Deep Semantic Role Labeling with Self-Attention

Zhixing Tan[1], Mingxuan Wang[2], Jun Xie[2], Yidong Chen[1], Xiaodong Shi[1]*
[1]School of Information Science and Engineering, Xiamen University, Xiamen, China
[2]Mobile Internet Group, Tencent Technology Co., Ltd, Beijing, China
playinf@stu.xmu.edu.cn,
{xuanswang, stiffxie}@tencent.com,
{ydchen, mandel}@xmu.edu.cn

Abstract

Semantic Role Labeling (SRL) is believed to be a crucial step towards natural language understanding and has been widely studied. Recent years, end-to-end SRL with recurrent neural networks (RNN) has gained increasing attention. However, it remains a major challenge for RNNs to handle structural information and long range dependencies. In this paper, we present a simple and effective architecture for SRL which aims to address these problems. Our model is based on self-attention which can directly capture the relationships between two tokens regardless of their distance. Our single model achieves $F_1 = 83.4$ on the CoNLL-2005 shared task dataset and $F_1 = 82.7$ on the CoNLL-2012 shared task dataset, which outperforms the previous state-of-the-art results by 1.8 and 1.0 F_1 score respectively. Besides, our model is computationally efficient, and the parsing speed is 50K tokens per second on a single Titan X GPU.

Introduction

Semantic Role Labeling is a shallow semantic parsing task, whose goal is to determine essentially "who did what to whom", "when" and "where". Semantic roles indicate the basic event properties and relations among relevant entities in the sentence and provide an intermediate level of semantic representation thus benefiting many NLP applications, such as Information Extraction (Bastianelli et al. 2013), Question Answering (Surdeanu et al. 2003; Moschitti, Morarescu, and Harabagiu 2003; Dan and Lapata 2007), Machine Translation (Knight and Luk 1994; Ueffing, Haffari, and Sarkar 2007; Wu and Fung 2009) and Multi-document Abstractive Summarization (Genest and Lapalme 2011).

Semantic roles are closely related to syntax. Therefore, traditional SRL approaches rely heavily on the syntactic structure of a sentence, which brings intrinsic complexity and restrains these systems to be domain specific. Recently, end-to-end models for SRL without syntactic inputs achieved promising results on this task (Zhou and Xu 2015; Marcheggiani, Frolov, and Titov 2017; He et al. 2017). As the pioneering work, Zhou and Xu (2015) introduced a stacked long short-term memory network (LSTM) and achieved the state-of-the-art results. He et al., (2017) reported further improvements by using deep highway bidirectional LSTMs with constrained decoding. These successes involving end-to-end models reveal the potential ability of LSTMs for handling the underlying syntactic structure of the sentences.

Despite recent successes, these RNN-based models have limitations. RNNs treat each sentence as a sequence of words and recursively compose each word with its previous hidden state. The recurrent connections make RNNs applicable for sequential prediction tasks with arbitrary length, however, there still remain several challenges in practice. The first one is related to memory compression problem (Cheng, Dong, and Lapata 2016). As the entire history is encoded into a single fixed-size vector, the model requires larger memory capacity to store information for longer sentences. The unbalanced way of dealing with sequential information leads the network performing poorly on long sentences while wasting memory on shorter ones. The second one is concerned with the inherent structure of sentences. RNNs lack a way to tackle the tree-structure of the inputs. The sequential way to process the inputs remains the network depth-in-time, and the number of nonlinearities depends on the time-steps.

To address these problems above, we present a deep attentional neural network (DEEPATT) for the task of SRL[1]. Our models rely on the self-attention mechanism which directly draws the global dependencies of the inputs. In contrast to RNNs, a major advantage of self-attention is that it conducts direct connections between two arbitrary tokens in a sentence. Therefore, distant elements can interact with each other by shorter paths ($O(1)$ v.s. $O(n)$), which allows unimpeded information flow through the network. Self-attention also provides a more flexible way to select, represent and synthesize the information of the inputs and is complementary to RNN based models. Along with self-attention, DEEPATT comes with three variants which uses recurrent (RNN), convolutional (CNN) and feed-forward (FFN) neural network to further enhance the representations.

Although DEEPATT is fairly simple, it gives remarkable empirical results. Our single model outperforms the previ-

* Corresponding author.

[1]Our source code is available at https://github.com/XMUNLP/Tagger

ous state-of-the-art systems on the CoNLL-2005 shared task dataset and the CoNLL-2012 shared task dataset by 1.8 and 1.0 F_1 score respectively. It is also worth mentioning that on the out-of-domain dataset, we achieve an improvement upon the previous end-to-end approach (He et al. 2017) by 2.0 F_1 score. The feed-forward variant of DEEPATT allows significantly more parallelization, and the parsing speed is 50K tokens per second on a single Titan X GPU.

Semantic Role Labeling

Given a sentence, the goal of SRL is to identify and classify the arguments of each target verb into semantic roles. For example, for the sentence *"Marry borrowed a book from John last week."* and the target verb *borrowed*, SRL yields the following outputs:

[$_{\text{ARG0}}$ Marry] [$_{\text{V}}$ borrowed] [$_{\text{ARG1}}$ a book]
[$_{\text{ARG2}}$ from John] [$_{\text{AM-TMP}}$ last week].

Here ARG0 represents the *borrower*, ARG1 represents the *thing borrowed*, ARG2 represents the *entity borrowed from*, AM-TMP is an adjunct indicating the timing of the action and V represents the verb.

Generally, semantic role labeling consists of two steps: identifying and classifying arguments. The former step involves assigning either a semantic argument or non-argument for a given predicate, while the latter includes labeling a specific semantic role for the identified argument. It is also common to prune obvious non-candidates before the first step and to apply post-processing procedure to fix inconsistent predictions after the second step. Finally, a dynamic programming algorithm is often applied to find the global optimum solution for this typical sequence labeling problem at the inference stage.

In this paper, we treat SRL as a BIO tagging problem. Our approach is extremely simple. As illustrated in Figure 1, the original utterances and the corresponding predicate masks are first projected into real-value vectors, namely embeddings, which are fed to the next layer. After that, we design a deep attentional neural network which takes the embeddings as the inputs to capture the nested structures of the sentence and the latent dependency relationships among the labels. On the inference stage, only the topmost outputs of attention sub-layer are taken to a logistic regression layer to make the final decision [2].

Deep Attentional Neural Network for SRL

In this section, we will describe DEEPATT in detail. The main component of our deep network consists of N identical layers. Each layer contains a nonlinear sub-layer followed by an attentional sub-layer. The topmost layer is the softmax classification layer.

Self-Attention

Self-attention or intra-attention, is a special case of attention mechanism that only requires a single sequence to compute its representation. Self-attention has been successfully applied to many tasks, including reading comprehension, abstractive summarization, textual entailment, learning task-independent sentence representations, machine translation and language understanding (Cheng, Dong, and Lapata 2016; Parikh et al. 2016; Lin et al. 2017; Paulus, Xiong, and Socher 2017; Vaswani et al. 2017; Shen et al. 2017).

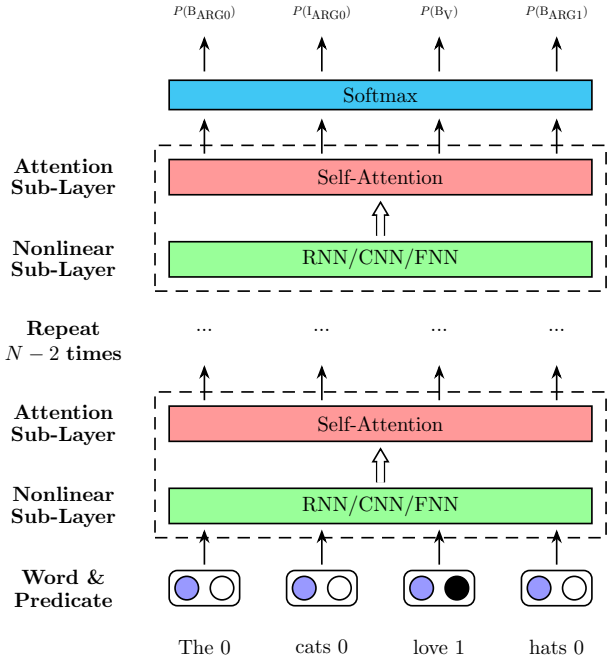

Figure 1: An illustration of our deep attentional neural network. Original utterances and corresponding predicate masks are taken as the only inputs for our deep model. For example, *love* is the predicate and marked as 1, while other words are marked as 0.

In this paper, we adopt the multi-head attention formulation by Vaswani et al. (2017). Figure 2 depicts the computation graph of multi-head attention mechanism. The center of the graph is the scaled dot-product attention, which is a variant of dot-product (multiplicative) attention (Luong, Pham, and Manning 2015). Compared with the standard additive attention mechanism (Bahdanau, Cho, and Bengio 2014) which is implemented using a one layer feed-forward neural network, the dot-product attention utilizes matrix production which allows faster computation. Given a matrix of n query vectors $\mathbf{Q} \in \mathbb{R}^{n \times d}$, keys $\mathbf{K} \in \mathbb{R}^{n \times d}$ and values $\mathbf{V} \in \mathbb{R}^{n \times d}$, the scaled dot-product attention computes the attention scores based on the following equation:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V} \qquad (1)$$

where d is the number of hidden units of our network.

The multi-head attention mechanism first maps the matrix of input vectors $\mathbf{X} \in \mathbb{R}^{t \times d}$ to queries, keys and values matrices by using different linear projections. Then h parallel heads are employed to focus on different part of channels

[2] In case of BIO violations, we simply treat the argument of the B tags as the argument of the whole span.

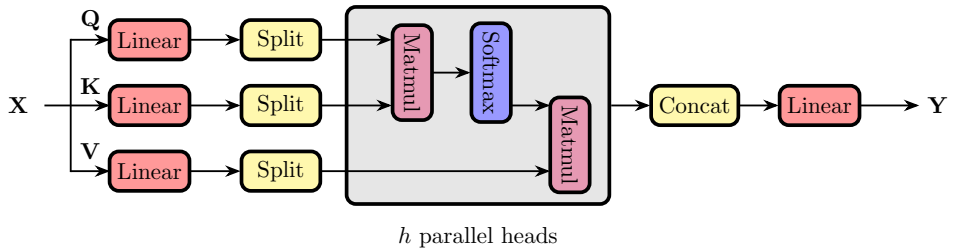

Figure 2: The computation graph of multi-head self-attention mechanism. All heads can be computed in parallel using highly optimized matrix multiplication codes.

of the value vectors. Formally, for the i-th head, we denote the learned linear maps by $\mathbf{W}_i^Q \in \mathbb{R}^{n\times d/h}$, $\mathbf{W}_i^K \in \mathbb{R}^{n\times d/h}$ and $\mathbf{W}_i^V \in \mathbb{R}^{n\times d/h}$, which correspond to queries, keys and values respectively. Then the scaled dot-product attention is used to compute the relevance between queries and keys, and to output mixed representations. The mathematical formulation is shown below:

$$\mathbf{M}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \tag{2}$$

Finally, all the vectors produced by parallel heads are concatenated together to form a single vector. Again, a linear map is used to mix different channels from different heads:

$$\mathbf{M} = \text{Concat}(\mathbf{M}_1, \ldots, \mathbf{M}_h) \tag{3}$$
$$\mathbf{Y} = \mathbf{M}\mathbf{W} \tag{4}$$

where $\mathbf{M} \in \mathbb{R}^{n\times d}$ and $\mathbf{W} \in \mathbb{R}^{d\times d}$.

The self-attention mechanism has many appealing aspects compared with RNNs or CNNs. Firstly, the distance between any input and output positions is 1, whereas in RNNs it can be n. Unlike CNNs, self-attention is not limited to fixed window sizes. Secondly, the attention mechanism uses weighted sum to produce output vectors. As a result, the gradient propagations are much easier than RNNs or CNNs. Finally, the dot-product attention is highly parallel. In contrast, RNNs are hard to parallelize owing to its recursive computation.

Nonlinear Sub-Layers

The successes of neural networks root in its highly flexible nonlinear transformations. Since attention mechanism uses weighted sum to generate output vectors, its representational power is limited. To further increase the expressive power of our attentional network, we employ a nonlinear sub-layer to transform the inputs from the bottom layers. In this paper, we explore three kinds of nonlinear sub-layers, namely recurrent, convolutional and feed-forward sub-layers.

Recurrent Sub-Layer We use bidirectional LSTMs to build our recurrent sub-layer. Given a sequence of input vectors $\{\mathbf{x}_t\}$, two LSTMs process the inputs in opposite directions. To maintain the same dimension between inputs and outputs, we use the sum operation to combine two representations:

$$\overrightarrow{\mathbf{h}}_t = \text{LSTM}(\mathbf{x}_t, \overrightarrow{\mathbf{h}}_{t-1}) \tag{5}$$
$$\overleftarrow{\mathbf{h}}_t = \text{LSTM}(\mathbf{x}_t, \overleftarrow{\mathbf{h}}_{t+1}) \tag{6}$$
$$\mathbf{y}_t = \overrightarrow{\mathbf{h}}_t + \overleftarrow{\mathbf{h}}_t \tag{7}$$

Convolutional Sub-Layer For convolutional sub-layer, we use the Gated Linear Unit (GLU) proposed by Dauphin et al. (2016). Compared with the standard convolutional neural network, GLU is much easier to learn and achieves impressive results on both language modeling and machine translation task (Dauphin et al. 2016; Gehring et al. 2017). Given two filters $\mathbf{W} \in \mathbb{R}^{k\times d\times d}$ and $\mathbf{V} \in \mathbb{R}^{k\times d\times d}$, the output activations of GLU are computed as follows:

$$\text{GLU}(\mathbf{X}) = (\mathbf{X} * \mathbf{W}) \odot \sigma(\mathbf{X} * \mathbf{V}) \tag{8}$$

The filter width k is set to 3 in all our experiments.

Feed-forward Sub-Layer The feed-forward sub-layer is quite simple. It consists of two linear layers with hidden ReLU nonlinearity (Nair and Hinton 2010) in the middle. Formally, we have the following equation:

$$\text{FFN}(\mathbf{X}) = \text{ReLU}(\mathbf{X}\mathbf{W}_1)\mathbf{W}_2 \tag{9}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d\times h_f}$ and $\mathbf{W}_2 \in \mathbb{R}^{h_f\times d}$ are trainable matrices. Unless otherwise noted, we set $h_f = 800$ in all our experiments.

Deep Topology

Previous works pointed out that deep topology is essential to achieve good performance (Zhou and Xu 2015; He et al. 2017). In this work, we use the residual connections proposed by He et al. (2016) to ease the training of our deep attentional neural network. Specifically, the output $\mathbf{Y}$ of each sub-layer is computed by the following equation:

$$\mathbf{Y} = \mathbf{X} + \text{Sub-Layer}(\mathbf{X}) \tag{10}$$

We then apply layer normalization (Ba, Kiros, and Hinton 2016) after the residual connection to stabilize the activations of deep neural network.

Position Encoding

The attention mechanism itself cannot distinguish between different positions. So it is crucial to encode positions of

each input words. There are various ways to encode positions, and the simplest one is to use an additional position embedding. In this work, we try the timing signal approach proposed by Vaswani et al. (2017), which is formulated as follows:

$$\text{timing}(t, 2i) = \sin(t/10000^{2i/d}) \quad (11)$$

$$\text{timing}(t, 2i+1) = \cos(t/10000^{2i/d}) \quad (12)$$

The timing signals are simply added to the input embeddings. Unlike the position embedding approach, this approach does not introduce additional parameters.

Pipeline

The first step of using neural networks to process symbolic data is to represent them by distributed vectors, also called embeddings (Bengio et al. 2003). We take the very original utterances and the corresponding predicate masks $\mathbf{m}$ as the input features. m_t is set to 1 if the corresponding word is a predicate, or 0 if not.

Formally, in SRL task, we have a word vocabulary $\mathcal{V}$ and mask vocabulary $\mathcal{C} = \{0, 1\}$. Given a word sequence $\{x_1, x_2, \ldots, x_T\}$ and a mask sequence $\{m_1, m_2, ..., m_T\}$, each word $x_t \in \mathcal{V}$ and its corresponding predicate mask $m_t \in \mathcal{C}$ are projected into real-valued vectors $\mathbf{e}(x_t)$ and $\mathbf{e}(m_t)$ through the corresponding lookup table layer, respectively. The two embeddings are then concatenated together as the output feature maps of the lookup table layers. Formally speaking, we have $\mathbf{x}_t = [\mathbf{e}(w_t), \mathbf{e}(m_t)]$.

We then build our deep attentional neural network to learn the sequential and structural information of a given sentence based on the feature maps from the lookup table layer. Finally, we take the outputs of the topmost attention sub-layer as inputs to make the final predictions.

Since there are dependencies between semantic labels, most previous neural network models introduced a transition model for measuring the probability of jumping between the labels. Different from these works, we perform SRL as a typical classification problem. Latent dependency information is embedded in the topmost attention sub-layer learned by our deep models. This approach is simpler and easier to implement compared to previous works.

Formally, given an input sequence $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$, the log-likelihood of the corresponding correct label sequence $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}$ is

$$\log p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{t=1}^{n} \log p(y_t|\boldsymbol{x}; \boldsymbol{\theta}). \quad (13)$$

Our model predict the corresponding label y_t based on the representation $\mathbf{h}_t$ produced by the topmost attention sub-layer of DEEPATT:

$$p(y_t|\boldsymbol{x}; \boldsymbol{\theta}) = p(y_t|\mathbf{h}_t; \boldsymbol{\theta}) \quad (14)$$

$$= \text{softmax}(\mathbf{W}_o \mathbf{h}_t)^T \delta_{y_t}, \quad (15)$$

Where W_o is the softmax matrix and δ_{y_t} is Kronecker delta with a dimension for each output symbol, so $\text{softmax}(\mathbf{W}_o \mathbf{h}_t)^T \delta_{y_t}$ is exactly the y_t'th element of the distribution defined by the softmax. Our training objective is to maximize the log probabilities of the correct output labels given the input sequence over the entire training set.

Experiments

We report our empirical studies of DEEPATT on the two commonly used datasets from the CoNLL-2005 shared task and the CoNLL-2012 shared task.

Datasets

The CoNLL-2005 dataset takes section 2-21 of the Wall Street Journal (WSJ) corpus as training set, and section 24 as development set. The test set consists of section 23 of the WSJ corpus as well as 3 sections from the Brown corpus (Carreras and Màrquez 2005). The CoNLL-2012 dataset is extracted from the OntoNotes v5.0 corpus. The description and separation of training, development and test set can be found in Pardhan et al. (2013).

Model Setup

Initialization We initialize the weights of all sub-layers as random orthogonal matrices. For other parameters, we initialize them by sampling each element from a Gaussian distribution with mean 0 and variance $\frac{1}{\sqrt{d}}$. The embedding layer can be initialized randomly or using pre-trained word embeddings. We will discuss the impact of pre-training in the analysis subsection.[3]

Settings and Regularization The settings of our models are described as follows. The dimension of word embeddings and predicate mask embeddings is set to 100 and the number of hidden layers is set to 10. We set the number of hidden units d to 200. The number of heads h is set to 8. We apply dropout (Srivastava et al. 2014) to prevent the networks from over-fitting. Dropout layers are added before residual connections with a keep probability of 0.8. Dropout is also applied before the attention softmax layer and the feed-froward ReLU hidden layer, and the keep probabilities are set to 0.9. We also employ label smoothing technique (Szegedy et al. 2016) with a smoothing value of 0.1 during training.

Learning Parameter optimization is performed using stochastic gradient descent. We adopt Adadelta (Zeiler 2012) ($\epsilon = 10^6$ and $\rho = 0.95$) as the optimizer. To avoid exploding gradients problem, we clip the norm of gradients with a predefined threshold 1.0 (Pascanu et al. 2013). Each SGD contains a mini-batch of approximately 4096 tokens for the CoNLL-2005 dataset and 8192 tokens for the CoNLL-2012 dataset. The learning rate is initialized to 1.0. After training 400k steps, we halve the learning rate every 100K steps. We train all models for 600K steps. For DEEPATT with FFN sub-layers, the whole training stage takes about two days to finish on a single Titan X GPU, which is 2.5 times faster than the previous approach (He et al. 2017).

Results

In Table 1 and 2, we give the comparisons of DEEPATT with previous approaches. On the CoNLL-2005 dataset, the sin-

[3]To be strictly comparable to previous work, we use the same vocabularies and pre-trained embeddings as He et al.(2017).

Model	Development				WSJ Test				Brown Test				Combined
	P	R	F1	Comp.	P	R	F1	Comp.	P	R	F1	Comp.	F1
He et al. (Ensemble) (2017)	83.1	82.4	82.7	64.1	85.0	84.3	84.6	66.5	**74.9**	72.4	73.6	46.5	83.2
He et al. (Single) (2017)	81.6	81.6	81.6	62.3	83.1	83.0	83.1	64.3	72.8	71.4	72.1	44.8	81.6
Zhou and Xu (2015)	79.7	79.4	79.6	-	82.9	82.8	82.8	-	70.7	68.2	69.4	-	81.1
FitzGerald et al. (Struct., Ensemble) (2015)	81.2	76.7	78.9	55.1	82.5	78.2	80.3	57.3	74.5	70.0	72.2	41.3	-
Täckström et al. (Struct.) (2015)	81.2	76.2	78.6	54.4	82.3	77.6	79.9	56.0	74.3	68.6	71.3	39.8	-
Toutanova et al. (Ensemble) (2008)	-	-	78.6	58.7	81.9	78.8	80.3	60.1	-	-	68.8	40.8	-
Punyakanok et al. (Ensemble) (2008)	80.1	74.8	77.4	50.7	82.3	76.8	79.4	53.8	73.4	62.9	67.8	32.3	77.9
DEEPATT (RNN)	81.2	82.3	81.8	62.4	83.5	84.0	83.7	65.2	72.5	73.4	72.9	44.7	82.3
DEEPATT (CNN)	82.1	82.8	82.4	63.6	83.6	83.9	83.8	65.4	72.8	72.7	72.7	45.9	82.3
DEEPATT (FFN)	82.6	83.6	83.1	65.2	84.5	85.2	84.8	66.4	73.5	74.6	74.1	48.4	83.4
DEEPATT (FFN, Ensemble)	**84.3**	**84.9**	**84.6**	**67.3**	**85.9**	**86.3**	**86.1**	**69.0**	74.6	**75.0**	**74.8**	**48.6**	**84.6**

Table 1: Comparison with previous methods on the CoNLL-2005 dataset. We report the results in terms of precision (P), recall (R), F_1 and percentage of completely correct predicates (Comp.). Our single and ensemble model lead to substantial improvements over the previous state-of-the-art results.

Model	Development				Test			
	P	R	F1	Comp.	P	R	F1	Comp.
He et al. (Ensemble) (2017)	83.5	83.2	83.4	67.5	**83.5**	83.3	83.4	68.5
He et al. (Single) (2017)	81.7	81.4	81.5	64.6	81.8	81.6	81.7	66.0
Zhou and Xu (2015)	-	-	81.1	-	-	-	81.3	-
FitzGerald et al. (Struct., Ensemble) (2015)	81.0	78.5	79.7	60.9	81.2	79.0	80.1	62.6
Täckström et al. (Struct., Ensemble) (2015)	80.5	77.8	79.1	60.1	80.6	78.2	79.4	61.8
Pradhan et al.(Revised) (2013)	-	-	-	-	78.5	76.6	77.5	55.8
DEEPATT (RNN)	81.0	82.3	81.6	64.6	80.9	82.2	81.5	65.7
DEEPATT (CNN)	80.1	82.5	81.3	65.0	79.8	82.6	81.2	66.1
DEEPATT (FFN)	82.2	83.6	82.9	66.7	81.9	83.6	82.7	67.5
DEEPATT (FFN, Ensemble)	**83.6**	**84.7**	**84.1**	**68.7**	83.3	**84.5**	**83.9**	**69.3**

Table 2: Experimental results on the CoNLL-2012 dataset. The metrics are the same as above. Again, our model achieves the state-of-the-art performance.

gle model of DEEPATT with RNN, CNN and FFN nonlinear sub-layers achieves an F_1 score of 82.3, 82.3 and 83.4 respectively. The FFN variant outperforms previous best performance by 1.8 F_1 score. Remarkably, we get 74.1 F_1 score on the out-of-domain dataset, which outperforms the previous state-of-the-art system by 2.0 F_1 score. On the CoNLL-2012 dataset, the single model of FFN variant also outperforms the previous state-of-the-art by 1.0 F_1 score. When ensembling 5 models with FFN nonlinear sub-layers, our approach achieves an F_1 score of 84.6 and 83.9 on the two datasets respectively, which has an absolute improvement of 1.4 and 0.5 over the previous state-of-the-art. These results are consistent with our intuition that the self-attention layers is helpful to capture structural information and long distance dependencies.

CoNLL-2005 Dataset						
#	Nonlinearity	PE	Embedding	Width	Depth	F_1
1	FFN	Timing	GloVe	200	10	83.1
2	FFN	Timing	GloVe	200	4	79.9
3	FFN	Timing	GloVe	200	6	82.0
4	FFN	Timing	GloVe	200	8	82.8
5	FFN	Timing	GloVe	200	12	83.0
6	FFN	Timing	GloVe	400	10	83.2
7	FFN	Timing	GloVe	600	10	83.4
8	FFN	Timing	Random	200	10	79.6
9	FFN	None	GloVe	200	10	20.0
10	FFN	Embedding	GloVe	200	10	79.4
11	None	Timing	GloVe	200	10	79.9

Table 3: Detailed results on the CoNLL-2005 development set. PE denotes the way to encoding word positions. GloVe refers to the GloVe embedding pre-trained on 6B tokens.

Analysis

In this subsection, we discuss the main factors that influence our results. We analyze the experimental results on the development set of CoNLL-2005 dataset.

Model Depth Previous works (Zhou and Xu 2015; He et al. 2017) show that model depth is the key to the success of end-to-end SRL approach. Our observations also coincide with previous works. Rows 1-5 of Table 3 show the effects of different number of layers. For DEEPATT with 4 layers, our model only achieves 79.9 F_1 score. Increasing depth consistently improves the performance on the development set, and our best model consists of 10 layers. For DEEPATT with 12 layers, we observe a slightly performance drop of 0.1 F_1.

Model Width We also conduct experiments with different model widths. We increase the number of hidden units from 200 to 400 and 400 to 600 as listed in rows 1, 6 and 7 of Table 3, and the corresponding hidden size h_f of FFN sub-layers is increased to 1600 and 2400 respectively. Increas-

Decoding	F_1	Speed
Argmax Decoding	**83.1**	**50K**
Constrained Decoding	83.0	17K

Table 4: Comparison between argmax decoding and constrained decoding on top of our model.

ing model widths improves the F_1 slightly, and the model with 600 hidden units achieves an F_1 of 83.4. However, the training and parsing speed are slower as a result of larger parameter counts.

Word Embedding Previous works found that the performance can be improved by pre-training the word embeddings on large unlabeled data (Collobert et al. 2011; Zhou and Xu 2015). We use the GloVe (Pennington, Socher, and Manning 2014) embeddings pre-trained on Wikipedia and Gigaword. The embeddings are used to initialize our networks, but are not fixed during training. Rows 1 and 8 of Table 3 show the effects of additional pre-trained embeddings. When using pre-trained GloVe embeddings, the F_1 score increases from 79.6 to 83.1.

Position Encoding From rows 1, 9 and 10 of Table 3 we can see that the position encoding plays an important role in the success of DEEPATT. Without position encoding, the DEEPATT with FFN sub-layers only achieves 20.0 F_1 score on the CoNLL-2005 development set. When using position embedding approach, the F_1 score boosts to 79.4. The timing approach is surprisingly effective, which outperforms the position embedding approach by 3.7 F_1 score.

Nonlinear Sub-Layers DEEPATT requires nonlinear sub-layers to enhance its expressive power. Row 11 of Table 3 shows the performance of DEEPATT without nonlinear sub-layers. We can see that the performance of 10 layered DEEPATT without nonlinear sub-layers only matches the 4 layered DEEPATT with FFN sub-layers, which indicates that the nonlinear sub-layers are the essential components of our attentional networks.

Constrained Decoding Table 4 show the effects of constrained decoding (He et al. 2017) on top of DEEPATT with FFN sub-layers. We observe a slightly performance drop when using constrained decoding. Moreover, adding constrained decoding slow down the decoding speed significantly. For DEEPATT, it is powerful enough to capture the relationships among labels.

Detailed Scores We list the detailed performance on frequent labels in Table 5. The results of the previous state-of-the-art (He et al. 2017) are also shown for comparison. Compared with He et al. (2017), our model shows improvement on all labels except AM-PNC, where He's model performs better. Table 6 shows the results of identifying and classifying semantic roles. Our model improves the previous state-of-the-art on both identifying correct spans as well as correctly classifying them into semantic roles. However, the majority of improvements come from classifying semantic roles. This indicates that finding the right constituents remains a bottleneck of our model.

Label	He et al. (2017)			DEEPATT		
	Precision	Recall	F_1	Precision	Recall	F_1
A0	89.6	90.7	90.2	91.0	91.9	91.4
A1	84.2	84.6	84.4	86.2	86.9	86.6
A2	73.6	72.7	73.2	75.0	77.0	76.0
A3	76.6	63.2	69.2	77.1	73.7	75.3
AM-ADV	64.8	60.6	62.6	69.5	64.5	66.9
AM-DIR	48.3	38.9	43.1	54.8	47.2	50.8
AM-LOC	59.8	58.3	59.0	62.5	61.9	62.2
AM-MNR	70.4	57.0	63.0	70.4	62.0	65.9
AM-PNC	65.8	64.2	65.0	65.3	60.5	62.8
AM-TMP	80.5	85.7	83.0	81.7	87.9	84.7
Overall	83.1	82.4	82.7	84.3	84.9	84.6

Table 5: Detailed scores on the development set of CoNLL-2005 dataset. We also list the previous state-of-the-art model (He et al. 2017) for comparison.

Model	Constituents	Semantic Roles
He et al. (2017)	91.87	87.10
DEEPATT	**91.92**	**88.88**

Table 6: Comparison with the previous work on identifying and classifying semantic roles. We list the percentage of correctly identified spans as well as the percentage of correctly classified semantic roles given the gold spans.

pred./gold	A0	A1	A2	A3	ADV	DIR	LOC	MNR	PNC	TMP
A0	-	48	12	7	0	0	0	0	7	0
A1	76	-	35	0	7	16	19	2	30	0
A2	10	37	-	42	15	33	28	35	38	0
A3	0	0	5	-	0	0	0	2	7	0
ADV	0	0	0	0	-	0	9	26	15	58
DIR	0	0	2	0	0	-	9	2	0	0
LOC	10	10	10	7	11	16	-	15	0	8
MNR	0	0	22	28	26	0	4	-	0	33
PNC	0	0	10	7	0	0	4	2	-	0
TMP	3	2	2	7	38	33	23	13	0	-

Table 7: Confusion matrix for labeling errors. Each cell shows the percentage of predicted labels for each gold label.

Labeling Confusion Table 7 shows a confusion matrix of our model for the most frequent labels. We only consider predicted arguments that match gold span boundaries. Compared with the previous work (He et al. 2017), our model still confuses ARG2 with AM-DIR, AM-LOC and AM-MNR, but to a lesser extent. This indicates that our model has some advantages on such difficult adjunct distinction (Kingsbury, Palmer, and Marcus 2002).

Related work

SRL Gildea and Jurafsky (2002) developed the first automatic semantic role labeling system based on FrameNet. Since then the task has received a tremendous amount of attention. The focus of traditional approaches is devising appropriate feature templates to describe the latent structure of utterances. Pradhan et al. (2005); Surdeanu et al. (2003);

Palmer, Gildea, and Xue (2010) explored the syntactic features for capturing the overall sentence structure. Combination of different syntactic parsers was also proposed to avoid prediction risk which was introduced by Surdeanu et al. (2003); Koomen et al. (2005); Pradhan et al. (2013).

Beyond these traditional methods above, Collobert et al. (2011) proposed a convolutional neural network for SRL to reduce the feature engineering. The pioneering work on building an end-to-end system was proposed by Zhou and Xu (2015), who applied an 8 layered LSTM model which outperformed the previous state-of-the-art system. He et al.(2017) improved further with highway LSTMs and constrained decoding. They used simplified input and output layers compared with Zhou and Xu (2015). Marcheggiani, Frolov, Titov (2017) also proposed a bidirectional LSTM based model. Without using any syntactic information, their approach achieved the state-of-the-art result on the CoNLL-2009 dataset.

Our method differs from them significantly. We choose self-attention as the key component in our architecture instead of LSTMs. Like He et al. (2017), our system take the very original utterances and predicate masks as the inputs without context windows. At the inference stage, we apply argmax decoding approach on top of a simple logistic regression while Zhou and Xu (2015) chose a CRF approach and He et al. (2017) chose constrained decoding. This approach is much simpler and faster than the previous approaches.

Self-Attention Self-attention have been successfully used in several tasks. Cheng, Dong, and Lapata (2016) used LSTMs and self-attention to facilitate the task of machine reading. Parikh et al. (2016) utilized self-attention to the task of natural language inference. Lin et al. (2017) proposed self-attentive sentence embedding and applied them to author profiling, sentiment analysis and textual entailment. Paulus, Xiong, and Socher (2017) combined reinforcement learning and self-attention to capture the long distance dependencies nature of abstractive summarization. Vaswani et al. (2017) applied self-attention to neural machine translation and achieved the state-of-the-art results. Very recently, Shen et al. (2017) applied self-attention to language understanding task and achieved the state-of-the-art on various datasets. Our work follows this line to apply self-attention for learning long distance dependencies. Our experiments also show the effectiveness of self-attention mechanism on the sequence labeling task.

Conclusion

We proposed a deep attentional neural network for the task of semantic role labeling. We trained our SRL models with a depth of 10 and evaluated them on the CoNLL-2005 shared task dataset and the CoNLL-2012 shared task dataset. Our experimental results indicate that our models substantially improve SRL performances, leading to the new state-of-the-art.

Acknowledgements

This work was done while the first author's internship at Tencent Technology. This work is supported by the Natural Science Foundation of China (Grant No. 61573294, 61303082, 61672440), the Ph.D. Programs Foundation of Ministry of Education of China (Grant No. 20130121110040), the Foundation of the State Language Commission of China (Grant No. WT135-10) and the Natural Science Foundation of Fujian Province (Grant No. 2016J05161). We also thank the anonymous reviews for their valuable suggestions.

References

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bastianelli, E.; Castellucci, G.; Croce, D.; and Basili, R. 2013. Textual inference and meaning representation in human robot interaction. In *Proceedings of the Joint Symposium on Semantic Processing. Textual Inference and Structures in Corpora*, 65–69.

Bengio, Y.; Ducharme, R.; Vincent, P.; and Janvin, C. 2003. A neural probabilistic language model. *Journal of Machine Learning Research* 3:1137–1155.

Carreras, X., and Màrquez, L. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, 152–164.

Cheng, J.; Dong, L.; and Lapata, M. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *Journal of Marchine Learning Research* 2493–2537.

Dan, S., and Lapata, M. 2007. Using semantic roles to improve question answering. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Dauphin, Y. N.; Fan, A.; Auli, M.; and Grangier, D. 2016. Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083*.

FitzGerald, N.; Täckström, O.; Ganchev, K.; and Das, D. 2015. Semantic role labeling with neural network factors. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*.

Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. N. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Genest, P.-E., and Lapalme, G. 2011. Framework for abstractive summarization using text-to-text generation. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, 64–73. Association for Computational Linguistics.

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

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

He, L.; Lee, K.; Lewis, M.; and Zettlemoyer, L. 2017. Deep semantic role labeling: What works and whats next. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Kingsbury, P.; Palmer, M.; and Marcus, M. 2002. Adding semantic annotation to the penn treebank. In *Proceedins of the human language technology conference*.

Knight, K., and Luk, S. K. 1994. Building a large-scale knowledge base for machine translation. In *AAAI*, volume 94, 773–778.

Koomen, P.; Punyakanok, V.; Roth, D.; and Yih, W.-t. 2005. Generalized inference with multiple semantic role labeling systems. In *Proceedings of the 9th Conference on Computational Natural Language Learning*, 181–184.

Lin, Z.; Feng, M.; Santos, C. N. d.; Yu, M.; Xiang, B.; Zhou, B.; and Bengio, Y. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Luong, M.-T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Marcheggiani, D.; Frolov, A.; and Titov, I. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. *arXiv preprint arXiv:1701.02593*.

Moschitti, A.; Morarescu, P.; and Harabagiu, S. M. 2003. Open domain information extraction via automatic semantic labeling. In *FLAIRS Conference'03*, 397–401.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807–814.

Palmer, M.; Gildea, D.; and Xue, N. 2010. *Semantic Role Labeling*. Synthesis Lectures on Human Language Technology Series. Morgan and Claypool.

Parikh, A. P.; Täckström, O.; Das, D.; and Uszkoreit, J. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Pascanu, R.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

Paulus, R.; Xiong, C.; and Socher, R. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*, 1532–1543.

Pradhan, S.; Hacioglu, K.; Ward, W.; Martin, J. H.; and Jurafsky, D. 2005. Semantic role chunking combining complementary syntactic views. In *Proceedings of the 9th Conference on Computational Natural Language Learning*, 217–220.

Pradhan, S.; Moschitti, A.; Xue, N.; Ng, H. T.; Björkelund, A.; Uryupina, O.; Zhang, Y.; and Zhong, Z. 2013. Towards robust linguistic analysis using ontonotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, 143–152.

Punyakanok, V.; Roth, D.; and tau Yih, W. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational linguistics* 6(9).

Shen, T.; Zhou, T.; Long, G.; Jiang, J.; Pan, S.; and Zhang, C. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *arXiv preprint arXiv:1709.04696*.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15(1):1929–1958.

Surdeanu, M.; Harabagiu, S.; Williams, J.; and Aarseth, P. 2003. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, 8–15.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2818–2826.

Täckström, O.; Ganchev, K.; and Das, D. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics* 3:29–41.

Toutanova, K.; Haghighi, A.; and Manning, C. D. 2008. A global joint model for semantic role labeling. *Computational Linguistics* 34:161–191.

Ueffing, N.; Haffari, G.; and Sarkar, A. 2007. Transductive learning for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 25–32.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Wu, D., and Fung, P. 2009. Semantic roles for smt: a hybrid two-pass model. In *Proceedings of Human Language Technologies*, 13–16. Association for Computational Linguistics.

Zeiler, M. D. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhou, J., and Xu, W. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.