Self-attentive Biaffine Dependency Parsing

Ying Li[1] , **Zhenghua Li**[1*] , **Min Zhang**[1] , **Rui Wang**[2] , **Sheng Li**[1] and **Luo Si**[2]

[1]Institute of Artificial Intelligence, School of Computer Science and Technology, Soochow University
[2]Alibaba Group, China

yingli_hlt@foxmail.com, {zhli13, minzhang}@suda.edu.cn,
lisheng.ls89@gmail.com, {masi.wr, luo.si}@alibaba-inc.com

Abstract

The current state-of-the-art dependency parsing approaches employ BiLSTMs to encode input sentences. Motivated by the success of the transformer-based machine translation, this work for the first time applies the self-attention mechanism to dependency parsing as the replacement of BiLSTM, leading to competitive performance on both English and Chinese benchmark data. Based on detailed error analysis, we then combine the power of both BiLSTM and self-attention via model ensembles, demonstrating their complementary capability of capturing contextual information. Finally, we explore the recently proposed contextualized word representations as extra input features, and further improve the parsing performance.

1 Introduction

As a fundamental task in NLP, dependency parsing has attracted a lot of research interest during the past decades due to its simplicity and multi-lingual applicability in capturing both syntactic and semantic information. Given an input sentence $\mathbf{s} = w_0 w_1 \dots w_n$, a dependency tree, as depicted in Figure 1, is defined as $\mathbf{d} = \{(h, m, l), 0 \leq h \leq n, 1 \leq m \leq n, l \in \mathcal{L}\}$, where (h, m, l) is a dependency from the head word w_h to the modifier word w_m with the relation label $l \in \mathcal{L}$, and w_0 is a pseudo word that points to the root word.

In recent years, neural network based approaches have achieved remarkable improvement and outperformed the traditional discrete-feature based approaches in dependency parsing by a large margin [Chen and Manning, 2014; Dyer *et al.*, 2015]. Most remarkably, Kiperwasser and Goldberg [2016] utilize BiLSTM as an encoder and achieve excellent results in both graph-based and transfer-based parsers. Based on this work, Dozat and Manning [2017] propose a simple yet effective deep biaffine graph-based parser and achieve the state-of-the-art accuracy on a variety of datasets and languages. The key of the biaffine parser is the use of multi-layer BiLSTMs to fully and globally encode the input sentence, along with reasonable design of scoring architecture and good settings of hyperparameters such as dropouts and initialization choices.

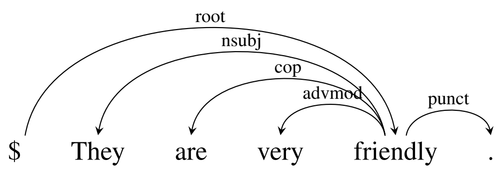

Figure 1: An example dependency parse tree.

As an alternative for encoding input sequences, the self-attention mechanism has proven to be especially effective and achieved much higher performance on machine translation [Vaswani *et al.*, 2017], known as the transformer. Intuitively, the self-attention mechanism is more suitable for capturing long-distance dependencies due to its capability of directly building connections between distant word pairs via attending. By contrast, the long-distance information may fade in the encoding process of BiLSTM. From another perspective, self-attention is also more efficient than the sequential BiLSTM since the attention computations of different tokens are independent and thus parallelizable.

In this work, we for the first time apply the self-attention-based encoder to dependency parsing as the replacement of BiLSTMs and make an in-depth study on the the differences between the two techniques. Our experiments on both the English and Chinese datasets demonstrate that the two encoders achieve similar overall performance, but further detailed analysis reveals a lot of local divergence with regard to dependency distances.

We then employ model ensemble to combine the power of the self-attention and BiLSTM encoders, leading to consistent improvements over the homogeneous ensembles, which indicates that the two encoders are complementary in capturing contextual information.

Finally, we further improve parsing performance by making good use of external contextualized word representations (ELMo & BERT), and the new state-of-the-art parsing accuracy of 95.22 on English and 89.90 on Chinese. We will also release all codes at https://github.com/SUDA-LA/IJCAI2019-dp-sa.

[*]Corresponding author

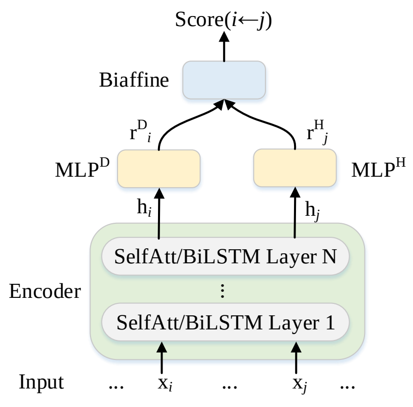

Figure 2: The framework of the biaffine parsing model

2 The Biaffine Parser

Given an input sentence $\mathbf{s} = w_0 w_1 \ldots w_n$, where w_0 is a pseudo word that represents the root node of the parse tree. The goal of dependency parsing is to find the highest-scoring dependency tree $\mathbf{d}^*$.

$$\mathbf{d}^* = \arg\max_{\mathbf{d}} score(\mathbf{s}, \mathbf{d}) \tag{1}$$

There exist two paradigms of dependency parsing, i.e., graph-based and transition-based approaches, depending on the definition of the scoring function $score(\mathbf{s}, \mathbf{d})$. In this work, we adopt the state-of-the-art deep biaffine parser as our basic parsing framework, which belongs to the graph-based paradigm and is proposed by Dozat and Manning [2017].

2.1 The Neural Network Architecture

Figure 2 shows the basic framework of the biaffine parser.

The input layer maps each input word w_i into a dense vector representation $\mathbf{x}_i$. In the original biaffine parser, $\mathbf{x}_i$ is the concatenation of the word and the POS-tag embedding.

$$\mathbf{x}_i = \mathbf{emb}^{\texttt{word}}_{w_i} \oplus \mathbf{emb}^{\texttt{tag}}_{t_i} \tag{2}$$

where $\mathbf{emb}^{\texttt{word}}_{w_i}$ is the pre-trained word embedding and $\mathbf{emb}^{\texttt{tag}}_{t_i}$ is the randomly initialized POS-tag embedding, which are both fine-tuned during training.

The encoder layer takes $\mathbf{x}_0\mathbf{x}_1 \ldots \mathbf{x}_n$ as the input dense vectors, and produces context-aware word representations $\mathbf{h}_i$ of all positions $0 \leq i \leq n$. Presumably, the produced representation $\mathbf{h}_i$ retains task-related sentence-level information and long-distance dependencies for word w_i after proper training.

The biaffine parser of Dozat and Manning [2017] employs a multi-layer BiLSTM encoder. The first-layer BiLSTM sequentially encodes the input $\mathbf{x}_0\mathbf{x}_1 \ldots \mathbf{x}_n$ in two independent directions, and the concatenated outputs of both directions are used as input for the second-layer BiLSTM, and so on. Finally, the outputs of the top-layer BiLSTM are used as the context-aware word representations $\mathbf{h}_i$. We omit the detailed equations of BiLSTMs due to space limitation.

In this work, we propose to employ the self-attention-based encoder instead of BiLSTMs, which will be introduced in Section 3.1 in detail.

The MLP representation layer takes the context-aware word representation $\mathbf{h}_i$ as input, and use two separate MLPs to get two lower-dimensional representation vectors for each position $0 \leq i \leq n$.

$$\begin{aligned} \mathbf{r}^{\mathrm{H}}_i &= \mathrm{MLP}^{\mathrm{H}}(\mathbf{h}_i) \\ \mathbf{r}^{\mathrm{D}}_i &= \mathrm{MLP}^{\mathrm{D}}(\mathbf{h}_i) \end{aligned} \tag{3}$$

where $\mathbf{r}^{\mathrm{H}}_i$ is the representation vector of w_i as a head word, and $\mathbf{r}^{\mathrm{D}}_i$ as a dependent. The MLP layer on the one hand reduces the dimension of $\mathbf{h}_i$, and more importantly on the other hand detains only syntax-related information in $\mathbf{r}_i$.

The biaffine scoring layer computes scores of all dependencies via a biaffine operation.

$$\texttt{score}(i \leftarrow j) = \begin{bmatrix} \mathbf{r}^{\mathrm{D}}_i \\ 1 \end{bmatrix}^{\mathrm{T}} \mathbf{W}^b \mathbf{r}^{\mathrm{H}}_j \tag{4}$$

where $\texttt{score}(i \leftarrow j)$ is the score of the dependency (j, i) and the matrix $\mathbf{W}^b$ is a biaffine parameter. Please note that scores of all dependencies can be efficiently computed at the same time in the matrix form.

2.2 Cross-entropy Training Loss

The parser defines a local cross-entropy loss for each position i. Assuming w_j is the gold-standard head of w_i, the corresponding loss is

$$loss(\mathbf{s}, i) = -\log \frac{e^{\texttt{score}(i \leftarrow j)}}{\sum_{0 \leq k \leq n, k \neq i} e^{\texttt{score}(i \leftarrow k)}} \tag{5}$$

2.3 Viterbi Decoding

Given the scores of all dependencies, the arc-factorization score of a dependency tree is

$$score(\mathbf{s}, \mathbf{d}) = \sum_{(j,i) \in \mathbf{d}} score(i \leftarrow j) \tag{6}$$

The highest-scoring tree $\mathbf{d}^*$ can be decoded with the dynamic programming algorithm known as maximum spanning tree [McDonald *et al.*, 2005].

2.4 Dependency Label Handling

The biaffine parser treats the classification of dependency labels as a separate task after finding $\mathbf{d}^*$. We denote the score of assigning a label l to a dependency $i \leftarrow j$ as $score(i \leftarrow j, l)$, which is also computed with biaffine operations. The unlabeled parsing task and the label classification task share most parameters except for the MLPs and biaffines.

3 Our Approach

In this work, we first propose to use the self-attention mechanism as the replacement of the BiLSTM-based encoder in dependency parsing. Then we explore the encoding capacity of both by model ensembles. Furthermore, we make in-depth empirical study on the use of the recently proposed contextualized word representations.

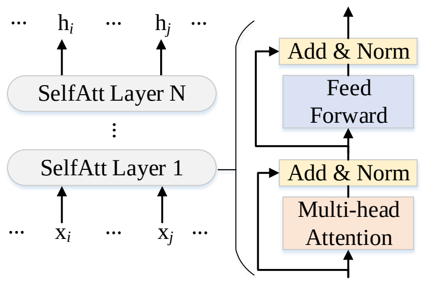

Figure 3: An overview of the self-attention encoder.

3.1 Self-Attention Encoder

As an encoding mechanism, self-attention has been successfully applied in several tasks. This work aims to make full investigation on how to effectively use self-attention in dependency parsing.

Similar to BiLSTM, the self-attention-based encoder takes $\mathbf{x}_0\mathbf{x}_1...\mathbf{x}_n$ as the inputs, and produces context-aware word representations $\mathbf{r}_i$ of all positions $0 \le i \le n$. We employ a stack of N identical self-attention layers, each having independent parameters. Figure 3 illustrates the structure of an N-layered self-attention-based encoder. Each layer is composed of a multi-head attention sublayer and a feed-forward sublayer. Residual connection and layer normalization are applied to the outputs of the two sublayers.

In the following, we illustrate the first layer of the self-attention-based encoder in detail:

Position Embedding as Extra Input

Unlike BiLSTMs, the self-attention mechanism need to use an extra position embedding $\mathbf{emb}_i^{\texttt{pos}}$, in order to inject relative or absolute position information.

$$\mathbf{x}_i = (\mathbf{emb}_{w_i}^{\texttt{word}} \oplus \mathbf{emb}_{t_i}^{\texttt{tag}}) + \mathbf{emb}_i^{\texttt{pos}} \tag{7}$$

where $\mathbf{emb}_i^{\texttt{pos}}$ is called the position embedding of the i-th position. We follow Vaswani *et al.* [2017] and use the sine and cosine functions of different frequencies to compose position embeddings.

$$\begin{aligned}\mathbf{emb}_i^{\texttt{pos}}[2j] &= sin(\frac{i}{10000^{2j/d_{\texttt{model}}}})\\ \mathbf{emb}_i^{\texttt{pos}}[2j+1] &= cos(\frac{i}{10000^{2j/d_{\texttt{model}}}})\end{aligned} \tag{8}$$

where $2j$ and $2j+1$ are the even and odd indices of the position embedding, and $d_{\texttt{model}}$ is the dimension of $\mathbf{emb}_i^{\texttt{pos}}$.[1] In this way, each dimension of the positional encoding corresponds to a sinusoid, allowing the model to easily learn to attend to relative positions.

Multi-Head Attention Sublayer

The inputs $\mathbf{x}_i$ $(0 \le i \le n)$ are then fed into a multi-head attention layer. We use $\mathbf{X} \in \mathcal{R}^{(n+1)\times d_{\texttt{model}}}$ to represent the inputs in the matrix form.

[1] We have also tried using a learned table of position embeddings as Kitaev and Klein [2018] did, but found a slight accuracy drop.

The basic idea of the self-attention mechanism is to reconstruct a representation vector for each position by attending to and aggregating the inputs of all positions. To be clearer, we begin with the single-head attention mechanism.

First, each input $\mathbf{x}_i$ is mapped into three vectors.

$$\begin{aligned}\mathbf{q}_i &= \mathbf{x}_i\mathbf{W}_q\\ \mathbf{k}_i &= \mathbf{x}_i\mathbf{W}_k\\ \mathbf{v}_i &= \mathbf{x}_i\mathbf{W}_v\end{aligned} \tag{9}$$

The resulting three vectors are called query, key, value vectors respectively, and have the same dimension (denoted as $d_{\texttt{head}}$).

Then, the query vector of the focused position i attends to the key vectors of all positions, and the value vectors of all positions are aggregated according to normalized attention weights, producing the single-head representation of i (of dimension $d_{\texttt{head}}$).

$$\texttt{SingleHead}(\mathbf{X}, i) = \sum_{j=0}^{n} \mathbf{v}_j \times \frac{e^{\mathbf{q}_i \cdot \mathbf{k}_j / \sqrt{d_{\texttt{head}}}}}{Z} \tag{10}$$

where $Z = \sum_{j=0}^{n} e^{\mathbf{q}_i \cdot \mathbf{k}_j / \sqrt{d_{\texttt{head}}}}$ is the normalization factor, also known as a softmax process.

Multi-head attention uses m separate sets of projection matrices $\mathbf{W}_{q/k/v}^{1..m}$ to produce m independent representations, which are then concatenated as the representation of i.

$$\texttt{MultiHead}(\mathbf{X}, i) = \text{Concate}(\texttt{SingleHead}^{1..m}(\mathbf{X}, i)) \tag{11}$$

Compared with single-head attention, multi-head attention is proven to be more effective due to the capability of representing the same position from different perspectives. Then, a linear map is used to mix different channels from different heads for each position i, following Tan *et al.* [2018].

$$\mathbf{a}_i = \texttt{MultiHead}(\mathbf{X}, i)\mathbf{W}_o \tag{12}$$

where $\mathbf{W}_o \in \mathcal{R}^{m d_{\texttt{head}} \times d_{\texttt{model}}}$.

Finally, residual connection and layer normalization [Ba *et al.*, 2016] are performed.

$$\mathbf{y}_i = \texttt{LayerNorm}(\mathbf{a}_i + \mathbf{x_i}) \tag{13}$$

Position-Wise Feed-Forward Sublayer

The outputs of the multi-head attention sublayer are then fed into a fully connected feed-forward network for each position i.

$$\mathbf{f}_i = \mathbf{W}_2\texttt{ReLU}(\mathbf{W}_1\mathbf{y}_i + \mathbf{b}_1) + \mathbf{b}_2 \tag{14}$$

where $\mathbf{W}_1 \in \mathcal{R}^{d_{\texttt{ff}}\times d_{\texttt{model}}}$, $\mathbf{W}_2 \in \mathcal{R}^{d_{\texttt{model}}\times d_{\texttt{ff}}}$, $\mathbf{b}_1 \in \mathcal{R}^{d_{\texttt{ff}}}$, and $\mathbf{b}_2 \in \mathcal{R}^{d_{\texttt{model}}}$ are the parameters.

Similar to the multi-head attention sublayer, residual connection and layer normalization are also performed.

$$\mathbf{z}_i = \texttt{LayerNorm}(\mathbf{f}_i + \mathbf{y_i}) \tag{15}$$

3.2 Model Ensemble

Model ensemble has been extensively adopted in real-life applications and shared task competitions [Petrov and McDonald, 2012; Zeman *et al.*, 2018]. In this work, we employ ensemble to combine the power of both self-attention- and BiLSTM-based encoders due to their divergent but complementary capability of capturing contextual information.

In particular, we use the product of experts approach [Hinton, 2002] to combine predictions of 6 models. First, we separately train 6 parsers in the n-fold jack-knifing way, where each parser is trained on a 5/6 subset of the whole training data. During evaluation, we employ the 6 parsers to independently produce a probability for each dependency. We then use the averaged probability as the final probability of a dependency. Finally, we employ Viterbi decoding to produce the optimal tree according to the aggregated probabilities.

3.3 Contextualized Word Representations

Recently proposed contextualized word representations, such as ELMo [Peters *et al.*, 2018] and BERT [Devlin *et al.*, 2018], have been shown to be extremely helpful in a variety of NLP tasks. The basic idea is to train a multi-layer LSTM- or self-attention-based encoder on large-scale unlabeled texts with the natural language model (or next-sentence prediction) loss.

In this work, we replace the word embeddings with the contextualized word representations as extra features without fine-tuning. We make a thorough comparison between ELMo and BERT for dependency parsing, and also experiment with different ways to combine the different encoder layers.

4 Experiments

We conduct experiments on the English Penn Treebank (PTB) dataset with Stanford dependencies and the Chinese dataset of the 2009 CoNLL shared task. For PTB, we directly use the data of Chen and Manning [2014], and they use the Stanford POS tagger. For Chinese CoNLL-2009, we use the data split and POS tags provided by the organizers [Hajic *et al.*, 2009].

For evaluation metrics, we use the standard labeled attachment score (LAS, percent of words that receives correct heads and labels) and unlabeled attachment score (UAS, ignoring labels). Following previous work [Dozat and Manning, 2017], we ignore all punctuation marks for PTB and keep them for the Chinese CoNLL-2009 dataset.

Each parser is trained for at most $1,000$ iterations, and the performance is evaluated on the dev data after each iteration for model selection. We stop the training if the peak performance does not increase in 100 consecutive iterations.

For pre-trained word embeddings, we directly adopt the GloVe embeddings released by Stanford for English[2], and train word2vec embeddings on Chinese Gigaword Third Edition for Chinese.

We directly use the English ELMo model released by AllenNLP[3]. For Chinese, we train ELMo for 10 iterations on the Chinese Gigaword Third Edition, consisting of about 1.2 million sentences. It takes about 7 days using 6 GPU nodes (GTX 1080Ti). For BERT, we use the released BERT-Base models for English (uncased) and Chinese.[4]

[2]https://nlp.stanford.edu/projects/glove/

[3]https://github.com/allenai/allennlp/blob/master/tutorials/how_to/elmo.md

[4]https://github.com/google-research/bert

Depth N	English		Chinese	
	UAS	LAS	UAS	LAS
10	95.57	93.72	88.35	85.20
8	**95.67**	**93.83**	**88.52**	**85.36**
6	95.45	93.55	88.30	85.12
4	95.31	93.33	87.90	84.66

Table 1: Results on the dev data regarding the depth (layer number N) of the self-attention encoder.

4.1 Hyperparameter Choices

We mainly follow the work of Dozat and Manning [2017] for most of the hyperparameter settings, which uses three BiLSTM layers as the encoder. The dimensions of the word/tag/position embeddings are 100/100/200; $d_{\texttt{model}}$ is 200; $d_{\texttt{ff}}$ is 800; head number m is 8, and thus $d_{\texttt{head}}$ is 25; dropout ratio before residual connection is 0.2; dropout ratios before entering multi-head attention and feed-forward are both 0.1; all other dropout ratios are 0.33; $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-6}$ for the Adam optimizer.

Preliminary experiments show that the our parser using the self-attention encoder is insensitive to most of the above parameters, while the number of self-attention layers and the learning rate have larger impact on the performance as shown in the following results.

Depth of the Self-attention Encoder (N)

Table 1 shows the results with different numbers of self-attention layers on the dev data. Increasing the self-attention layer number from 4 to 8 consistently improves the performance on both English and Chinese datasets. The performance drops slightly when using 10 self-attention layers. These results indicate that the use of more self-attention layers increases the complexity of the model, allowing it to capture richer contextual information, but also makes the model prone to overfit the training data, which is consistent with previous findings [Vaswani *et al.*, 2017; Tan *et al.*, 2018].

Learning Rate

The results on the dev data with different learning rates are shown in Table 2. When we use the same learning rate 0.002 (or larger) as the original biaffine parser of Dozat and Manning [2017], the model converges quickly but achieves unsatisfactory performance. Reducing the value to certain extent improves the parsing accuracy. Based on the results, we set the learning rate to 0.0012 for English and 0.0011 for Chinese afterwards.

4.2 Contextualized Word Representations

Instead of using the pretrained word embedding, we also experiment with contextualized word representations from both ELMo and BERT.

The results of **ELMo** are shown in Table 3. We can see that it leads to much better performance using only the third (top) layer outputs of ELMo than the first (bottom) layer outputs. We then use the weighted sum of all three layers, with the weight values learned during training, leading to further improvement on both English and Chinese. However, it is

	English		Chinese	
	UAS	LAS	UAS	LAS
0.003	94.70	92.65	86.14	82.54
0.002	95.16	93.22	87.44	84.17
0.0015	95.29	93.45	87.99	84.90
0.0012	**95.67**	**93.83**	88.39	85.23
0.0011	95.58	93.71	**88.52**	**85.36**
0.001	95.42	93.57	88.16	85.03
0.0005	94.88	92.77	87.96	84.75

Table 2: The effect of different learning rates on the dev data.

	English		Chinese	
	UAS	LAS	UAS	LAS
Basic Parser	95.67	93.83	88.52	85.36
Replace $\mathbf{emb}^{\text{word}}$ with ELMo outputs				
Only layer #3	96.12	94.23	89.40	86.15
Only layer #2	95.98	94.07	89.55	86.21
Only layer #1	95.60	93.75	87.35	84.02
Weighted sum	96.36	94.54	90.16	87.06
Averaged sum	**96.40**	**94.56**	**90.25**	**87.27**
Concatenate $\mathbf{emb}^{\text{word}}$ with ELMo outputs				
Averaged sum	96.22	94.37	89.41	86.32
Replace $\mathbf{emb}^{\text{word}}$ with BERT outputs				
Averaged sum	**96.57**	**94.69**	**92.01**	**89.20**

Table 3: Results of the self-attentive parser on the dev data regarding the different ways to use ELMo/BERT outputs.

interesting to see that the simple averaged sum even slightly outperforms the weighted sum.

Instead of replacing word embeddings with ELMo outputs, we also try to directly combine them, which decreases the performance, especially for Chinese. The reason may be that it is difficult for the model to reach a balance between word embeddings as shallow and context-free representations and ELMo outputs as deep and contextualized representations.

For **BERT**, We conduct the same experiments and find the trends are similar: using the averaged sum of the top-4 layer outputs is the best. We can see that BERT can greatly boost parsing accuracy over ELMo by about 2% for Chinese. The gain is much smaller (about $0.1 - 0.2\%$) for English since the performance is already at a high level.

4.3 Model Ensemble

The results for the model ensemble experiments are shown in Table 4. 3×SelfAtt+3×BiLSTM means among six training data folds, three are used to train three parsers with the self-attention encoder, and the remaining three are for those with the BiLSTM encoder.

First, compared with the results of single self-attentive parsers in Table 3, model ensemble can consistently improve parsing accuracy on both languages. For example, 6×SelfAtt with BERT outperforms the single self-attentive parser with BERT by 0.18 (94.87 vs. 94.69) on English and 0.53 (89.73 vs. 89.20) on Chinese in LAS.

	English		Chinese	
	UAS	LAS	UAS	LAS
Ensemble of basic parsers				
6×SelfAtt	95.76	93.97	88.68	85.47
6×BiLSTM	95.76	94.01	88.63	85.44
3×SelfAtt+3×BiLSTM	**95.89**	**94.12**	**88.90**	**85.77**
Ensemble of parsers with ELMo outputs				
6×SelfAtt	96.40	94.56	90.51	87.41
6×BiLSTM	96.42	94.65	90.34	87.37
3×SelfAtt+3×BiLSTM	**96.54**	**94.74**	**90.72**	**87.72**
Ensemble of parsers with BERT outputs				
6×SelfAtt	96.63	94.87	92.39	89.62
6×BiLSTM	96.61	94.80	92.20	89.45
3×SelfAtt+3×BiLSTM	**96.66**	**94.91**	**92.50**	**89.73**

Table 4: Ensemble results on the dev data.

	English		Chinese	
	UAS	LAS	UAS	LAS
Hajic *et al.* [2009]	-	-	-	76.51
K&G [2016]	93.20	91.20	-	-
Andor *et al.* [2016]	94.61	92.79	84.72	80.85
Ma *et al.* [2018]	95.87	94.19	-	-
Clark *et al.* [2018]	**96.60**	**95.00**	-	-
BiLSTM (D&M [2017])	95.74	94.08	88.90	85.38
BiLSTM (our impl)	95.79	94.08	88.70	85.50
SelfAtt	95.93	94.19	88.77	85.58
SelfAtt w/ ELMo	96.59	94.91	90.32	87.27
SelfAtt w/ BERT	**96.67**	**95.03**	**92.24**	**89.29**
Ensemble	96.15	94.49	89.26	86.08
Ensemble w/ ELMo	96.68	95.07	90.76	87.75
Ensemble w/ BERT	**96.83**	**95.22**	**92.76**	**89.90**

Table 5: Final results on the test data.

Second, we can see that the 6×SelfAtt parsers achieve very similar results (slightly higher in most cases) compared with the 6×BiLSTM, showing that the two frameworks have similar power of encoding contextual information.

Finally and most importantly, we find that the hybrid 3×SelfAtt+3×BiLSTM ensembles achieve consistently higher accuracy than the homogeneous 6×SelfAtt/BiLSTM ensembles. This clearly demonstrates that the two encoding techniques are complementary and can certainly benefit from each other.

4.4 Final Results on the Test Data

Table 5 shows the final results and makes comparison with previous works on the test data. Our implemented biaffine parser with the BiLSTM-based encoder achieves slightly higher performance than the original results reported in Dozat and Manning [2017], and our proposed self-attentive parser outperforms our BiLSTM parser by small margin. In the ensemble setting (3×SelfAtt+3×BiLSTM), the ensemble models with BERT achieve the best performance, leading to new

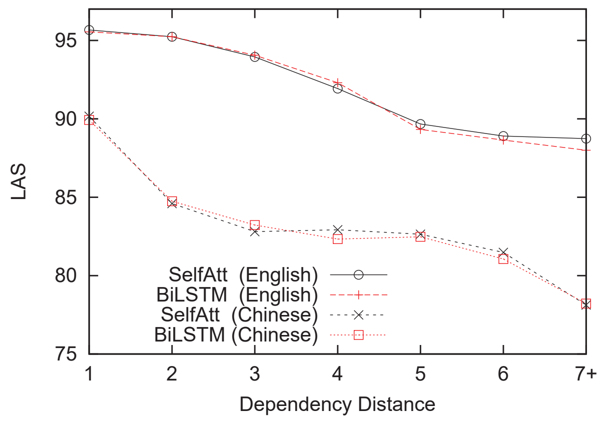

Figure 4: Accuracy curves regarding dependency distances.

state-of-the-art results on both languages.

4.5 Analysis

Although the self-attention and BiLSTM encoders achieve very close overall parsing accuracy, we conduct more detailed analysis in order to gain more insights on the differences between them. We divide the gold-standard dependencies into seven subsets according to the absolute distance between the two words, and calculate the accuracy for each subset. Figure 4 compares the accuracy curves of the two single parsers with the self-attention or BiLSTM encoder with regard to the dependency distance on the test data. On the one hand, we can see that the two encoders actually exhibit divergent encoding power for dependencies of different distances. On the other hand, there seems no clear pattern on the wins and loses, indicating that self-attention and BiLSTM, as two different encoders, both have the competitive capability of capturing short- and long-range dependencies.

5 Related Work

LSTM, as the mainstream encoding technique, has been extensively used in many NLP tasks including parsing. Dyer *et al.* [2015] first apply LSTMs to encode history actions and partially built trees in a transition-based dependency parser. Kiperwasser and Goldberg [2016] utilize BiLSTM to encode the input sentence for both transition-based and graph-based dependency parsing. As the baseline of this work, Dozat and Manning [2017] propose a graph-based biaffine dependency parser which largely relies on a multi-layer BiLSTM encoder.

Self-attention has been successfully applied to neural machine translation by Vaswani *et al.* [2017], leading to large improvement on translation quality, known as the transformer. Foster *et al.* [2018] extend the transformer for machine translation by combining BiLSTM and self-attention encoders in a cascaded stacking architecture. In this work, we have also tried their method in dependency parsing, but found little improvement. So far, self-attention has been successfully applied to a variety of NLP tasks, such as language understanding [Shen *et al.*, 2018] and semantic role labeling [Tan *et al.*, 2018]. Most closely related to this work, Kitaev and Klein [2018] replace BiLSTM with self-attention in the constituent parser of Stern *et al.* [2017]. They propose two useful techniques to improve parsing performance, i.e., factored content and position attention and replacing POS-tag embeddings with CharLSTM word representations. In this work, we have also implemented the two techniques of Kitaev and Klein [2018], but our preliminary experiments show they are not helpful for dependency parsing.

Contextualized word representations are first proposed by Peters *et al.* [2018]. They train LSTM encoders on large-scale unlabeled data with language model loss to obtain context-aware word representations, known as ELMo, which is shown to be extensively helpful for many NLP tasks. Radford *et al.* [2018] propose to pre-train transformer instead of the LSTM encoder on large-scale unlabeled corpus with language model loss, and finetune the model on the labeled data of the focused task, known as GPT. Devlin *et al.* [2018] further extend the idea of ELMo and GPT by 1) replacing the less efficient BiLSTM with self-attention, 2) using both bidirectional masked language model loss and next-sentence prediction loss, and 3) using a larger model and much more data. As another interesting direction, Clark *et al.* [2018] propose a semi-supervised learning framework to learn from both labeled and unlabeled data by designing clever auxiliary tasks, instead of using language model loss, leading to very promising performance on English dependency parsing.

Model ensemble is a commonly used strategy to integrate different parsing models in traditional discrete-feature dependency parsing [Nivre and McDonald, 2008]. Kuncoro *et al.* [2016] propose an knowledge distilling approach to train a graph-based neural parser on the voting results of many transition-based neural parsers trained with different initialization starts. Liu *et al.* [2018] propose to distill knowledge from the local classifiers employed by the transition-based parser.

6 Conclusions

This work for the first time applies the self-attention encoder to dependency parsing under the state-of-the-art graph-based biaffine parsing framework, obtaining competitive performance on both English and Chinese benchmark data. The comparative analysis shows that in spite of local divergences, the two encoders actually both have the power of modeling short- or long-range dependencies, which motivates us to extend our experiments into model ensembles. The hybrid ensemble models achieve consistently higher performance than the homogeneous ensemble ones, demonstrating that self-attention and BiLSTM are complementary in encoding contextual information. Furthermore, we find that proper utilization of the contextualized word representations substantially improves parsing accuracy, leading to new state-of-the-art parsing performance.

Acknowledgments

We thank our anonymous reviewers for their helpful comments. This work was supported by National Natural Science Foundation of China (Grant No. 61525205, 61876116, 61432013), and was partially supported by the joint research project of Alibaba and Soochow University.

References

[Andor *et al.*, 2016] Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. In *Proceedings of ACL*, pages 2442–2452, 2016.

[Ba *et al.*, 2016] Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

[Chen and Manning, 2014] Danqi Chen and Christopher D. Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*, pages 740–750, 2014.

[Clark *et al.*, 2018] Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc V. Le. Semi-supervised sequence modeling with cross-view training. In *Proceedings of EMNLP*, pages 1914–1925, 2018.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Dozat and Manning, 2017] Timothy Dozat and Christopher Manning. Deep biaffine attention for neural dependency parsing. In *Proceedings of ICLR*, 2017.

[Dyer *et al.*, 2015] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL*, pages 334–343, 2015.

[Foster *et al.*, 2018] George Foster, Ashish Vaswani, Jakob Uszkoreit, Wolfgang Macherey, Lukasz Kaiser, Orhan Firat, Llion Jones, Noam Shazeer, Yonghui Wu, Ankur Bapna, Melvin Johnson, Mike Schuster, Zhifeng Chen, Macduff Hughes, Niki Parmar, and Mia Xu Chen. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of ACL*, pages 76–86, 2018.

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

[Hinton, 2002] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[Kiperwasser and Goldberg, 2016] Eliyahu Kiperwasser and Yoav Goldberg. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *CoRR*, abs/1603.04351, 2016.

[Kitaev and Klein, 2018] Nikita Kitaev and Dan Klein. Constituency parsing with a self-attentive encoder. In *Proceedings of ACL*, pages 2675–2685, 2018.

[Kuncoro *et al.*, 2016] Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. Distilling an ensemble of greedy dependency parsers into one MST parser. In *Proceedings of EMNLP*, pages 1744–1753, 2016.

[Liu *et al.*, 2018] Yijia Liu, Wanxiang Che, Huaipeng Zhao, Bing Qin, and Ting Liu. Distilling knowledge for search-based structured prediction. In *Proceedings of ACL*, 2018.

[Ma *et al.*, 2018] Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard H. Hovy. Stack-pointer networks for dependency parsing. In *Proceedings of ACL*, pages 1403–1414, 2018.

[McDonald *et al.*, 2005] Ryan T. McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT/EMNLP*, pages 523–530, 2005.

[Nivre and McDonald, 2008] Joakim Nivre and Ryan McDonald. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL*, pages 950–958, 2008.

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

[Petrov and McDonald, 2012] Slav Petrov and Ryan McDonald. Overview of the 2012 shared task on parsing the web. In *Proceedings of SANCL*, 2012.

[Radford *et al.*, 2018] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Technical Report*, 2018.

[Shen *et al.*, 2018] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Proceedings of AAAI*, pages 5446–5455, 2018.

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

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

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of NIPS*, pages 6000–6010, 2017.

[Zeman *et al.*, 2018] Daniel Zeman, Jan Haji{č}, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. Conll 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of CoNLL*, pages 1–21, 2018.