use 512-dimensional MLP layers and 256-dimensional feed-forward networks. For candidates pruning, we set \(\lambda_p = 0.4\) and \(\lambda_a = 0.6\) for pruning predicates and arguments, \(m_p = 30\) and \(m_a = 300\) for max numbers of predicates and arguments respectively. For constituent span scorer, we apply a hidden size of 250-dimensional feed-forward networks. For dependency head scorer, we employ two 1024-dimensional MLP layers with the ReLU as the activation function for learning specific representation and a 1024-dimensional parameter matrix for biaffine attention.

In addition, when augmenting our model with ELMo, BERT and XLNet, we set 4 layers of self-attention for ELMo and 2 layers of self-attention for BERT and XLNet.

**Training Details** we use 0.33 dropout for biaffine attention and MLP layers. All models are trained for up to 150 epochs with batch size 150 on a single NVIDIA GeForce GTX 1080Ti GPU with Intel i7-7800X CPU. We use the same training settings as (Kitaev and Klein, 2018a) and (Kitaev and Klein, 2018b).

4.2 Joint Span Syntactic Parsing

This subsection examines joint span syntactic parsing decoder 3.5 with semantic parsing both of dependency and span. The weight parameter \(\lambda_H\) plays an important role to balance the syntactic span and dependency scores. When \(\lambda_H\) is set to 0 or 1, the joint span parser works as the dependency-only parser or constituent-only parser respectively. \(\lambda_H\) set to between 0 to 1 indicates the general joint span syntactic parsing, providing both constituent and dependency structure prediction. We set the \(\lambda_H\) parameter from 0 to 1 increased by 0.1 step as shown in Figure 3. The best results come out when \(\lambda_H\) is set to 0.8 which achieves the best performance of both syntactic parsing.

In addition, we compare the joint span syntactic parsing decoder with a separate learning constituent syntactic parsing model which takes the same token representation, self-attention encoder and joint learning setting of semantic parsing on PTB dev set. The constituent syntactic parsing results are also converted into dependency ones by PTB-SD for comparison.

Table 1 shows that joint span decoder benefit both of constituent and dependency syntactic parsing. Besides, the comparison also shows that the directly predicted dependencies from our model are better than those converted from the predicted constituent parse trees in UAS term.

4.3 Joint Learning Analysis

Table 2 compares the different joint setting of semantic (SRL) and syntactic parsing to examine
whether semantics and syntax can enjoy their joint learning.

In the end-to-end mode, we find that constituent syntactic parsing can boost both styles of semantics while dependency syntactic parsing cannot. Moreover, the results of the last two rows indicate that semantics can benefit syntax simply by optimizing the joint objectives. While in the given predicate mode, both constituent and dependency syntactic parsing can enhance SRL. In addition, joint learning of our uniform SRL performs better than separate learning of either dependency or span SRL in both modes.

Overall, joint semantic and constituent syntactic parsing achieve relatively better SRL results than the other settings. Thus, the rest of the experiments are done with multi-task learning of semantics and constituent syntactic parsing (wo/dep). Since semantics benefits both of two syntactic formalisms, we also compare the results of joint learning with semantics and two syntactic parsing models (w/dep).

### 4.4 Syntactic Parsing Results

In the wo/dep setting, we convert constituent syntactic parsing results into dependency ones by PTB-SD for comparison and set λ_H described in 3.5 to 1 for generating constituent syntactic parsing only.

Compared to the existing state-of-the-art models without pre-training, our performance exceeds (Zhou and Zhao, 2019) nearly 0.2 in LAS of dependency and 0.3 F1 of constituent syntactic parsing which are considerable improvements on such strong baselines. Compared with (Strubell et al., 2018) shows that our joint model setting boosts

| System                  | SEM_{span} | SEM_{dep} | SYN_{span} | SYN_{dep} | F1 | F1 | F1 | UAS | LAS |
|-------------------------|------------|-----------|------------|-----------|----|----|----|-----|-----|
| End-to-end              |            | 82.27     |            |            |    |    |    |     |     |
| SEM_{span}              | 82.27      |          |            |            |    |    |    |     |     |
| SEM_{dep}               | 82.50      | 84.92     |            |            |    |    |    |     |     |
| SEM_{span}, SEM_{dep}   | 83.81      | 84.95     | 93.98      |            |    |    |    |     |     |
| SEM_{span}, SYN_{span}  | 83.13      | 84.24     | 95.80      | 94.40     |    |    |    |     |     |
| SEM_{span}, SYN_{dep}   | 83.12      | 83.90     | 93.98      | 95.95     | 94.51|

Table 2: Joint learning analysis on CoNLL-2005, CoNLL-2009, and PTB dev sets.

### 4.5 Semantic Parsing Results

We present all results using the official evaluation presentation to compare with other models. Our version of BERT and XLNet as the sole token representation to compare with other models. Our single model in XLNet setting achieving 96.18 F1 score of constituent syntactic parsing, 97.23% UAS and 95.65% LAS of dependency syntactic parsing.

### 4.6 Table 3: Dependency syntactic parsing on WSJ test set

| System                  | LR | LP | F1 |
|-------------------------|----|----|----|
| (Gaddy et al., 2018)    | 91.76 | 92.41 | 92.08 |
| (Stern et al., 2017)    | 92.57 | 92.56 | 92.56 |
| (Kitaev and Klein, 2018a) | 93.20 | 93.90 | 93.55 |
| (Zhou and Zhao, 2019)   | 93.64 | 93.92 | 93.78 |
| Ours (wo/dep)           | 95.36 | 94.01 | 94.79 |
| Ours (w/dep)            | 96.90 | 95.32 | 95.52 |

Table 4: Constituent syntactic parsing on WSJ test set

both of syntactic parsing and SRL which are consistent with (Shi et al., 2016) that syntactic parsing and SRL benefit relatively more from each other.

We augment our parser with ELMo, a larger version of BERT and XLNet as the sole token representation to compare with other models. Our single model in XLNet setting achieving 96.18 F1 score of constituent syntactic parsing, 97.23% UAS and 95.65% LAS of dependency syntactic parsing.

### 4.5 Semantic Parsing Results

We present all results using the official evaluation script from the CoNLL-2005 and CoNLL-2009 shared tasks, and compare our model with previous state-of-the-art models in Table 5, 6. The upper part of the tables presents results from end-to-end mode while the lower part shows the results of given predicate mode to compare to more previous works with pre-identified predicates. In

| System                  | UAS | LAS |
|-------------------------|-----|-----|
| (Dozat and Manning, 2017) | 95.74 | 94.08 |
| (Ma et al., 2018)       | 95.87 | 94.19 |
| (Strubell et al., 2018) | 94.92 | 91.87 |
| (Fernández-González and Gómez-Rodríguez, 2019) | 96.04 | 94.43 |
| (Zhou and Zhao, 2019)   | 96.09 | 94.68 |
| Ours (wo/dep)           | 95.20 | 93.90 |
| Ours (w/dep)            | 96.15 | 94.85 |

Table 3: Dependency syntactic parsing on WSJ test set.
given predicate mode, we simply replace predicate candidates with the gold predicates without other modification on the input or encoder.

**Span SRL Results**  Table 5 shows results on CoNLL-2005 in-domain (WSJ) and out-domain (Brown) test sets. It is worth noting that (Strubell et al., 2018) injects state-of-the-art predicted parses in terms of setting of (Dozat and Manning, 2017) at test time and aims to use syntactic information to help SRL. While our model not only excludes other auxiliary information during test time but also benefits both syntax and semantics. We obtain comparable results with the latest state-of-the-art model (Strubell et al., 2018) and outperform all recent models without additional information in test time.

After incorporating with pre-training contextual representations, our model achieves new state-of-the-art both of end-to-end and given predicate mode and both of in-domain and out-domain text. 

**Dependency SRL Results**  Table 6 presents the results on CoNLL-2009. We obtain new state-of-the-art both of end-to-end and given predicate mode and both of in-domain and out-domain text. These results demonstrate that our improved uniform SRL representation can be adapted to perform dependency SRL and achieves impressive performance gains.

### 5 Related Work

In the early work of SRL, most of the researchers focus on feature engineering based on training corpus. The traditional approaches to SRL focused on developing rich sets of linguistic features templates and then employ linear classifiers such as SVM (Zhao et al., 2009). Recently, especially with the impressive success of neural networks, considerable attention has been paid to syntactic features (Strubell et al., 2018; Kasai et al., 2019; He et al., 2018b; Li et al., 2018b).

Besides, both span and dependency are effective formal representations for both semantics and syntax. On one hand, researchers are interested in two forms of SRL models that may benefit from each other rather than their separated development, which has been roughly discussed in (Johansson and Nugues, 2008). (He et al., 2018a) is the first to apply span-graph structure based on contextualized span representations to span SRL.
and (Li et al., 2019) built on these span representations achieves state-of-art results on both span and dependency SRL using the same model but training individually.

On the other hand, researchers have discussed how to encode lexical dependencies in phrase structures, like lexicalized tree adjoining grammar (LTAG) (Schabes et al., 1988) and head-driven phrase structure grammar (HPSG) (Pollard and Sag, 1994) which is a constraint-based highly lexicalized non-derivational generative grammar framework.

6 Conclusions

This paper presents the first joint learning model which is evaluated on four tasks: span and dependency SRL, constituent and dependency syntactic parsing. We exploit the relationship between semantics and syntax and conclude that not only syntax can help semantics but also semantics can improve syntax performance. Besides, we propose two structure representations, uniform SRL and joint span of syntactic structure, to combine the span and dependency forms. From experiments on these four parsing tasks, our single model achieves state-of-the-art or competitive results.

References

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A Full End-to-End Semantic Role Labeler, Syntactic-agnostic Over Syntactic-aware? In COLING.

Xavier Carreras and Lluís Márquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In CoNLL.

Richard A. Caruana. Multitask Learning: A Knowledge-Based Source of Inductive Bias. Machine Learning Proceedings, 10(1).

N. Chomsky. 1981. Lectures on government and binding.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. CoRR, abs/1810.04805.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. arXiv preprint arXiv:1611.01734.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2019. Left-to-right dependency parsing with pointer networks. abs/1903.08445.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What’s Going On in Neural Constituency Parsers? An Analysis. In NAACL.

Jan Hajic, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antónia Martí, Luíís Márquez, 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 CoNLL.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018a. Jointly Predicting Predicates and Arguments in Neural Semantic Role Labeling. In ACL.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep Semantic Role Labeling: What Works and What’s Next. In ACL.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018b. Syntax for Semantic Role Labeling, To Be, Or Not To Be. In ACL.

Richard Johansson and Pierre Nugues. 2008. Dependency-based Semantic Role Labeling of PropBank. In EMNLP.

Jungo Kasai, Dan Friedman, Robert Frank, Dragomir R. Radev, and Owen Rambow. 2019. Syntax-aware neural semantic role labeling with supertags. abs/1903.05260.

Nikita Kitaev and Dan Klein. 2018a. Constituency Parsing with a Self-Attentive Encoder. In ACL.

Nikita Kitaev and Dan Klein. 2018b. Multilingual Constituency Parsing with Self-Attention and Pre-Training. arXiv preprint arXiv:1812.11760.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint A* CCG Parsing and Semantic Role Labelling. In EMNLP.

Zuchao Li, Shexia He, Jiaxun Cai, Zhuosheng Zhang, Hai Zhao, Gongshen Liu, Linlin Li, and Luo Si. 2018. A Unified Syntax-aware Framework for Semantic Role Labeling. In EMNLP.

Zuchao Li, Shexia He, Hai Zhao, Yiqing Zhang, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2019. Dependency or span, end-to-end uniform semantic role labeling. volume abs/1901.05280.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramón Fernández, Silvio Amir, Luís Marujo, and Tiago Luís. 2015. Finding Function in Form: Compositional Character Models for Open Vocabularly Word Representation. In EMNLP.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-Pointer Networks for Dependency Parsing. In ACL.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A Simple and Accurate Syntax-Agnostic Neural Model for Dependency-based Semantic Role Labeling. In CoNLL.
Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In EMNLP.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. Computational Linguistics, 19(2).

M. Marneffe, B. Maccartney, and C. Manning. 2006. Generating typed dependency parses from phrase structure parses. In IN PROC. INTL CONF. ON LANGUAGE RESOURCES AND EVALUATION (LREC).

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2018. A Span Selection Model for Semantic Role Labeling. In EMNLP.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. American Journal of Computational Linguistics, 31(1).

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In EMNLP.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In NAACL.

Carl Pollard and Ivan A Sag. 1994. Head-Driven Phrase Structure Grammar. University of Chicago Press.

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

Michael Roth and Mirella Lapata. 2016. Neural Semantic Role Labeling with Dependency Path Embeddings. In ACL.

Yves Schabes, Anne Abeille, and Aravind K. Joshi. 1988. Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars. In COLING.

Peng Shi, Zhiyang Teng, and Yue Zhang. 2016. Exploiting Mutual Benefits between Syntax and Semantic Roles using Neural Network. In EMNLP.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017. Effective Inference for Generative Neural Parsing. In EMNLP, pages 1695–1700.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. In EMNLP.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep Semantic Role Labeling with Self-Attention.