Yang LE, Sun MS, Cheng Y *et al.* Neural parse combination. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 32(4): 749–757 July 2017. DOI 10.1007/s11390-017-1756-5

Neural Parse Combination

Lin-Er Yang [1,2,3], *Student Member, CCF*, Mao-Song Sun [1,2,3,4], *Senior Member, CCF*, Yong Cheng [5]
Jia-Cheng Zhang [1,2,3], *Student Member, CCF*, Zheng-Hao Liu [1,2,3], *Student Member, CCF*, Huan-Bo Luan [1,2,3]
and Yang Liu [1,2,3,4,*], *Senior Member, CCF*

[1] *Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China*

[2] *State Key Laboratory of Intelligent Technology and Systems, Tsinghua University, Beijing 100084, China*

[3] *Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing 100084, China*

[4] *Jiangsu Collaborative Innovation Center for Language Ability, Jiangsu Normal University, Xuzhou 221009, China*

[5] *Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing 100084, China*

E-mail: lineryang@gmail.com; sms@mail.tsinghua.edu.cn; chengyong3001@gmail.com; grit31@126.com
liuzhenghao0819@163.com; luanhuanbo@gmail.com; liuyang2011@tsinghua.edu.cn

Received December 22, 2016; revised June 12, 2017.

Abstract Analyzing the syntactic structure of natural languages by parsing is an important task in artificial intelligence. Due to the complexity of natural languages, individual parsers tend to make different yet complementary errors. We propose a neural network based approach to combine parses from different parsers to yield a more accurate parse than individual ones. Unlike conventional approaches, our method directly transforms linearized candidate parses into the ground-truth parse. Experiments on the Penn English Treebank show that the proposed method improves over a state-of-the-art parser combination approach significantly.

Keywords neural network, parsing, parse combination

1 Introduction

Parsing, which aims to analyze the syntactic structure of nature languages conforming to formal grammars, is an important task in artificial intelligence. The past two decades have witnessed the rapid development of natural language parsing. While early parsers rely on either generative or discriminative statistical models[1-8], neural network based approaches to parsing[9-16] have received increasing attention recently.

Despite the apparent success of data-driven parsing, it is widely observed that an individual parser can only capture partial aspects of parsing regularities due to the complexity of natural languages. In constituency parsing, Henderson and Brill[17] found that isolated constituent precision varies significantly across three constituency parsers[1,18]. In dependency parsing, McDonald and Nivre[19] observed that the graph-based MSTParser[4] and the transition-based MaltParser[6] make different errors: while MSTParser has an advantage for the root category, MaltParser achieves considerably better precision for nominal categories. Fortunately, errors made by different parsers are usually observed to be complementary[19], suggesting that integrating different parsers can potentially yield better parses.

Therefore, a number of authors have explored approaches to parser combination[17,19-25]. These approaches can be roughly divided into two broad categories: parse combination and model combination. While parse combination approaches which include parse selection and parse hybridization, focus on com-

Regular Paper
Special Issue on Deep Learning
This work is supported by the National Basic Research 973 Program of China under Grant No. 2014CB340501, the Key Program of the National Natural Science Foundation of China under Grant No. 61331013, and the National Natural Science Foundation of China under Grant No. 61522204.

*Corresponding Author

bining parses either at the parse level or at the constituent level in the post-processing phase[17,20-23,25], model combination approaches integrate different parsing models at the learning phase[19,24]. As compared with model combination that requires candidate models to be combinable, parse combination can be more easily applied to integrate the output of arbitrary parsers.

However, parse combination still suffers from a major problem: it is difficult to find the ground-truth parse in the search space of combination. For parse-level combination, the search space only consists of a handful of candidate single parses in which the ground-truth parse is usually not included[17]. Enlarging the search space by organizing it as a weighted parse chart to represent more candidates, constituent-level combination is more likely to contain the ground-truth parse. However, it is still impossible to yield the ground-truth parse if there exist ground-truth constituents excluded in candidate parses.

In this work, we propose a neural network based approach to parse combination. Inspired by Vinyals *et al.*[12], we propose a multi-sequence-to-sequence model: the input is a list of linearized parses produced by different parsers and the output is the ground-truth parse. Unlike conventional parse combination approaches, our model learns to combine candidate parses directly into the ground-truth parse. We leverage LSTM (Long-Short Term Memory)[26] to capture long-distance contextual information and use the attention mechanism[27] to dynamically select relevant parts from candidate parses. Experiments on the Penn English Treebank① show that our method is superior to a state-of-the-art parse combination approach.

2 Background

2.1 Problem Statement

The goal of parse combination is to integrate the strengths of individual parsers to improve parsing accuracy. For example, Fig.1(a) shows a candidate parse produced by one parser. While the parse correctly identifies "Making computers smaller" as a verb phrase (i.e., VP), its prediction on "means sacrificing memory" is wrong. On the contrary, as shown in Fig.1(b), the candidate produced by another parser correctly analyzes "means sacrificing memory" but is wrong with "Making computer smaller". Since the two candidate parses are complementary, combining them can potentially yield the ground-truth parse shown in Fig.1(c).

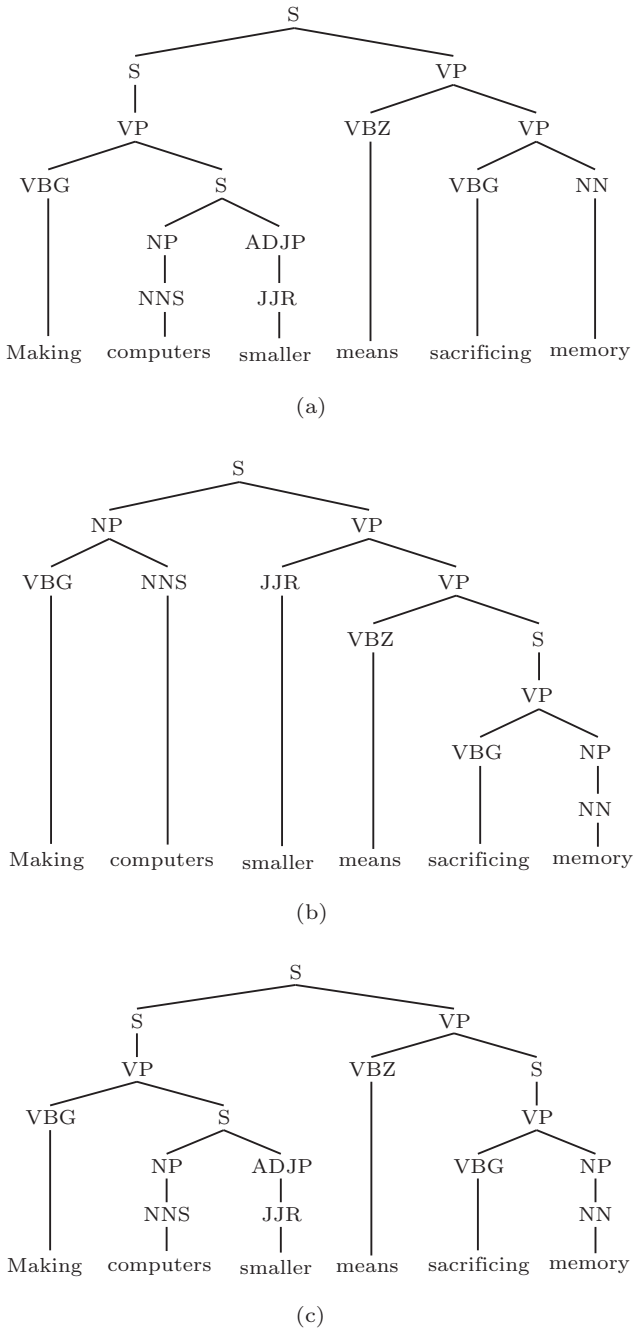


Fig.1. (a) One candidate parse. (b) Another candidate parse. (c) Combined parse. Parser combination aims to integrate the strengths of individual parsers to yield more accurate parses.

2.2 Parse Selection

To combine parses, a straightforward idea is to select a single promising parse from the candidate set, which is often referred to as parse selection. This can be done either by maximizing the expected precision or

①https://catalog.ldc.upenn.edu/LDC99T42, June 2017.

F_1 score of the selected parse[17,21] or by discriminative reranking with hand-crafted features[22-23].

A key limitation of such parse-level combination is that the search space often rules out the ground-truth parses. For example, as neither Fig.1(a) nor Fig.1(b) is identical to the ground-truth parse, it is impossible for parse selection approaches to yield the correct parse.

2.3 Parse Hybridization

To alleviate this problem, a number of researchers proposed parse hybridization approaches that combine parses at the constituent level[17,20-21,25]. The central idea is to choose constituents on which all parses agree. For example, in Fig.1(a) and Fig.1(b), both parses assign "sacrificing memory" the same constituent $(\text{VP}, 4, 6)$, suggesting that it is likely to be correct because two parses reach a consensus.

More formally, given N parses $\{T^{(1)}, \ldots, T^{(N)}\}$ and their associated probabilities, a parse hybridization model as described in [20] computes the weight of a constituent by summing weights of all parses that contain the constituent as follows:

$$w((\ell, i, j)) = \sum_{n=1}^{N} P(T^{(n)}) [\![(\ell, i, j) \in T^{(n)}]\!],$$

where $w((l, i, j))$ is the weight of a constituent spanning from the i-th word to the j-th word with label ℓ, $P(T^{(n)})$ is the probability of the n-th candidate parse, and $[\![(\ell, i, j) \in T^{(n)}]\!]$ returns 1 if $T^{(n)}$ contains the constituent and 0 otherwise.

After calculating constituent weights, Sagae and Lavie[20] populated a chart with high-weight constituents and used a CKY-style parse algorithm to find the heaviest tree. One advantage of this approach over parse selection is the enlarged search space: new trees that are not included in the candidate set can be generated.

Although parse hybridization is superior to parse selection by introducing agreement between candidates at a smaller granularity, it is still difficult to yield the ground-truth parse if constituents in the ground-truth parse are not observed in candidate parses. In addition, the weight of constituents is simply calculated by voting and ignores contexts. Fossum and Knight[21] indicated that parse hybridization approaches allow arbitrary context-free productions because parsing with weighted constituents is not constrained by formal grammars, which may hinder downstream applications.

3 Neural Parse Combination

3.1 Model

In this work, we propose a neural network based approach to parse combination. The basic idea is to utilize a single, large neural network to directly transform multiple candidate parses to a combined parse. This can be done by linearizing parses[12] and extending the attention-based sequence-to-sequence architecture[27] to take multiple sequences as input and the ground-truth parse as output.

Given N candidate parses $\{T^{(1)}, \ldots, T^{(N)}\}$, we first follow Vinyals *et al.*[12] to obtain their linearized forms. For example, Fig.1(a) can be linearized as

$$\text{(S (S (VP VBG (S (NP NNS)NP ....}$$

Unlike [12] that aims to transform a natural language sentence into a linearized parse, our goal is to combine multiple linearized parses into one linearized parse.

Formally, given a set of linearized candidate parses $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\}$ and a linearized ground-truth parse $\boldsymbol{y}$, the neural parse combination model is defined as

$$\begin{aligned} & P(\boldsymbol{y}|\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}; \boldsymbol{\theta}) \\ = & \prod_{i=1}^{J} P(y_i|\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}, \boldsymbol{y}_{<j}; \boldsymbol{\theta}), \end{aligned} \tag{1}$$

where y_i is the i-th token in the linearized ground-truth parse, $\boldsymbol{y}_{<i} = \{y_1, \ldots, y_{i-1}\}$ is a partial output sequence, and $\boldsymbol{\theta}$ is a set of model parameters.

Following Bahdanau *et al.*[27], we define each conditional probability in (1) as

$$\begin{aligned} & P(y_i|\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}, \boldsymbol{y}_{<i}; \boldsymbol{\theta}) \\ = & \frac{\exp\big(g(y_{i-1}, \boldsymbol{c}_i, \boldsymbol{s}_i, \boldsymbol{\theta})\big)}{\sum_{y'} \exp\big(g(y', \boldsymbol{c}_i, \boldsymbol{s}_i, \boldsymbol{\theta})\big)}, \end{aligned}$$

where $g(\cdot)$ is a non-linear function, $\boldsymbol{c}_i$ is a context vector that encodes the relevant context on the input side (i.e., $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\}$) and $\boldsymbol{s}_i$ is a hidden state for time i that represents the context on the output side (i.e., $\boldsymbol{y}_{<i}$).

The hidden state $\boldsymbol{s}_i$ is calculated by a recurrent neural network (RNN) using the LSTM unit[26]:

$$\boldsymbol{s}_i = f(\boldsymbol{s}_{i-1}, y_{i-1}, \boldsymbol{c}_i),$$

where $f(\cdot)$ is a non-linear function.

3.2 Calculating Attention Scores

On the input side, for each sequence $\boldsymbol{x}^{(n)}$ that has $J^{(n)}$ tokens, we follow Bahdanau *et al.*[27] to use bidirectional RNNs to concatenate forward and backward states to obtain annotations $\{\boldsymbol{h}_j^{(n)}\}_{j=1}^{J^{(n)}}$. The method of calculating the context vector $\boldsymbol{c}_i$, however, is significantly different from that of Bahdanau *et al.*[27] because the input is a list of sequences rather than a single sequence. Therefore, we propose two approaches to calculating attention as well as the context vector.

3.2.1 Local Attention

Taking multiple sequences as input in the encoder-decoder framework has been investigated in machine translation[28-29]. Zoph and Knight[28] proposed a method to integrate locally normalized attention for multi-source neural machine translation.

As shown in Fig.2(a), we follow Zoph and Knight[28] to calculate local attention for each input sequence $\boldsymbol{x}^{(n)}$:

$$\alpha_{ij}^{(n)} = \frac{\exp\left(\boldsymbol{a}(\boldsymbol{s}_{i-1}, \boldsymbol{h}_j^{(n)})\right)}{\sum_{j'=1}^{J^{(n)}} \exp\left(\boldsymbol{a}(\boldsymbol{s}_{i-1}, \boldsymbol{h}_{j'}^{(n)})\right)},$$

where $\boldsymbol{a}(\cdot)$ is an alignment matrix.

The corresponding local context vector $\boldsymbol{c}_i^{(n)}$ is computed as a weighted sum of annotations:

$$\boldsymbol{c}_i^{(n)} = \sum_{j=1}^{J^{(n)}} \alpha_{ij}^{(n)} \boldsymbol{h}_j^{(n)}.$$

Then, the global context vector $\boldsymbol{c}_i$ can be obtained by averaging local context vectors:

$$\boldsymbol{c}_i = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{c}_i^{(n)}.$$

3.2.2 Global Attention

Inspired by [29] that introduces a single attention mechanism shared across all language pairs in multi-way, multilingual neural machine translation, we propose an approach to directly compute global attention for all input sequences:

$$\alpha_{ij}^{(n)} = \frac{\exp\left(\boldsymbol{a}(\boldsymbol{s}_{i-1}, \boldsymbol{h}_j^{(n)})\right)}{\sum_{n=1}^{N} \sum_{j'=1}^{J^{(n)}} \exp\left(\boldsymbol{a}(\boldsymbol{s}_{i-1}, \boldsymbol{h}_{j'}^{(n)})\right)}.$$

The global context vector $\boldsymbol{c}_i$ is given by

$$\boldsymbol{c}_i = \sum_{n=1}^{N} \sum_{j=1}^{J^{(n)}} \alpha_{ij}^{(n)} \boldsymbol{h}_j^{(n)}.$$

3.3 Training and Parsing

In training, we use the standard maximum likelihood criterion to find optimal model parameters:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \left\{ P(\boldsymbol{y} | \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}; \boldsymbol{\theta}) \right\}.$$

Given trained model parameters, we use a beam search algorithm[27] to find a linearized parse that approximately maximizes the conditional probability:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} \left\{ P(\boldsymbol{y} | \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}; \hat{\boldsymbol{\theta}}) \right\}.$$

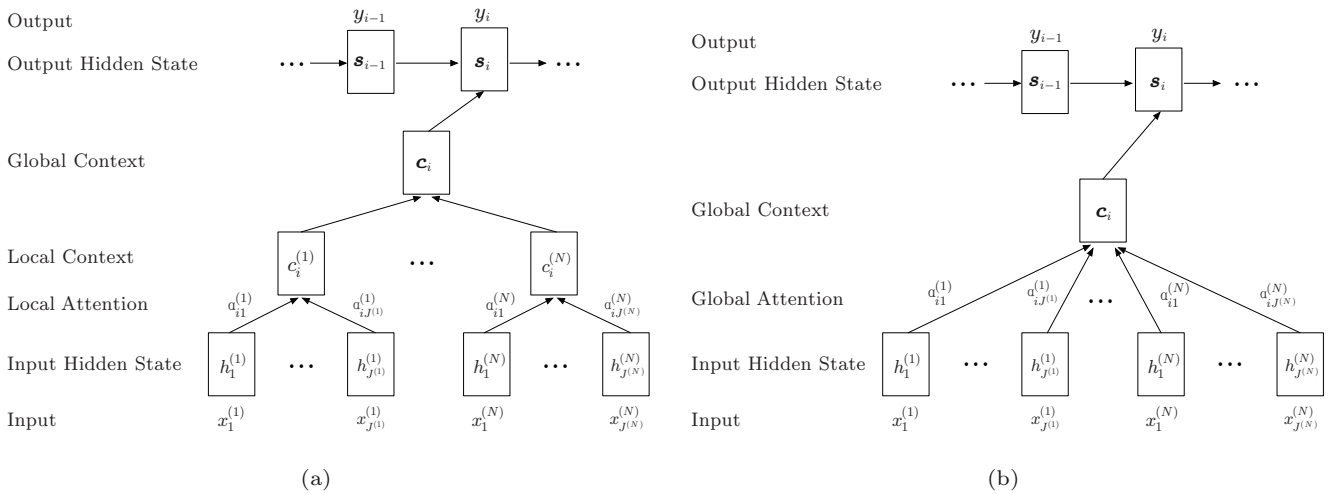


Fig.2. (a) Local attention. (b) Global attention. While local attention is normalized within each candidate parse, global attention is distributed among all candidate parses.

4 Experiments

4.1 Setup

We evaluate our approach on the Penn English Treebank② using the standard split: sections 02∼21 as the training set (39 832 sentences), section 22 as the development set (1 700 sentences), and section 23 as the test set (2 416 sentences). We tag the training data using Stanford POS tagger[30] with 10-way jackknifing (accuracy ≈ 97.3%).

The following two constituency parsers are used in our parse combination experiments:

1) Berkeley[5]: a state-of-the-art generative parser using latent-variable PCFGs in which non-terminals are refined by a split-merge procedure;

2) ZPar[31]: a state-of-the-art discriminative parser using global discriminative training and shift-reduce parsing with beam search.

Following Collins and Koo[32], we run both parsers on a 10-fold cross-validated training set. The original training set is divided into 10 equalized folds (i.e., around 3 983 sentences in each fold on average). The parses for each fold are generated by Berkeley and ZPar trained on the trees in the other folds. Then, the development and test sets are parsed using the two "out-of-the-box" parsers.

We compare the following two parse combination approaches:

1) ReParsing[20]: a state-of-the-art parse hybridization approach that reparses with weighted constituents (see Section 2);

2) NeuralComb: the proposed approach that exploits the attention-based encoder-decoder framework to combine candidate parses.

NeuralComb uses a training set consisting of two parts: the candidate parses produced by Berkeley and ZPar as input and the ground-truth parses in the treebank as output. We implement our approach using TensorFlow. In our experiments, we use a model with three LSTM layers and 256 units in each layer. The vocabulary size is set to 132. We use the Adam optimizer[33] for stochastic optimization with an initial learning rate of 0.001. We also apply dropout[34] with probability 0.5 to LSTM layers and gradient clipping[35] when the norm of gradients exceeds 5. We train our model on a single GPU (GeForce GTX TITAN X) for up to 120 000 epochs, with a rough runtime of 30 minutes per 1 000 epochs. The beam size for searching the optimal combined parse in the decoder is set to 10. The model that achieves the highest accuracy on the development set is selected for evaluation on the test set.

4.2 Complementariness of Individual Parsers

Our work is based on the assumption that two individual parsers are complementary. To validate this hypothesis, we plot the F_1 scores achieved by Berkeley and ZPar on the development set, as shown in Fig.3. For each point that denotes a sentence in the development set, the horizontal axis denotes the F_1 score of Berkeley on the sentence and the vertical axis denotes the F_1 score of ZPar on the same sentence. We observe that Berkeley achieves the same F_1 score with ZPar on 1 032 sentences, better on 644 sentences, and worse on 740 sentences. This finding suggests that two parsers are complementary and combining them can potentially yield better parses.

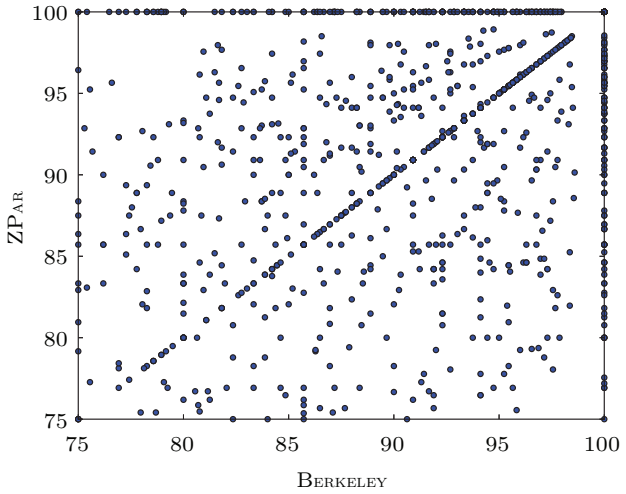


Fig.3. Plots of F_1 scores of Berkeley and ZPar on the development set.

4.3 Local and Global Attention

We first compare local and global attention models on the development set. As shown in Table 1, using local attention achieves surprisingly much higher precision, recall and F_1 than using global attention. We conjecture that it is more difficult to model global attention because the concatenated sequence is much longer than individual sequences. Therefore, we use local attention in the following experiments.

②https://catalog.ldc.upenn.edu/LDC99T42, June 2017.

Table 1. Comparison Between Local and Global Attention on the Development Set

Attention	Precision	Recall	F_1
Local	91.34	90.39	90.88
Global	90.04	89.60	89.82

4.4 Effect of Beam Size

We investigate the effect of beam size on the development set. As shown in Table 2, with the increase of beam size, the precision, the recall, and F_1 rise significantly. However, a larger beam size also results in longer parsing time. To achieve a balance between accuracy and efficiency, we set the beam size to 10 in the following experiments.

Table 2. Effect of Beam Size on the Development Set

Beam Size	Precision	Recall	F_1
1	91.34	90.39	90.88
5	91.68	90.80	91.24
10	91.82	90.83	91.32

4.5 Comparison with Parse Hybridization

Table 3 shows the F_1 scores achieved by individual and combined parsers on the development and test sets. BERKELEY and ZPAR achieve similar results around 89.7. REPARSING achieves an F_1 score of 90.35 by reparsing with weighted constituents collected from the parses produced by the two individual parsers. Our approach NEURALCOMB improves over individual parsers by +1.1% and REPARSING by 0.3%. All the differences are statistically significant.

Table 3. Comparison of F_1 Scores Between Individual and Combined Parsers

Type	Parser	Development Set	Test Set
Individual	BERKELEY[5]	90.13	89.77
	ZPAR[31]	90.34	89.73
Combined	REPARSING[20]	90.41	90.35
	NEURALCOMB	91.32	**90.86**

4.6 F_1 Scores over Sentence Lengths

A key limitation of REPARSING is that the constituents used in reparsing are restricted to the set of constituents from BERKELEY and ZPAR parses.

We find that 2 488 constituents in the ground-truth parses are not included in the candidate constituent set on the test set. In contrast, NEURALCOMB is capable of recovering 504 such unseen constituents thanks to the attention-based encoder-decoder architecture.

Fig.4 shows the F_1 scores of BERKELEY and ZPAR on the test set with respect to sentence lengths. We find that our approach generally outperforms BERKELEY, ZPAR, and REPARSING on all sentence lengths. The margin seems to rise with the increase of sentence lengths. However, slight oscillation is observed for sentences longer than 50 words. One possible reason is that it is difficult for LSTMs and the attention mechanism to handle very long sentences. Note that the length of a linearized parse is usually much longer than that of the corresponding sentence.

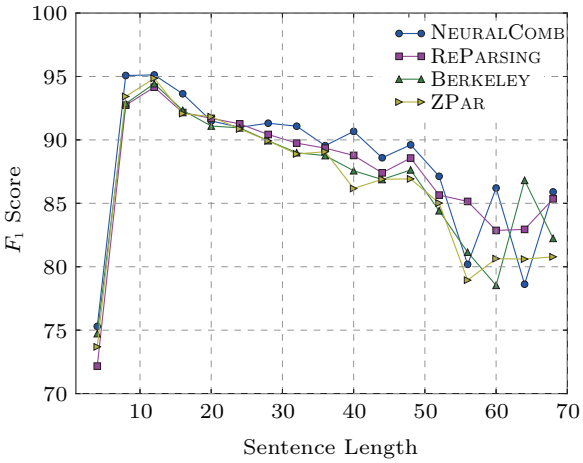


Fig.4. F_1 scores on the test set over various sentence lengths.

4.7 Comparison of Constituent Precision

To measure how precise a parser predicts a constituent label, we compute constituent precision as follows:

$$\text{Precision}_\ell = \frac{\sum_{k=1}^{K}\sum_{\langle \ell', i, j\rangle \in \tilde{T}^{(k)}} [\![\ell' = \ell \wedge \langle \ell', i, j\rangle \in T^{(k)}]\!]}{\sum_{k=1}^{K}\sum_{\langle \ell', i, j\rangle \in \tilde{T}^{(k)}} [\![\ell' = \ell]\!]},$$

where ℓ is a constituent label (e.g., NP), K is the number of sentences in a dataset, $\tilde{T}^{(k)}$ is the k-th parse produced by the parser, and $T^{(k)}$ is the k-th ground-truth parse.

Table 4 shows constituent precision for all four parsers. We can reconfirm that BERKELEY and ZPAR are complementary: while BERKELEY excels at predicting "FRAG", "S", "NP", "VP", and "WHNP", ZPAR is superior on "CONJP", "LST", "NAC", "NX", and "PP". Clearly, our approach improves precision for most labels, especially for high-frequency labels such as "NP", "VP", "S", "PP", and "SBAR".

Table 4. Comparison of Constituent Precision

Label	Berkeley		ZPar		ReParsing		NeuralComb	
	Count	Precision	Count	Precision	Count	Precision	Count	Precision
ADJP	626	77.28	617	77.61	662	73.15	649	82.99
ADVP	1 186	85.82	1 180	88.92	1 241	83.51	1 218	90.49
CONJP	18	69.23	10	83.33	16	72.73	17	77.27
FRAG	9	28.13	3	15.79	9	27.27	10	83.33
INTJ	8	66.67	5	71.43	9	69.23	10	90.91
LST	4	50.00	1	100.00	4	50.00	1	100.00
NAC	20	74.07	19	82.61	18	66.66	21	95.45
NP	17 027	91.58	17 161	91.51	17 509	89.54	17 288	93.05
NX	19	48.72	15	78.95	21	52.50	32	71.11
PP	4 707	85.09	4 832	87.49	4 819	85.79	4 880	88.42
PRN	117	72.67	111	82.22	125	71.84	112	86.15
QP	437	88.46	439	88.33	438	87.95	444	90.24
S	5 125	90.95	5 109	90.52	5 180	90.04	5 200	92.66
SBAR	1 518	88.46	1 553	87.79	1 563	87.81	1 588	91.58
SBARQ	7	70.00	4	57.14	7	70.00	6	66.67
SINV	145	90.63	130	90.28	146	90.12	141	90.97
SQ	11	61.11	11	84.62	12	63.16	16	84.21
UCP	9	39.13	7	53.85	9	37.50	8	80.00
VP	8 013	91.31	7 950	90.94	8 126	90.00	8 148	93.09
WHADJP	2	100.00	3	100.00	2	100.00	2	100.00
WHADVP	120	96.00	124	96.88	124	95.28	124	97.64
WHNP	413	95.60	412	95.15	416	93.69	416	95.19
WHPP	24	96.00	25	100.00	24	100.00	25	100.00
X	2	40.00	4	57.14	4	40.00	4	100.00

5 Related Work

Our work is inspired by two lines of research: 1) parser combination, and 2) attention-based encoder-decoder framework.

5.1 Parser Combination

Previous work on parser combination can be divided into two categories: parse combination at the post-processing phase[17,20] and model combination at the learning phase[19,24]. Our approach falls into the first category. While previous parse combination approaches focus on picking single parses[17,23] or recombining constituents[20-21], our approach adopts the attention-based encoder-decoder framework[27] to directly transform multiple linearized parses. One major advantage is that our approach is capable of guiding the model to learn to directly produce ground-truth parses, which are usually ruled out by conventional parse combination approaches. Another difference is that we leverage the attention mechanism to automatically select relevant parts from candidate parses rather than to assign constituents weights by voting.

5.2 Attention-Based Encoder-Decoder Framework

Our work extends the standard attention-based encoder-decoder framework[27] by taking multiple sequences as input. We are also inspired by [12] that linearizes a parse into a sequence and introduces the first sequence-to-sequence parsing model. Our local attention model is similar to the multi-source attention proposed by Zoph and Knight[28] except that we use standard content-based attention instead of position-aware attention. Our global attention model is inspired by the shared attention mechanism described by Firat *et al.*[29] The difference is that parse combination only needs to yield one sequence while Firat *et al.*[29] focused on generating multiple sequences.

6 Conclusions

We presented a neural network based approach to parse combination. The central idea is to treat linearized parses as sequences and use the attention-based encoder-decoder architecture to directly transform multiple candidate parses into a single parse. One major

advantage of our approach is that the model learns how to directly yield ground-truth parses at the training phase. Experiments on Penn English Treebank showed that our approach significantly improves over the state-of-the-art reparsing approach.

In the future, we will investigate novel attention models to better capture combination regularities between input and output. It is also interesting to adopt the minimum risk training[36] to optimize our model directly with respect to F_1 score. As our framework can be applied to arbitrary sequences, we plan to verify its effectiveness on more natural language processing tasks.

References

[1] Collins M. Three generative, lexicalised models for statistical parsing. In *Proc. ACL*, July 1997, pp.16-23.

[2] Charniak E. A maximum-entropy-inspired parser. In *Proc. NAACL*, April 29-May 4, 2000, pp.132-139.

[3] Klein D, Manning C D. Accurate unlexicalized parsing. In *Proc. ACL*, July 2003, pp.423-430.

[4] McDonald R, Pereira F, Ribarov K, Hajič J. Non-projective dependency parsing using spanning tree algorithms. In *Proc. EMNLP*, October 2005, pp.523-530.

[5] Petrov S, Barrett L, Thibaux R, Klein D. Learning accurate, compact, and interpretable tree annotation. In *Proc. ACL*, July 2006, pp.433-440.

[6] Nivre J. Incremental non-projective dependency parsing. In *Proc. HLT-NAACL*, April 2007, pp.396-403.

[7] Huang L, Sagae K. Dynamic programming for linear-time incremental parsing. In *Proc. ACL*, July 2010, pp.1077-1086.

[8] Zhang Y, Clark S. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 2011, 37(1): 105-151.

[9] Socher R, Lin C, Ng A Y, Manning C D. Parsing natural scenes and natural language with recursive neural networks. In *Proc. ICML*, June 28-July 2, 2011, pp.129-136.

[10] Chen D, Manning C D. A fast and accurate dependency parser using neural networks. In *Proc. EMNLP*, October 2014, pp.740-750.

[11] Dyer C, Ballesteros M, Ling W, Matthews A, Smith N A. Transition-based dependency parisng with stack long short-memory. In *Proc. ACL*, July 2015, pp.334-343.

[12] Vinyals O, Kaiser L, Koo T, Petrov S, Sutskever I, Hinton G. Grammar as a foreign language. In *Proc. NIPS*, December 2015, pp.2773-2781.

[13] Weiss D, Alberti C, Collins M, Petrov S. Structured training for neural network transition-based parsing. In *Proc. ACL*, July 2015, pp.323-333.

[14] Andor D, Alberti C, Weiss D, Severyn A, Presta A, Ganchev K, Petrov S, Collins M. Globally normalized transition-based neural networks. In *Proc. ACL*, August 2016, pp.2442-2452.

[15] Dyer C, Kuncoro A, Ballesteros M, Smith N A. Recurrent neural network grammars. In *Proc. HLT-NAACL*, June 2016, pp.199-209.

[16] Kiperwasser E, Goldberg Y. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 2016, 4: 313-327.

[17] Henderson J C, Brill E. Exploiting diversity in natural language processing. In *Proc. EMNLP*, June 1999, pp.187-194.

[18] Charniak E. Statistical parsing with a context-free grammar and word statistics. In *Proc. AAAI*, July 1997, pp.598-603.

[19] McDonald R, Nivre J. Analyzing and integrating dependency parsers. *Computational Linguistics*, 2011, 37(1): 197-230.

[20] Sagae K, Lavie A. Parser combination by reparsing. In *Proc. HLT-NAACL*, June 2006, pp.129-132.

[21] Fossum V, Knight K. Combining constituent parsers. In *Proc. HLT-NAACL*, May 31-June 5, 2009, pp.253-256.

[22] Zhang H, Zhang M, Tan C L, Li H. K-best combination of syntactic parsers. In *Proc. EMNLP*, August 2009, pp.1552-1560.

[23] Johnson M, Ural A E. Reranking the Berkeley and Brown parsers. In *Proc. HLT-NAACL*, June 2010, pp.665-668.

[24] Narayan S, Cohen S B. Diversity in spectral learning for natural language parsing. In *Proc. EMNLP*, September 2015, pp.1868-1878.

[25] Choe D K, McClosky D, Charniark E. Syntactic parse fusion. In *Proc. EMNLP*, September 2015, pp.1360-1366.

[26] Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Computation*, 1997, 9(8): 1735-1780.

[27] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*, May 2015.

[28] Zoph B, Knight K. Multi-source neural translation. In *Proc. HLT-NAACL*, June 2016, pp.30-34.

[29] Firat O, Cho K, Bengio Y. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proc. HLT-NAACL*, June 2016, pp.866-875.

[30] Toutanova K, Klein D, Manning C D, Singer Y. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. HLT-NAACL*, May 27-June 1, 2003, pp.173-180.

[31] Zhu M, Zhang Y, Chen W, Zhang M, Zhu J. Fast and accurate shift-reduce constituent parsing. In *Proc. ACL*, August 2013, pp.434-443.

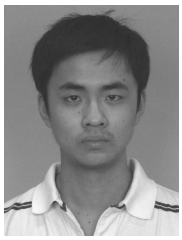
[32] Collins M, Koo T. Discriminative reranking for natural language parsing. *Computational Linguistics*, 2005, 31(1): 25-70.

[33] Kingma D, Ba J. Adam: A method for stochastic optimization. In *Proc. ICLR*, May 2015.

[34] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014, 15(1): 1929-1958.

[35] Pascanu R, Mikolov T, Bengio Y. On the difficulty of training recurrent neural networks. In *Proc. ICML*, June 2013, pp.1310-1318.

[36] Shen S, Cheng Y, He Z, He W, Wu H, Sun M, Liu Y. Minimum risk training for neural machine translation. In *Proc. ACL*, August 2016, pp.1683-1692.



Lin-Er Yang is a Ph.D. student at the Department of Computer Science and Technology in Tsinghua University, Beijing. He received his Bachelor's degree in computer science from Beijing Jiaotong University, Beijing, in 2006. His current research interests include representation learning and syntactic parsing.

Mao-Song Sun is a professor at the Department of Computer Science and Technology in Tsinghua University, Beijing. He received his Ph.D. degree in computational linguistics from City University of Hong Kong, Hong Kong, in 2004. His research interests include natural language processing, Web intelligence, and machine learning.

Yong Cheng is a Ph.D. student in the Institute for Interdisciplinary Information Sciences at Tsinghua University, Beijing. He received his Bachelor's degree in information security from Beijing Jiaotong University, Beijing, in 2012. His research interests mainly focus on neural machine translation.

Jia-Cheng Zhang is a master student at the Department of Computer Science and Technology in Tsinghua University, Beijing. He received his Bachelor's degree in computer science from Tsinghua University, Beijing, in 2015. His current research area is machine translation.

Zheng-Hao Liu is a Ph.D. student at the Department of Computer Science and Technology in Tsinghua University, Beijing. He got his Bachelor's degree in computer science from Northeastern University, Shenyang, in 2016. His current research interests include representation learning and syntactic parsing.

Huan-Bo Luan is the deputy executive director of NExT Search Center at both Tsinghua University, Beijing, and National University of Singapore, Singapore. He received his Ph.D. degree in computer science from Institute of Computing Technology, Chinese Academy of Sciences, Beijing, in 2008. His research interests include multimedia information retrieval, social media, and big data analysis.

Yang Liu is an associate professor in the Department of Computer Science and Technology at Tsinghua University, Beijing. He received his Ph.D. degree from Institute of Computing Technology, Chinese Academy of Sciences, Beijing, in 2007. His research areas include natural language processing and machine translation.