A Span Selection Model for Semantic Role Labeling

Hiroki Ouchi[2,3] **Hiroyuki Shindo**[1,2] **Yuji Matsumoto**[1,2]
[1] Nara Institute of Science and Technology
[2] RIKEN Center for Advanced Intelligence Project (AIP)
[3] Tohoku University
hiroki.ouchi@riken.jp, { shindo, matsu }@is.naist.jp

Abstract

We present a simple and accurate span-based model for semantic role labeling (SRL). Our model directly takes into account all possible argument spans and scores them for each label. At decoding time, we greedily select higher scoring labeled spans. One advantage of our model is to allow us to design and use span-level features, that are difficult to use in token-based BIO tagging approaches. Experimental results demonstrate that our ensemble model achieves the state-of-the-art results, 87.4 F1 and 87.0 F1 on the CoNLL-2005 and 2012 datasets, respectively.

1 Introduction

Semantic Role Labeling (SRL) is a shallow semantic parsing task whose goal is to recognize the predicate-argument structure of each predicate. Given a sentence and a target predicate, SRL systems have to predict semantic arguments of the predicate. Each argument is a *span*, a unit that consists of one or more words. A key to the argument span prediction is how to represent and model spans.

One popular approach to it is based on BIO tagging schemes. State-of-the-art neural SRL models adopt this approach (Zhou and Xu, 2015; He et al., 2017; Tan et al., 2018). Using features induced by neural networks, they predict a BIO tag for each word. Words at the beginning and inside of argument spans have the "B" and "I" tags, and words outside argument spans have the tag "O." While yielding high accuracies, this approach reconstructs argument spans from the predicted BIO tags instead of directly predicting the spans.

Another approach is based on labeled span prediction (Täckström et al., 2015; FitzGerald et al., 2015). This approach scores each span with its label. One advantage of this approach is to allow us to design and use span-level features, that are difficult to use in BIO tagging approaches. However, the performance has lagged behind that of the state-of-the-art BIO-based neural models.

To fill this gap, this paper presents a simple and accurate span-based model. Inspired by recent span-based models in syntactic parsing and coreference resolution (Stern et al., 2017; Lee et al., 2017), our model directly scores all possible labeled spans based on span representations induced from neural networks. At decoding time, we greedily select higher scoring labeled spans. The model parameters are learned by optimizing log-likelihood of correct labeled spans.

We evaluate the performance of our span-based model on the CoNLL-2005 and 2012 datasets (Carreras and Màrquez, 2005; Pradhan et al., 2012). Experimental results show that the span-based model outperforms the BiLSTM-CRF model. In addition, by using contextualized word representations, ELMo (Peters et al., 2018), our ensemble model achieves the state-of-the-art results, 87.4 F1 and 87.0 F1 on the CoNLL-2005 and 2012 datasets, respectively. Empirical analysis on these results shows that the label prediction ability of our span-based model is better than that of the CRF-based model. Another finding is that ELMo improves the model performance for span boundary identification.

In summary, our main contributions include:

- A simple span-based model that achieves the state-of-the-art results.
- Quantitative and qualitative analysis on strengths and weaknesses of the span-based model.
- Empirical analysis on the performance gains by ELMo.

Our code and scripts are publicly available.[1]

[1] https://github.com/hiroki13/span-based-srl.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1630–1642
Brussels, Belgium, October 31 - November 4, 2018.

2 Model

We treat SRL as *span selection*, in which we select appropriate spans from a set of possible spans for each label. This section formalizes the problem and provides our span selection model.

2.1 Span Selection Problem

Problem Setting

Given a sentence that consists of T words $w_{1:T} = w_1, \cdots, w_T$ and the target predicate position index p, the goal is to predict a set of labeled spans $Y = \{\langle i, j, r\rangle_k\}_{k=1}^{|Y|}$.

$$\textbf{Input}: X = \{w_{1:T}, p\},$$
$$\textbf{Output}: Y = \{\langle i, j, r\rangle_k\}_{k=1}^{|Y|} \ .$$

Each labeled span $\langle i, j, r\rangle$ consists of word indices i and j in the sentence ($1 \leq i \leq j \leq T$) and a semantic role label $r \in \mathcal{R}$.

One simple method to predict Y is to select the highest scoring span (i, j) from all possible spans $\mathcal{S}$ for each label r,

$$\underset{(i,j)\in\mathcal{S}}{\text{argmax}} \ \textsc{Score}_r(i, j), \ r \in \mathcal{R} \ . \qquad (1)$$

Function $\textsc{Score}_r(i, j)$ returns a real value for each span $(i, j) \in \mathcal{S}$ (described in Section 2.2 in more detail). The number of possible spans $\mathcal{S}$ in the input sentence $w_{1:T}$ is $\frac{T(T+1)}{2}$, and $\mathcal{S}$ is defined as follows,

$$\mathcal{S} = \{(i, j) \mid i, j \in \{1, \cdots, T\}, i \leq j\} \ .$$

Note that some semantic roles may not appear in the sentence. To deal with the absence of some labels, we define the predicate position span (p, p) as a NULL span and train a model to select the NULL span when there is no span for the label.[2]

Example

Consider the following sentence with the set of correct labeled spans Y.

$$\begin{array}{cccc} \text{She}_1 & \underline{\text{kept}}_2 & \text{a}_3 & \text{cat}_4 \\ [\ \texttt{A0}\ ] & & [\quad \texttt{A1}\quad] & \end{array}$$

$$Y = \{\ \langle 1, 1, \texttt{A0}\rangle,\ \langle 3, 4, \texttt{A1}\rangle,$$
$$\langle 2, 2, \texttt{A2}\rangle, \cdots, \langle 2, 2, \texttt{TMP}\rangle\ \}$$

[2] Since the predicate itself can never be an argument of its own, we define the position as the NULL span.

The input sentence is $w_{1:4}$ = "She kept a cat", and the target predicate position is $p = 2$. The correct labeled span $\langle 1, 1, \texttt{A0}\rangle$ indicates that the A0 argument is "She", and $\langle 3, 4, \texttt{A1}\rangle$ indicates that the A1 argument is "a cat". The other labeled spans $\langle 2, 2, *\rangle$ indicate there are no arguments.

All the possible spans in this sentence are as follows,

$$\mathcal{S}_{w_{1:4}} = \{(1, 1), (1, 2), (1, 3), (1, 4), (2, 2),$$
$$(2, 3), (2, 4), (3, 3), (3, 4), (4, 4)\} \ ,$$

where the predicate span $(2, 2)$ is treated as the NULL span. Among these candidates, we select the highest scoring span for each label. As a result, we can obtain correct labeled spans Y.

2.2 Scoring Function

As the scoring function for each span in Eq. 1, we model normalized distribution over all possible spans $\mathcal{S}$ for each label r,

$$\begin{aligned} \textsc{Score}_r(i, j) &= \mathrm{P}_\theta(i, j \mid r) \\ &= \frac{\exp(\mathrm{F}_\theta(i, j, r))}{\sum_{(i',j')\in\mathcal{S}} \exp(\mathrm{F}_\theta(i', j', r))} \ , \end{aligned} \qquad (2)$$

where function F_θ returns a real value.

We train the parameters θ of F_θ on a training set,

$$\mathcal{D} = \{(X^{(n)}, Y^{(n)})\}_{n=1}^{|\mathcal{D}|} \ ,$$
$$X = \{w_{1:T}, p\} \ ,$$
$$Y = \{\langle i, j, r\rangle_k\}_{k=1}^{|Y|} \ .$$

To train the parameters θ of F_θ, we minimize the cross-entropy loss function,

$$\mathcal{L}(\theta) = \sum_{(X,Y)\in\mathcal{D}} \ell_\theta(X, Y) \ , \qquad (3)$$
$$\ell_\theta(X, Y) = \sum_{\langle i,j,r\rangle\in Y} \log \mathrm{P}_\theta(i, j|r) \ ,$$

where function $\ell_\theta(X, Y)$ is a loss for each sample.

2.3 Function F_θ

Function F_θ in Eq. 2 consists of three types of functions; the base feature function f_{base}, the span feature function f_{span} and the labeling function f_{label} as follows,

$$\mathbf{h}_{1:T} = f_{base}(w_{1:T}, p) \ , \qquad (4)$$
$$\mathbf{h}_s = f_{span}(\mathbf{h}_{1:T}, s) \ , \qquad (5)$$
$$\mathrm{F}_\theta(i, j, r) = f_{label}(\mathbf{h}_s, r) \ . \qquad (6)$$

Firstly, f_{base} calculates a base feature vector $\mathbf{h}_t$ for each word $w_t \in w_{1:T}$. Then, from a sequence of the base feature vectors $\mathbf{h}_{1:T}$, f_{span} calculates a span feature vector $\mathbf{h}_s$ for a span $s = (i, j)$. Finally, using $\mathbf{h}_s$, f_{label} calculates the score for the span $s = (i, j)$ with a label r.

Each function in Eqs. 4, 5 and 6 can arbitrarily be defined. In Section 3, we describe our functions used in this paper.

2.4 Inference

The simple argmax inference (Eq. 1) selects one span for each label. While this argmax inference is computationally efficient, it faces the following two problematic issues.

(a) The argmax inference sometimes selects spans that overlap with each other.

(b) The argmax inference cannot select multiple spans for one label.

In terms of (a), for example, when $\langle 1, 3, \texttt{A0} \rangle$ and $\langle 2, 4, \texttt{A1} \rangle$ are selected, a part of these two spans overlaps. In terms of (b), consider the following sentence.

He came to the U.S. yesterday at 5 p.m.
[A0] [A4] [TMP] [TMP]

In this example, the label TMP is assigned to the two spans ("yesterday" and "at 5 p.m."). Semantic role labels are mainly categorized into (i) *core labels* or (ii) *adjunct labels*. In the above example, the labels A0 and A4 are regarded as core labels, which indicate obligatory arguments for the predicate. In contrast, the labels like TMP are regarded as adjunct labels, which indicate optional arguments for the predicate. As the example shows, adjunct labels can be assigned to multiple spans.

To deal with these issues, we use a greedy search that keeps the consistency among spans and can return multiple spans for adjunct labels. Specifically, we greedily select higher scoring labeled spans subject to two constraints.

Overlap Constraint: Any spans that overlap with the selected spans cannot be selected.

Number Constraint: While multiple spans can be selected for each adjunct label, at most one span can be selected for each core label.

As a precise description of this algorithm, we describe the pseudo code and its explanation in Appendix A.

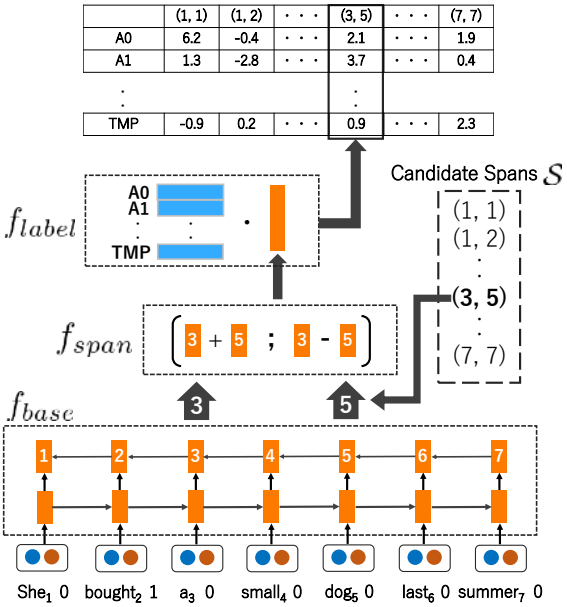

Figure 1: Overall architecture of our BiLSTM-span model.

3 Network Architecture

To compute the score for each span, we have introduced three functions ($f_{base}, f_{span}, f_{label}$) in Section 2.3. As an instantiation of each function, we use neural networks. This section describes our neural networks for each function and the overall network architecture.

3.1 BiLSTM-Span Model

Figure 1 illustrates the overall architecture of our model. The first component f_{base} uses bidirectional LSTMs (BiLSTMs) (Schuster and Paliwal, 1997; Graves et al., 2005, 2013) to calculate the base features. From the base features, the second component f_{span} extracts span features. Based on them, the final component f_{label} calculates the score for each labeled span. In the following, we describe these three components in detail.

Base Feature Function

As the base feature function f_{base}, we use BiLSTMs,

$$f_{base}(w_{1:T}, p) = \text{BiLSTM}(w_{1:T}, p) \ .$$

There are some variants of BiLSTMs. Following the deep SRL models proposed by Zhou and Xu (2015) and He et al. (2017), we stack BiLSTMs in an interleaving fashion. The stacked BiLSTMs process an input sequence in a left-to-right manner at odd-numbered layers and in a right-to-left manner at even-numbered layers.

The first layer of the stacked BiLSTMs receives word embeddings $\mathbf{x}^{word} \in \mathbb{R}^{d^{word}}$ and predicate mark embeddings $\mathbf{x}^{mark} \in \mathbb{R}^{d^{mark}}$. As the word embeddings, we can use existing word embeddings. The mark embeddings are created from the mark feature which has a binary value. The value is 1 if the word is the target predicate and 0 otherwise. For example, at the bottom part of Figure 1, the word "bought" is the target predicate and assigned 1 as its mark feature.

Receiving these inputs, the stacked BiLSTMs calculates the hidden states until the top layer. We use these hidden states as the input feature vectors $\mathbf{h}_{1:T}$ for the span feature function f_{span} (Eq. 5). Each vector $\mathbf{h}_t \in \mathbf{h}_{1:T}$ has d^{hidden} dimensions. We provide a detailed description of the stacked BiLSTMs in Appendix B.

Span Feature Function

From the base features induced by the BiLSTMs, we create the span feature representations,

$$f_{span}(\mathbf{h}_{1:T}, s) = [\mathbf{h}_i + \mathbf{h}_j; \mathbf{h}_i - \mathbf{h}_j] \ , \quad (7)$$

where the addition and subtraction features of the i-th and j-th hidden states are concatenated and used as the feature for a span $s = (i, j)$. The resulting vector $\mathbf{h}_s$ is a $2d^{hidden}$ dimensional vector.

The middle part of Figure 1 shows an example of this process. For the span $(3, 5)$, the span feature function f_{span} receives the 3rd and 5th features ($\mathbf{h}_3$ and $\mathbf{h}_5$). Then, these two vectors are added, and the 5th vector is subtracted from the 3rd vector. The resulting vectors are concatenated and given to the labeling function f_{label}.

Our design of the span features is inspired by the span (or segment) features used in syntactic parsing (Wang and Chang, 2016; Stern et al., 2017; Teranishi et al., 2017). While these neural span features cannot be used in BIO-based SRL models, they can easily be incorporated into span-based models.

Labeling Function

Taking a span representation $\mathbf{h}_s$ as input, the labeling function f_{label} returns the score for the span $s = (i, j)$ with a label r. Specifically, we use the following labeling function,

$$f_{label}(\mathbf{h}_s, r) = \mathbf{W}[r] \cdot \mathbf{h}_s \ , \quad (8)$$

where $\mathbf{W} \in \mathbb{R}^{|\mathcal{R}| \times 2d^{hidden}}$ has a row vector associated with each label r, and $\mathbf{W}[r]$ denotes the r-th row vector. As the result of the inner product of $\mathbf{W}[r]$ and $\mathbf{h}_s$, we obtain the score for a span (i, j) with a label r.

The upper part of Figure 1 shows an example of this process. The span representation $\mathbf{h}_s$ for the span $s = (3, 5)$ is created from addition and subtraction of $\mathbf{h}_3$ and $\mathbf{h}_5$. Then, we calculate the inner product of $\mathbf{h}_s$ and $\mathbf{W}[r]$. The score for the label `A0` is 2.1, and the score for the label `A1` is 3.7. In the same manner, by calculating the scores for all the spans $\mathcal{S}$ and labels $\mathcal{R}$, we can obtain the score matrix (at the top part of Figure 1).

3.2 Ensembling

We propose an ensemble model that uses span representations from multiple models. Each base model trained with different random initializations has variance in span representations. To take advantage of it, we introduce a variant of a mixture of experts (MoE) (Shazeer et al., 2017), [3]

$$\mathbf{h}_s^{\text{moe}} = \mathbf{W}_s^{\text{moe}} \cdot \sum_{m=1}^{M} \alpha_m \, \mathbf{h}_s^{(m)} \ , \quad (9)$$

$$f_{label}^{\text{moe}}(\mathbf{h}_s^{\text{moe}}, r) = \mathbf{W}^{\text{moe}}[r] \cdot \mathbf{h}_s^{\text{moe}} \ . \quad (10)$$

Firstly, we combine span representations $\mathbf{h}_s^{(m)}$ from each model $m \in \{1, \cdots, M\}$. $\mathbf{W}_s^{\text{moe}}$ is a parameter matrix and $\{\alpha_m\}_{m=1}^{M}$ are trainable, softmax-normalized parameters. Then, using the combined span representation $\mathbf{h}_s^{\text{moe}}$, we calculate the score in the same way as Eq. 8. We use the same greedy search algorithm used for our base model (Section 2.4).

During training, we update only the parameters of the ensemble model, i.e., $\{\mathbf{W}_s^{\text{moe}}, \mathbf{W}^{\text{moe}}, \{\alpha_m\}_{m=1}^{M}\}$. That is, we fix the parameters of each trained model m. As the loss function, we use the cross-entropy (Eq. 3).

4 Experiments

4.1 Datasets

We use the CoNLL-2005 and 2012 datasets[4]. We follow the standard train-development-test split and use the official evaluation script[5] from the CoNLL-2005 shared task on both datasets.

[3] One popular ensemble model for SRL is the product of experts (PoE) model (FitzGerald et al., 2015; He et al., 2017; Tan et al., 2018). In our preliminary experiments, we tried the PoE model but it did not improve the performance.

[4] We use the version of OntoNotes downloaded at: http://cemantix.org/data/ontonotes.html.

[5] The script can be downloaded at: http://www.lsi.upc.edu/ srlconll/soft.html

Emb	Model	Development			Test WSJ			Test Brown			Test ALL		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1
Senna	CRF	81.7	81.3	81.5	83.3	82.5	82.9	72.6	70.0	71.3	81.9	80.8	81.4
	Span	83.6	81.4	82.5	84.7	82.3	83.5	76.0	70.4	73.1	83.6	80.7	82.1
	Span (Ensemble)	85.6	82.6	84.1	86.6	83.6	85.1	78.2	71.8	74.8	85.5	82.0	83.7
ELMo	CRF	86.6	86.8	86.7	87.4	87.3	87.3	78.5	78.3	78.4	86.2	86.1	86.1
	Span	87.4	86.3	86.9	88.2	87.0	87.6	79.9	77.5	78.7	87.1	85.7	86.4
	Span (Ensemble)	**88.0**	**86.9**	**87.4**	**89.2**	**87.9**	**88.5**	**81.0**	**78.4**	**79.6**	**88.1**	**86.6**	**87.4**

Table 1: Experimental results on the CoNLL-2005 dataset, in terms of precision (P), recall (R) and F1. The bold numbers denote the highest precision, recall and F1 scores among all the models.

Emb	Model	Development			Test		
		P	R	F1	P	R	F1
Senna	CRF	82.8	81.9	82.4	82.9	81.9	82.4
	Span	84.3	81.5	82.9	84.4	81.7	83.0
	Span (Ensemble)	86.0	83.0	84.5	86.1	83.3	84.7
ELMo	CRF	86.1	**85.8**	85.9	86.0	**85.7**	85.9
	Span	87.2	85.5	86.3	87.1	85.3	86.2
	Span (Ensemble)	**88.6**	85.7	**87.1**	**88.5**	85.5	**87.0**

Table 2: Experimental results on the CoNLL-2012 dataset.

	CoNLL-05			CoNLL12
	WSJ	Brown	ALL	
Single Model				
ELMo-Span	**87.6**	78.7	**86.4**	**86.2**
He+ 18	87.4	**80.4**	-	85.5
Peters+ 18	-	-	-	84.6
Strubell+ 18	83.9	72.6	-	-
Tan+ 18	84.8	74.1	83.4	82.7
He+ 17	83.1	72.1	81.6	81.7
Zhou+ 15	82.8	69.4	81.1	81.3
FitzGerald+ 15	79.4	71.2	-	79.6
Täckström+ 15	79.9	71.3	-	79.4
Toutanova+ 08	79.7	67.8	-	-
Punyakanok+ 08	79.4	67.8	77.9	-
Ensemble Model				
ELMo-Span	**88.5**	**79.6**	**87.4**	**87.0**
Tan+ 18	86.1	74.8	84.6	83.9
He+ 17	84.6	73.6	83.2	83.4
FitzGerald+ 15	80.3	72.2	-	80.1
Toutanova+ 08	80.3	68.8	-	-
Punyakanok+ 08	79.4	67.8	77.9	-

Table 3: Comparison with existing models. The numbers denote F1 scores on each test set.

4.2 Baseline Model

For comparison, as a model based on BIO tagging approaches, we use the BiLSTM-CRF model proposed by Zhou and Xu (2015). The BiLSTMs for the base feature function f_{base} are the same as those used in our BiLSTM-span model.

4.3 Model Setup

As the base function f_{base}, we use 4 BiLSTM layers with 300 dimensional hidden units. To optimize the model parameters, we use Adam (Kingma and Ba, 2014). Other hyperparameters are described in Appendix C in detail.

Word Embeddings

Word embeddings have a great influence on SRL models. To validate the model performance, we use two types of word embeddings.

- Typical word embeddings, SENNA[6] (Collobert et al., 2011)
- Contextualized word embeddings, ELMo[7] (Peters et al., 2018)

SENNA and ELMo can be regarded as different types of embeddings in terms of the context sensitivity. SENNA and other typical word embeddings always assign an identical vector to each word regardless of the input context. In contrast, ELMo assigns different vectors to each word depending on the input context. In this work, we use these word embeddings that have different properties.[8] These embeddings are fixed during training.

Training

As the objective function, we use the cross-entropy $\mathcal{L}_\theta$ in Eq. 3 with L2 weight decay,

$$\mathcal{L}_\theta = \sum_{(X,Y)\in\mathcal{D}} \ell_\theta(X,Y) + \frac{\lambda}{2}||\theta||^2 \ , \qquad (11)$$

where the hyperparameter λ is the coefficient governing the L2 weight decay.

[6] http://ronan.collobert.com/senna/

[7] http://allennlp.org/elmo

[8] In our preliminary experiments, we also used the GloVe embeddings (Pennington et al., 2014), but the performance was worse than SENNA.

4.4 Results

We report averaged scores across five different runs of the model training.

Tables 1 and 2 show the experimental results on the CoNLL-2005 and 2012 datasets. Overall, our span-based ensemble model using ELMo achieved the best F1 scores, 87.4 F1 and 87.0 F1 on the CoNLL-2005 and CoNLL-2012 datasets, respectively. In comparison with the CRF-based single model, our span-based single model consistently yielded better F1 scores regardless of the word embeddings, SENNA and ELMO. Although the performance difference was small between these models using ELMO, it seems natural because both models got much better results and approached to the performance upper bound.

Table 3 shows the comparison with existing models in F1 scores. Our single and ensemble models using ELMO achieved the best F1 scores on all the test sets except the Brown test set.

5 Analysis

To better understand our span-based model, we addressed the following questions and obtained the following findings.

Questions

(a) What are strengths and weaknesses of our span-based model compared with the CRF-based model?

(b) What aspect of SRL does ELMo improve?

Findings

(a) While the CRF-based model is better at span boundary identification (Section 5.1), the span-based model is better at label prediction, especially for A2 (Section 5.2).

(b) ELMo improves the model performance for span boundary identification (Section 5.1).

In addition, we have conducted qualitative analysis on span and label representations learned in the span-based model (Section 5.3).

5.1 Performance for Span Boundary Identification

We analyze the results predicted by the single models. We evaluate F1 scores only for the span boundary match, shown by Table 4. We regard a predicted boundary $\langle i, j, * \rangle$ as correct if it matches the gold annotation regardless of its label.

		CoNLL-05		CoNLL-12	
EMB	MODEL	F1	diff	F1	diff
SENNA	SPAN	86.6	-0.4	87.3	-0.6
	CRF	87.0		87.9	
ELMO	SPAN	90.5	-0.7	90.3	-0.6
	CRF	91.2		90.9	

Table 4: F1 scores only for span boundary match.

		CoNLL-05		CoNLL-12	
EMB	MODEL	Acc.	diff	Acc.	diff
SENNA	SPAN	95.3	+1.5	95.1	+1.5
	CRF	93.8		93.6	
ELMO	SPAN	96.1	+0.9	95.7	+1.3
	CRF	95.2		94.4	

Table 5: Accuracies only for semantic role labels.

On both datasets, the CRF-based models achieved better F1 than that of the span-based models. Also, compared with SENNA, ELMO yielded much better F1 by over 3.0. This suggests that a factor of the overall SRL performance gain by ELMO is the improvement of the model ability to identify span boundaries.

5.2 Performance for Label Prediction

We analyze labels of the predicted results. For labeled spans whose boundaries match the gold annotation, we evaluate the label accuracies. As Table 5 shows, the span-based models outperformed the CRF-based models. Also, interestingly, the performance gap between SENNA and ELMO was not so big as that for span boundary identification.

Label-wise Performance

Table 6 shows F1 scores for frequent labels on the CoNLL-2005 and 2012 datasets. For A0 and A1, the performances of the CRF-based and span-based models were almost the same. For A2, the span-based models outperformed the CRF-based model by about 1.0 F1 on the both datasets. [9]

Label Confusion Matrix

Figure 2 shows a confusion matrix for labeling errors of the span-based model using ELMo.[10] Following He et al. (2017), we only count predicted arguments that match the gold span boundaries.

[9] The PNC label got low scores on the CoNLL-2012 dataset in Table 6. Almost all the gold PNC (purpose) labels are assigned to only the news article domain texts of the CoNLL-2012 dataset. The other 6 domain texts have no or very few PNC labels. This can lead to the low performance.

[10] We have observed the same tendency of labeling confusions between the models using ELMo and SENNA.

	CoNLL-2005				CoNLL-2012			
	SENNA		ELMO		SENNA		ELMO	
Label	CRF	SPAN	CRF	SPAN	CRF	SPAN	CRF	SPAN
A0	89.9	90.2	93.0	93.2	89.9	90.0	92.5	92.5
A1	83.2	83.8	89.1	89.2	84.7	85.1	88.7	89.0
A2	70.9	73.1	80.0	81.2	78.6	79.4	83.2	84.2
A3	64.4	71.2	78.8	78.5	61.9	62.9	69.0	70.7
ADV	59.3	61.9	68.1	67.0	63.2	63.7	67.5	67.0
DIR	43.2	47.3	56.6	54.5	54.1	52.0	61.1	59.7
LOC	58.2	60.5	68.1	68.3	65.8	65.0	72.0	72.0
MNR	61.4	61.3	66.5	67.7	64.4	65.7	70.5	71.1
PNC	57.3	60.2	68.8	67.7	18.5	13.7	20.2	16.1
TMP	81.8	82.7	86.1	86.0	82.2	82.3	86.1	86.2
Overall	81.5	82.5	86.7	86.9	82.4	82.9	85.9	86.3

Table 6: F1 Scores for frequent labels on the development set of the CoNLL-2005 and 2012 datasets.

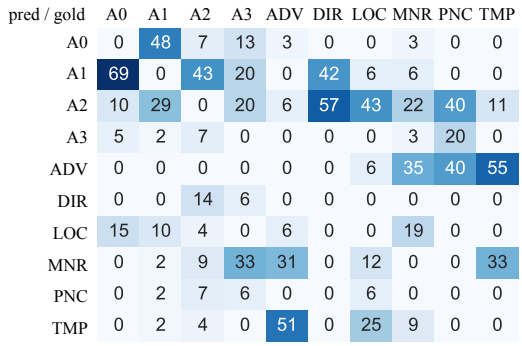

Figure 2: Confusion matrix for labeling errors of our span-based model using ELMo. Each cell shows the percentage of predicted labels for each gold label.

The span-based model confused A0 and A1 arguments the most. In particular, the model confused them for ergative verbs. Consider the following two sentences:

People start their own business ...
[A0]

.. Congress has started to jump on ...
[A1]

where the constituents located at the syntactic subjective position fulfill a different role A0 or A1 according to their semantic properties, such as animacy. Such arguments are difficult for SRL models to correctly identify.

Another point is the confusions of A2 with DIR and LOC. As He et al. (2017) pointed out, A2 in a lot of verb frames represents semantic relations such as direction or location, which can cause the confusions of A2 with such location-related adjuncts. To remedy these two problematic issues, it can be a promising approach to incorporate frame knowledge into SRL models by using verb frame dictionaries.

"... toy makers to move [across the border] ." GOLD:A2 PRED:DIR		
Nearest neighbors of "across the border"		
1	DIR	across the Hudson
2	DIR	outside their traditional tony circle
3	DIR	across the floor
4	DIR	through this congress
5	A2	off their foundations
6	DIR	off its foundation
7	DIR	off the center field wall
8	A3	out of bed
9	A2	through cottage rooftops
10	DIR	through San Francisco

Table 7: Example of the CoNLL-2005 development set, in which our model misclassified the label for the span "across the border". We collect 10 nearest neighbors of this span from the training set.

5.3 Qualitative Analysis on Our Model

On Span Representations

Our span-based model computes and uses span representations (Eq. 7) for label prediction. To investigate a relation between the span representations and predicted labels, we qualitatively analyze nearest neighbors of each span representation with its predicted label. Specifically, for each predicted span in the development set, we collect 10 nearest neighbor spans with their gold labels from the training set.

Table 7 shows 10 nearest neighbors of a span "across the border" for the predicate "move". The label of this span was misclassified, i.e., the predicted label is DIR but the gold is A2. Looking at its nearest neighbor spans, they have different gold labels, such as DIR, A2 and A3. Like this case, we have observed that spans with a misclassified label often have their nearest neighbors with inconsistent labels.

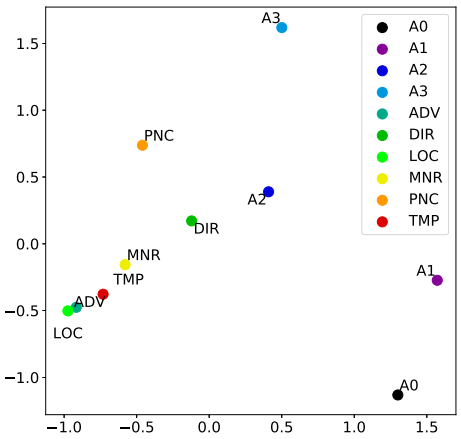

Figure 3: Label embedding distribution of our span-based model.

On Label Embeddings

We analyze the label embeddings in the labeling function (Eq. 8). Figure 3 shows the distribution of the learned label embeddings. The adjunct labels are close to each other, which are likely to be less discriminative. Also, the core label A2 is close to the adjunct label DIR, which are often confused by the model. To enhance the discriminative power, it is promising to apply techniques that keep label representations far away from each other (Wen et al., 2016; Luo et al., 2017).

6 Related Work

6.1 Semantic Role Labeling Tasks

Automatic SRL has been widely studied (Gildea and Jurafsky, 2002). There have been two main styles of SRL.

- FrameNet-style SRL (Baker et al., 1998)
- PropBank-style SRL (Palmer et al., 2005)

In this paper, we have tackled PropBank-style SRL.[11]

In PropBank-style SRL, there have been two main task settings.

- Span-based SRL: CoNLL-2004 and 2005 shared tasks (Carreras and Marquez, 2004; Carreras and Màrquez, 2005)
- Dependency-based SRL: CoNLL-2008 and 2009 shared tasks (Surdeanu et al., 2008; Hajič et al., 2009)

[11] Detailed descriptions on FrameNet-style and PropBank-style SRL can be found in Baker et al. (1998); Das et al. (2014); Kingsbury and Palmer (2002); Palmer et al. (2005).

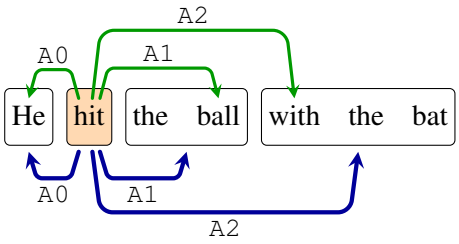

Figure 4: Example of dependency-based SRL (the upper part) and span-based SRL (the lower part).

Figure 4 illustrates an example of span-based and dependency-based SRL. In dependency-based SRL (at the upper part of Figure 4), the correct A2 argument for the predicate "hit" is the word "with". On one hand, in span-based SRL (at the lower part of Figure 4), the correct A2 argument is the span "with the bat".

For span-based SRL, the CoNLL-2004 and 2005 shared tasks (Carreras and Marquez, 2004; Carreras and Màrquez, 2005) provided the task settings and datasets. In the task settings, various SRL models, from traditional pipeline models to recent neural ones, have been proposed and competed with each other (Pradhan et al., 2005; He et al., 2017; Tan et al., 2018). For dependency-based SRL, the CoNLL-2008 and 2009 shared tasks (Surdeanu et al., 2008; Hajič et al., 2009) provided the task settings and datasets. As in span-based SRL, recent neural models achieved high-performance in dependency-based SRL (Marcheggiani et al., 2017; Marcheggiani and Titov, 2017; He et al., 2018b; Cai et al., 2018). This paper focuses on span-based SRL.

6.2 BIO-based SRL Models

Span-based SRL can be solved as BIO sequential tagging (Hacioglu et al., 2004; Pradhan et al., 2005; Màrquez et al., 2005).

Neural models State-of-the-art SRL models use neural networks based on the BIO tagging approach. The pioneering neural SRL model was proposed by Collobert et al. (2011). They use convolutional neural networks (CNNs) and CRFs. Instead of CNNs, Zhou and Xu (2015) and He et al. (2017) used stacked BiLSTMs and achieved strong performance without syntactic inputs. Tan et al. (2018) replaced stacked BiLSTMs with self-attention architectures. Strubell et al. (2018) improved the self-attention SRL model by incorporating syntactic information.

Word representations Typical word representations, such as SENNA (Collobert et al., 2011) and GloVe (Pennington et al., 2014), have been used and contributed to the performance improvement (Collobert et al., 2011; Zhou and Xu, 2015; He et al., 2017). Recently, Peters et al. (2018) integrated contextualized word representation, ELMo, into the model of He et al. (2017) and improved the performance by 3.2 F1 score. Strubell and McCallum (2018) also integrated ELMo into the model of Strubell et al. (2018) and reported the performance improvement.

6.3 Span-based SRL Models

Another line of approaches to SRL is labeled span modeling (Xue and Palmer, 2004; Koomen et al., 2005; Toutanova et al., 2005).

Typical models Typically, in this approach, models firstly identify candidate argument spans (argument identification) and then classify each span into one of the semantic role labels (argument classification). For inference, several effective methods have been proposed, such as structural constraint inference by using integer linear programming (Punyakanok et al., 2008) or dynamic programming (Täckström et al., 2015; FitzGerald et al., 2015).

Recent span-based model A very recent work, He et al. (2018a), proposed a span-based SRL model similar to our model. They also used BiLSTMs to induce span representations in an end-to-end fashion. A main difference is that while they model $\mathrm{P}(r|i,j)$, we model $\mathrm{P}(i,j|r)$. In other words, while their model seeks to select an appropriate label for each span (*label selection*), our model seeks to select appropriate spans for each label (*span selection*). This point distinguishes between their model and ours.

FrameNet span-based model For FrameNet-style SRL, Swayamdipta et al. (2017) used a segmental RNN (Kong et al., 2016), combining bidirectional RNNs with semi-Markov CRFs (Sarawagi and Cohen, 2004). Their model computes span representations using BiLSTMs and learns a conditional distribution over all possible labeled spans of an input sequence. Although we cannot compare our results with theirs, we can regard that our model is simpler and effective for PropBank-style SRL.

6.4 Span-based Models in Other NLP Tasks

In syntactic parsing, Wang and Chang (2016) proposed an LSTM-based sentence segment embedding method named LSTM-Minus. Stern et al. (2017); Kitaev and Klein (2018) incorporated the LSTM Minus into their parsing model and achieved the best results in constituency parsing. In coreference resolution, Lee et al. (2017, 2018) presented an end-to-end coreference resolution model, which considers all spans in a document as potential mentions and learn distributions over possible antecedents for each. Our model can be regarded as an extension of their model.

7 Conclusion and Future Work

We have presented a simple and accurate span-based model. We treat SRL as *span selection* and our model seeks to select appropriate spans for each label. Experimental results have demonstrated that despite the simplicity, the model outperforms a strong BiLSTM-CRF model. Also, our span-based ensemble model using ELMo achieves the state-of-the-art results on the CoNLL-2005 and 2012 datasets. Through empirical analysis, we have obtained some interesting findings. One of them is that the span-based model is better at label prediction compared with the CRF-based model. Another one is that ELMo improves the model performance for span boundary identification.

An interesting direction for future work concerns evaluating span representations from our span-based model. Since the investigation on the characteristics of the representations can lead to interesting findings, it is worthwhile evaluating them intrinsically and extrinsically. Another promising direction is to explore methods of incorporating frame knowledge into SRL models. We have observed that a lot of label confusions arise due to the lack of such knowledge. The use of frame knowledge to reduce these confusions is a straightforward approach.

Acknowledgments

This work was partially supported by JST CREST Grant Number JPMJCR1513 and JSPS KAKENHI Grant Number 18K18109. We are grateful to the members of the NAIST Computational Linguistics Laboratory, the members of Tohoku University Inui-Suzuki Laboratory, Kentaro Inui, Jun Suzuki, Yuichiro Matsubayashi, and the anonymous reviewers for their insightful comments.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of ACL-COLING*, pages 86–90.

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

Xavier Carreras and Lluís Marquez. 2004. Introduction to the conll-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL*, pages 89–97.

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

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. In *Journal of Machine Learning Research*.

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. 2014. Frame-semantic parsing. *Computational linguistics*, 40(1):9–56.

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

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AISTATS*, pages 249–256.

Alan Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *Proceedings of Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop*.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional LSTM networks for improved phoneme classification and recognition. In *Proceedings of International Conference on Artificial Neural Networks*, pages 799–804.

Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James H Martin, and Daniel Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *Proceedings of CoNLL*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL*, pages 1–18.

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

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

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

D.P. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv: 1412.6980*.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *Proceedings of LREC*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of ACL*, pages 2676–2686.

Lingpeng Kong, Chris Dyer, and Noah A Smith. 2016. Segmental recurrent neural networks. In *Proceedings of ICLR*.

Peter Koomen, Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *Proceedings of CoNLL*, pages 181–184.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of EMNLP*, pages 188–197.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of NAACL*, pages 687–692.

Lingkun Luo, Xiaofang Wang, Shiqiang Hu, Chao Wang, Yuxing Tang, and Liming Chen. 2017. Close yet distinctive domain adaptation. *arXiv preprint arXiv:1704.04235*.

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

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

Lluís Màrquez, Pere Comas, Jesús Giménez, and Neus Catala. 2005. Semantic role labeling as sequential tagging. In *Proceedings of CoNLL*, pages 193–196.

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

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

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

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H Martin, and Daniel Jurafsky. 2005. Semantic role chunking combining complementary syntactic views. In *Proceedings of CoNLL*, pages 217–220.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Proceedings of EMNLP-CoNLL*, pages 1–40.

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

Sunita Sarawagi and William W Cohen. 2004. Semi-markov conditional random fields for information extraction. In *Proceedings of NIPS*, pages 1185–1192.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, pages 2673–2681.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of ACL*, pages 818–827.

Emma Strubell and Andrew McCallum. 2018. Syntax helps elmo understand semantics: Is syntax still relevant in a deep neural architecture for srl? In *Proceedings of the Workshop on the Relevance of Linguistic Structure in Neural Architectures for NLP*, pages 19–27.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. *arXiv preprint arXiv:1804.08199*.

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

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A Smith. 2017. Frame-semantic parsing with softmax-margin segmental rnns and a syntactic scaffold. *arXiv preprint arXiv:1706.09528*.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of ACL*, 3:29–41.

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

Hiroki Teranishi, Hiroyuki Shindo, and Yuji Matsumoto. 2017. Coordination boundary identification with similarity and replaceability. In *Proceedings of IJCNLP*, pages 264–272.

Kristina Toutanova, Aria Haghighi, and Christopher D Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of ACL*, pages 589–596.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of ACL*, pages 2306–2315.

Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. 2016. A discriminative feature learning approach for deep face recognition. In *Proceedings of ECCV*, pages 499–515.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP*.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of ACL-IJCNLP*, pages 1127–1137.

A Span-Consistent Greedy Search

Algorithm 1 Span-Consistent Greedy Search

1: **Input:** Score Matrix $\mathbf{M} \in \mathbb{R}^{|\mathcal{R}|\times|\mathcal{S}|}$,
2: Predicate Position Index p
3: Core Label Set $\mathcal{R}^{(\text{core})}$
4: spans $\leftarrow \phi$
5: used_cores $\leftarrow \phi$
6: $\mathcal{U} \leftarrow \{(i, j, r, \text{score}) \in flatten(\mathbf{M})\}$
7: $\mathcal{U} \leftarrow filter(\mathcal{U}, p)$
8: **for** $(i, j, r, \text{score}) \in sort(\mathcal{U})$ **do**
9: **if** $r \notin$ used_cores **and**
10: is_overlap$((i, j), \text{spans})$ **is False then**
11: spans $\leftarrow$ spans $\cup \{\langle i, j, r\rangle\}$
12: **if** $r \in \mathcal{R}^{(\text{core})}$ **then**
13: used_cores $\leftarrow$ used_cores $\cup \{r\}$
14: **return** spans

Algorithm 1 describes the pseudo code of the greedy search algorithm introduced in Section 2.4. This algorithm receives the three inputs (line 1-3). $\mathbf{M}$ is the score matrix illustrated at the top part of Figure 1 in Section 3. Each cell of the matrix represents the score of each span. p is a target predicate position index. $\mathcal{R}^{(\text{core})}$ is the set of core labels. At line 4, the variable "spans" is initialized. This variable stores the selected spans to be returned as the output. At line 5, the variable "used_cores" is initialized. This variable keeps track of the already selected core labels.

At line 6, the score matrix $\mathbf{M}$ is converted to tuples, (i, j, r, score), by the function $flatten(\cdot)$. These tuples are stored in the variable $\mathcal{U}$. At line 7, from $\mathcal{U}$, we remove the tuples that fall into any one of the followings, (i) the tuples whose boundary (i, j) overlaps with the predicate position p or (ii) the tuples whose score is lower than that of the predicate span tuples. In terms of (i), since spans whose boundary (i, j) overlaps with the predicate position, $i \leq p \leq j$, can never be a correct argument, we remove such tuples. In terms of (ii), we remove the tuples $(*, *, r, \text{score})$ whose score is lower than that of the predicate span tuple (p, p, r, score). In Section 2, we define the predicate span (p, p) as the NULL span, implying that we can regard the spans whose score is lower than that of the NULL span as an inappropriate argument. Thus, we remove such tuples from the set of the candidates $\mathcal{U}$.

The main processing starts from line 8. Based on the scores, the function $sort(\cdot)$ sorts the tuples (i, j, r, score) in a descending order. At line 9-10, there are constraints for output spans. At line 9, "$r \notin$ used_cores" represents the constraint that at most one span can be selected for each core label. At line 10, the function is_overlap$(\cdot)$ takes as input a span (i, j) and the set of the selected spans, and returns the boolean value ("True" or "False") that represents whether the span overlaps with any one of the selected spans or not.

At line 11, the span is added to the set of the selected spans. At line 12-13, if the label r is included in the core labels $\mathcal{R}^{(\text{core})}$, the label is added to "used_cores". At line 14, as the final output, the set of the selected spans "spans" is returned.

B BiLSTMs

As the base feature function f_{base} (Eq. 4 in Section 2.3), we use BiLSTMs,

$$f_{base}(w_{1:T}, p) = \text{BiLSTM}(w_{1:T}, p) \ .$$

In particular, we use the stacked BiLSTMs in an interleaving fashion (Zhou and Xu, 2015; He et al., 2017). The stacked BiLSTMs process an input sequence in a left-to-right manner for odd-numbered layers and in a right-to-left manner for even-numbered layers.

The stacked BiLSTMs consist of L layers. The hidden state in each layer $\ell \in \{1, \cdots, L\}$ is calculated as follows,

$$\mathbf{h}_t^{(\ell)} = \begin{cases} \text{LSTM}^{(\ell)}(\mathbf{x}_t^{(\ell)}, \mathbf{h}_{t-1}^{(\ell)}) & (\ell = \text{odd}) \\ \text{LSTM}^{(\ell)}(\mathbf{x}_t^{(\ell)}, \mathbf{h}_{t+1}^{(\ell)}) & (\ell = \text{even}) \end{cases} .$$

Both of the odd- and even-numbered layers receive $\mathbf{x}_t^{(\ell)}$ as the first input of the LSTM. For the second input, odd-numbered layers receive $\mathbf{h}_{t-1}^{(\ell)}$, whereas even-numbered layers receive $\mathbf{h}_{t+1}^{(\ell)}$.

Between the LSTM layers, we use the following connection (Zhou and Xu, 2015),

$$\mathbf{x}_t^{(\ell+1)} = \text{ReLU}(\mathbf{W}^{(\ell)} \cdot [\mathbf{x}_t^{(\ell)}; \mathbf{h}_t^{(\ell)}]) \ .$$

Here, we firstly concatenate $\mathbf{x}_t^{(\ell)}$ and $\mathbf{h}_t^{(\ell)}$, and then calculate the inner product of the concatenated vector and the parameter matrix $\mathbf{W}^{(\ell)}$ with the rectified linear units (ReLU). As a result, we obtain the input representation $\mathbf{x}_t^{(\ell+1)}$ for the next $(\ell + 1\text{-th})$ LSTM layer.

In the first layer, LSTM$^{(1)}$ receives an input feature vector $\mathbf{x}_t^{(1)}$. Following He et al. (2017), we

create this vector by concatenating a word embedding and predicate mark embedding,

$$\mathbf{x}_t^{(1)} = [\mathbf{x}_t^{word}; \mathbf{x}_t^{mark}] \ ,$$

where $\mathbf{x}^{word} \in \mathbb{R}^{d^{word}}$ and $\mathbf{x}^{mark} \in \mathbb{R}^{d^{mark}}$. The mark embedding is created from the binary mark feature. The value is 1 if the word is the target predicate and 0 otherwise.

After the L-th LSTM layer runs, we obtain $\mathbf{x}_{1:T}^{(L+1)} = \mathbf{x}_1^{(L+1)}, \cdots, \mathbf{x}_T^{(L+1)}$. We use them as the input of the span feature function f_{span} (Eq. 5 in Section 2.3), i.e., $\mathbf{h}_{1:T} = \mathbf{x}_{1:T}^{(L+1)}$. Each vector $\mathbf{h}_t \in \mathbf{h}_{1:T}$ has d^{hidden} dimensions.

C Hyperparameters

Name	Value
Word Embedding d^{word}	50-dimensional SENNA 1024-dimensional ELMo
Mark Embedding d^{mark}	50-dimensional vector
LSTM Layers L	4
LSTM Hidden Units d^{hidden}	300 dimensions
Mini-batch Size	32
Optimization	Adam
Learning Rate	0.001
L2 Regularization λ	0.0001
Dropout Ratio for BiLSTMs	0.1
Dropout Ratio for ELMo	0.5

Table 8: Hyperparameters for our span-based model.

C.1 Span-based Model

Table 8 lists the hyperparameters used for our span-based model.

Word representation setup As word embeddings $\mathbf{x}^{word}$, we use two types of embeddings, (i) SENNA (Collobert et al., 2011), 50-dimensional word vectors ($d^{word} = 50$), and (ii) ELMo (Peters et al., 2018), 1024-dimensional vectors ($d^{word} = 1024$). During training, we fix these word embeddings (not update them). As predicate mark embeddings $\mathbf{x}^{mark}$, we use randomly initialized 50-dimensional vectors ($d^{mark} = 50$). During training, we update them.

Network setup As the base feature function f_{base}, we use 4 stacked BiLSTMs (2 forward and 2 backward LSTMs) with 300-dimensional hidden units ($d^{hidden} = 300$). Following He et al. (2017), we initialize all the parameter matrices in BiLSTMs with random orthonormal matrices (Saxe et al., 2013). Other parameters are initialized following Glorot and Bengio (2010), and bias parameters are initialized with zero vectors.

Regularization We set the coefficient λ for the L2 weight decay (Eq. 11 in Section 4.3) to 0.0001. We apply dropout (Srivastava et al., 2014) to the input vectors of each LSTM with dropout ratio of 0.1 and the ELMo embeddings with dropout ratio of 0.5.

Training To optimize the parameters, we use Adam (Kingma and Ba, 2014) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate is initialized to 0.001. After training 50 epochs, we halve the learning rate every 25 epochs. Parameter updates are performed in mini-batches of 32. The number of training epochs is set to 100. We save the parameters that achieve the best F1 score on the development set and evaluate them on the test set. Training our model on the CoNLL-2005 training set takes about one day and on the CoNLL-2012 training set takes about two days on a single GPU, respectively.

C.2 Ensemble Model

Our ensemble model uses span representations $\mathbf{h}_s^{(m)}$ from base models $m \in \{1, \cdots, M\}$ (Section 3.2). We use 5 base models ($M = 5$) learned over different runs. Note that, during training, we fix the parameters of the five base models and update only the parameters of the ensemble model.

Network setup The parameter matrix $\mathbf{W}_s^{\text{moe}}$ (Eq. 9 in Section 3.2) is initialized with the identity matrix. The scalar parameters $\{\alpha_m\}_{m=1}^M$ (Eq. 9) are initialized with 0. Each row vector $\mathbf{W}^{\text{moe}}[r]$ of the parameter matrix $\mathbf{W}^{\text{moe}}$ (Eq. 10) is initialized with the averaged vector over the row vectors $\mathbf{W}^{(m)}[r]$ of each model m, i.e., $\frac{1}{M}\sum_{m=1}^M \mathbf{W}^{(m)}[r]$.

Training To optimize the parameters, we use Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate is set to 0.0001. Parameter updates are performed in mini-batches of 8. The number of training epochs is set to 20. We save the parameters that achieve the best F1 score on the development set and evaluate them on the test set. Training one ensemble model on the CoNLL-2005 and 2012 training sets takes about one day on a single GPU.