PAPER

Improving Feature-Rich Transition-Based Constituent Parsing Using Recurrent Neural Networks

Chunpeng MA[†*a)], **Akihiro TAMURA**[††b)], **Lemao LIU**[††**], **Tiejun ZHAO**[†], ***Nonmembers***, ***and*** **Eiichiro SUMITA**[††], ***Member***

SUMMARY Conventional feature-rich parsers based on manually tuned features have achieved state-of-the-art performance. However, these parsers are not good at handling long-term dependencies using only the clues captured by a prepared feature template. On the other hand, recurrent neural network (RNN)-based parsers can encode unbounded history information effectively, but they perform not well for small tree structures, especially when low-frequency words are involved, and they cannot use prior linguistic knowledge. In this paper, we propose a simple but effective framework to combine the merits of feature-rich transition-based parsers and RNNs. Specifically, the proposed framework incorporates RNN-based scores into the feature template used by a feature-rich parser. On English WSJ treebank and SPMRL 2014 German treebank, our framework achieves state-of-the-art performance (91.56 F-score for English and 83.06 F-score for German), without requiring any additional unlabeled data.

key words: *constituent parsing, recurrent neural network, system combination*

1. Introduction

Constituent parsing, which aims to represent the phrase structure of a sentence as a tree structure, is a fundamental task in natural language processing (NLP). Feature-rich parsing, either chart-based [1], [2] or transition-based [3]–[5], has yielded impressive results and been a dominant parsing method. Transition-based parsing has two advantages over chart-based parsing. First, its computational complexity is $O(n)$. Second, it can theoretically incorporate arbitrary feature templates, including both local and global features. These feature templates can efficiently encode prior linguistic knowledge for parsing. However, because of feature sparsity, only local features and quite limited global features are usually practically integrated into transition-based parsing, therefore this approach fails to capture long-term syntactic structures.

On the other hand, recently recurrent neural networks (RNNs) have been applied successfully to many NLP tasks, particularly to constituent parsing [6]–[9]. By encoding the unbounded history into a dense representation, RNNs are able to capture the long-term dependencies and thus make reliable parsing decisions. Despite these successes, it has been shown that neural networks (NNs) are not good at encoding low-frequency words [10], especially when the size of the training corpus is limited. Our experiments (see Sect. 5.3 for details) reveal that RNNs cannot perform well for small subtree structures containing low-frequency words (e.g. $X \Rightarrow w_{low\_freq}, w_{high\_freq}$, where X is the label of the root node of the subtree with w_{low_freq} and w_{high_freq} as two leaf nodes), which can be easily handled by feature-rich models using simple back-off features (e.g. features involving POS Tags). In addition, integrating prior linguistic knowledge is a non-trivial task for RNN models, especially given the inherent inefficiency of jointly training RNNs and feature-rich models.

This paper proposes a simple yet effective framework to avoid the respective problems of feature-rich models and RNN-based models, meanwhile retaining their advantages. The basic idea is quite simple: evaluating the whole parsing history using RNN models and then integrating these RNN-based evaluations into the feature-rich model as global features. In this way, the proposed model is able to not only capture long-term syntactic structures using the RNN but also make reliable parsing decisions for small subtree structures using feature templates incorporating linguistic knowledge. As an instance under this framework, this paper utilizes the sequence-to-sequence RNN model as the implementation of RNN models, and the transition-based parser [4] as a feature-rich model. Also, since the top-down linearization method used by the sequence-to-sequence RNN model in [8] is inconsistent with the transition-based incremental decoding, we propose several novel bottom-up linearization methods (Sect. 3). In order to capture the parsing history in multi-granularity, we further incorporate multiple RNN models, which are different in linearization methods, into the feature-rich transition model (Sect. 4.1). For the sake of training efficiency, we train the feature-rich and the RNN models seperately rather than jointly (Sect. 4.2).

Our study makes two contributions.

- We systematically and quantitively analyze the respective demerits of feature-rich transition-based parsers and RNN-based parsers. Based on the analysis, we propose a simple framework that incorporates RNNs into

Manuscript received January 4, 2017.
Manuscript revised April 14, 2017.
Manuscript publicized June 5, 2017.

[†]The authors are with the Machine Intelligence and Translation Laboratory, Harbin Institute of Technology, China.
[††]The authors are with the ASTREC, National Institute of Information and Communications Technology (NICT), Kyoto-fu, 619–0289 Japan.
[*]This work was done during the internship of the author at NICT.
[**]Presentky, with Tencent AI Lab, China.
a) E-mail: cpma@hit.edu.cn
b) E-mail: tamura@cs.ehime-u.ac.jp
DOI: 10.1587/transinf.2017EDP7003

a transition-based feature-rich constituent parser. Additionally, we explore several bottom-up linearization methods for incremental decoding in transition-based parsing.

- We show that the RNN model improves the performance of the baseline feature-rich parser significantly through experiments, demonstrating the effectiveness of our framework. In addition, the experiments show that the performance is further improved by combining different RNN models with different linearization methods, achieving state-of-the-art performance without any additional data, for both English and German.

The proposed framework is general to be applied on top of other RNN models in [7], [9], [11], although the sequence-to-sequence RNN model is instantiated in this paper.

2. Revisiting Feature-Rich Transition-Based Parsing

A transition-based parser processes a sentence through a sequence of transition actions between states. Given an input sentence $\mathbf{w} = \langle w_1, w_2, \cdots, w_{|\mathbf{w}|} \rangle$, the transition-based parser employs a stack of partially constructed constituent tree structures and a queue of input words. At each step, a transition action is applied to a state $[S, k, f, i, \rho]$, where S represents the stack of partially constructed trees, k indicates the index of the next input word w_k in the queue, f is a flag indicating the completion of parsing, i.e., whether the ROOT of a constituent tree covering all the input words has been generated, i is the total number of actions leading to the state, ρ is the scores of the state (to be defined later).

Figure 1 summarizes the transition action set of our parser, following [4]. Here s_t represents the t-th element from the top of the stack, and $S|X$ means that the root label of the subtree on the top of the stack S is X ($S|s_1 s_0$ can be interpreted similarly). "L" or "R" in the action **"reduce-L/R-X"** indicates that the head child used for feature extraction should be located in the left or right branch, respectively. ρ_{sh}, ρ_{re}, ρ_{un}, ρ_{fi}, and ρ_{id} represent the incremental scores of the different transition actions.

Note that **reduce-L/R-X** action operates on a pair of trees in the stack S, which implies that such a transition-based parser works on the binarizied constituency parse

$$\begin{array}{rl}
\text{input:} & w_1, w_2, \cdots, w_{|\mathbf{w}|} \\
\text{axiom:} & [\emptyset, 0, false, 0, 0] \\
\text{goal:} & [S, |\mathbf{w}|, true, m : 2|\mathbf{w}| \le m \le 4|\mathbf{w}|, C]
\end{array}$$

$$\begin{array}{rl}
\textbf{shift} & \dfrac{[S, k, false, i, \rho]}{[S|w, k+1, false, i+1, \rho + \rho_{\text{sh}}]} \\
\textbf{reduce-L/R-X} & \dfrac{[S|s_1 s_0, k, false, i, \rho]}{[S|X, k, false, i+1, \rho + \rho_{\text{re}}]} \\
\textbf{unary-X} & \dfrac{[S|s_0, k, false, i, \rho]}{[S|X, k, false, i+1, \rho + \rho_{\text{un}}]} \\
\textbf{finish} & \dfrac{[S, |\mathbf{w}|, false, i, \rho]}{[S, |\mathbf{w}|, true, i+1, \rho + \rho_{\text{fi}}]} \\
\textbf{idle} & \dfrac{[S, |\mathbf{w}|, true, i, \rho]}{[S, |\mathbf{w}|, true, i+1, \rho + \rho_{\text{id}}]}
\end{array}$$

Fig. 1 Transition action set of the baseline parser [4].

trees. Figure 2 shows a binarized constituent tree and its non-binarized tree of the sentence "John has a dog.". Under the above action set, a transition-based parser can achieve the binarized parse tree in Fig. 2 through the sequence of actions "**shift unary-NP shift shift shift reduce-R-NP reduce-L-VP shift reduce-L-S∗ reduce-R-S finish**".

Since each state is achieved by successively applying actions $\mathbf{a} = \langle a_1, a_2, \cdots, a_{|\mathbf{a}|} \rangle$ over the initial state $[\emptyset, 0, false, 0, 0]$, we denote a state as its action sequence $\mathbf{a}$ leading to itself, omitting the initial state for simplicity reason throughout this paper. While parsing a sentence $\mathbf{w}$, the score $\rho(\mathbf{a})$ of each state is formally defined as the total score of all its prefix states $a_1^i = \langle a_1, a_2, \cdots, a_i \rangle$:

$$\rho(\mathbf{a}, \mathbf{w}; \theta) = \sum_{i=1}^{|\mathbf{a}|} \theta \cdot \Phi(a_1^i, \mathbf{w}), \tag{1}$$

where θ is a vector as the model parameter and Φ is the feature vector of the state.

In this paper, Φ is specified by the feature template presented in [4], which mainly includes n-gram POS tag, constituent label, and word features, thus fails to capture long-term syntactic structures, which are beneficial to predicting the constituent label for relatively long spans, as to be shown in our experiments.

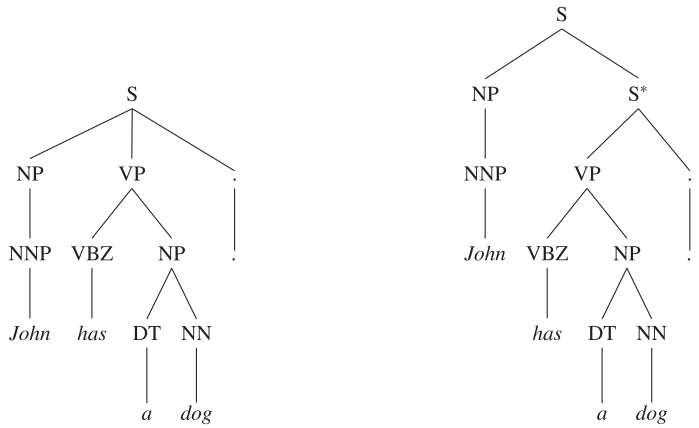
3. Sequence-to-Sequence RNN Parsing with Buttom-Up Linearization

As feature-rich models can not make full use of all actions in parse history when making a decision, we employ sequence-to-sequence RNN models, which are able to encode unbounded action history during decoding in a feature-rich transition-based model. In this section, we first overview a conventional sequence-to-sequence RNN model (Sect. 3.1), and then propose novel linearization methods to integrate the RNN model into decoding of transition-based models (Sect. 3.2).

3.1 Sequence-to-Sequence RNN Model

A sequence-to-sequence RNN model is a general framework for sequence-to-sequence transformation tasks, where both the input and output are sequences of tokens. The model is simple yet powerful, and thus has been widely used for many NLP tasks such as machine translation [12], [13], and image captioning [14]. In particular, [8] utilized this model for constituent parsing and demonstrated very competitive results.

Suppose that the input sequence is $\mathbf{w}$ and the output sequence is $\mathbf{o} = \langle o_1, \cdots, o_{|\mathbf{o}|} \rangle$. Formally, the sequence-to-sequence RNN model is the probability under the encoder-decoder framework and can be calculated as following:



(a) non-binarized (b) binarized

Fig. 2 Example of a constituent tree and its corresponding binarized tree. The "temporary nodes" (marked by a star symbol (*) in the binarized tree) should be removed when the binarized tree is recovered to the non-binarized tree.

Table 1 Comparison of different linearization methods for the constituent tree in Fig. 2. "TD" is the top-down linearization method of [8]. The black box in the last row means that the model have to wait for the decoder at that time step.

Method	Result			
TD	(S (NP NNP)$_{NP}$ (VP VBZ (NP DT NN)$_{NP}$)$_{VP}$.)$_{S}$			
"bo1"	SHIFT	UNARY-NP	SHIFT	SHIFT
	SHIFT	REDUCE-R-NP	REDUCE-L-VP	SHIFT
	REDUCE-L-S*	REDUCE-R-S	FINISH	
"bo2"	SHIFT	UNARY-NP-LEAF	SHIFT	SHIFT
	SHIFT	REDUCE-R-NP-{LEAF,LEAF}	REDUCE-L-VP-{LEAF,NP}	SHIFT
	REDUCE-L-S*-{VP,LEAF}	REDUCE-R-S-{NP,S*}	FINISH	
"nb"	SHIFT	UNARY-NP	SHIFT	SHIFT
	SHIFT	REDUCE-NP-2	REDUCE-VP-2	SHIFT
	■	REDUCE-S-3	FINISH	

$$P(\mathbf{o} \mid \mathbf{w}; \theta) = \prod_{i=1}^{|\mathbf{o}|} P(o_i \mid o_1^{i-1}, \mathbf{w}; \theta) = \prod_{i=1}^{|\mathbf{o}|} \mathbf{softmax}(\varphi(h_i))[o_i], \quad (2)$$

where θ is the model parameter; h_i denotes a hidden vector of $\mathbf{o}$ at timestep i; φ is the activation function of the hidden layer. **softmax** is the softmax function, and $[o_i]$ denotes the component in a vector corresponding to the index of o_i in its vocabulary. Furthermore, h_i is defined by a recurrent function over both the previous hidden vector h_{i-1} and the context $c(\mathbf{w}, o_1^{i-1})$, i.e., $h_i = f(h_{i-1}, c(\mathbf{w}, o_1^{i-1}))$. In this paper, we employ the GRU [15] to define h_i and an attention mechanism [13] to define $c(\mathbf{w}, o_1^{i-1})$.

In [8], the input sequence $\mathbf{w}$ is a sentence of words while the output sequence $\mathbf{o}$ is a sequence of tokens obtained via linearizing a parsing tree of $\mathbf{w}$ in a top-down manner by depth-first traversal order. Table 1 shows such a linearized sequence for the constituent tree in Fig. 2.

Since transition-based parsing models process the word sequence in a bottom-up style, the RNN model with the top-down linearization method cannot be incorporated into decoding of transition-based models[†]. Therefore, in the next subsection, we propose several linearization methods to convert a tree into a sequence of tokens in a bottom-up manner.

For simplicity, in the next subsection, each bottom-up linearization is defined on a complete tree. However, it is straightforward to extend their definitions on a set of subtrees in the stack S.

3.2 Bottom-Up Style Linearization

3.2.1 Binarized Order-1 Linearization ("bo1")

The transition-based parser represents a constituent tree as a transition action sequence, which is generated in a bottom-up manner, and this correspondence (i.e., a constituent tree and its action sequence) is one-to-one. Therefore, it is natural to linearize a tree using the transition action sequence of the transition-based parser. In other words, each token in the linearized sequence is exactly the same as an action in the action sequence. For example, the tree in Fig. 2 (a) is firstly

[†]The RNN model with top-down linearization can be used for reranking, but as shown by our experiments, reranking using RNN models does not work (see Sect. 5.2 for details).

binarized into the tree in Fig. 2 (b) by our baseline transition-based parser based on the binarization rules of [1], which is then linearized as a sequence of transition actions in Fig. 1, generating the token sequence in the second row of Table 1.

3.2.2 Binarized Order-2 Linearization ("bo2")

Inspired by [16], we propose a binarized order-2 linearization method ("bo2"), which can utilize the information of high-order features, which have been proven to be useful for parsing. Concretely, for the "**unary-X**" or "**reduce-L/R-X**" action, its corresponding token is augmented with the labels for its child nodes. Note that for all nodes representing the POS-tag (i.e., the pre-leaf node), we assign a special symbol "*LEAF*" to their labels regardless of its actual label. The other actions are the same as "bo1". By using this "bo2" method, the tree in Fig. 2 (b) is linearized to the token sequence in the third row of Table 1.

The "bo2" linearization method is fine-grained, utilizing more information for decision compared with "bo1". However, because the label information of the child nodes is involved in "bo2", the token vocabulary size of the linearized sequences is much larger than that of "bo1", which might make the prediction of each token more difficult.

3.2.3 Non-Binarized Linearization ("nb")

The above two linearization methods operate on binarized trees, and the resulting linearized sequence is longer than the number of nodes in its non-binarized tree, due to the import of "temporary nodes". Since it is well known that prediction on long sequences is challenging for RNN models [17], [18], we propose another linearization model directly operating on non-binarized trees, making the token sequence shorter. Instead of discriminating "**unary**" and "**reduce**", in the "nb" method, we directly record the number of subtrees involved in an "**reduce**" action. This method generates the linearized token sequence in the last row of Table 1.

The "nb" linearization shortens the length of the sequence. Since the constituent tree generated by the baseline transition-based parser is a binarized tree, while the "nb" linearization method does not need binarization, the "nb" model has to delay the decision of the next action, which is represented as the black square in Table 1 (see Sect. 4.1 for details).

4. Improved Feature-Rich Transition-Based Parsing with RNN Models

4.1 Decoding

In order to cope with the limitation on long-term syntactic structures for a feature-rich transition-based model, we integrate multiple sequence-to-sequence models encoding the whole action history into the feature-rich transition-based model as global features. Formally, we augment the feature-rich model with RNN models using the linear combination:

$$\hat{\rho}(\mathbf{a}, \mathbf{w}; \hat{\theta}) = \rho(\mathbf{a}, \mathbf{w}; \theta) + \sum_{j=1}^{J} \lambda^j \cdot P(\mathbf{o}^j(\mathbf{a}) \mid \mathbf{w}; \theta^j), \quad (3)$$

where $\hat{\theta} = \{\theta, \theta^1, \cdots, \theta^J, \lambda^1, \cdots, \lambda^J\}$ denotes the parameter of our proposed model $\hat{\rho}$, and $\rho(\mathbf{a}, \mathbf{w}; \theta)$ is the feature-rich model as defined in Eq. (1). J is the number of different RNN models with respect to different linearization methods $\mathbf{o}^j(\mathbf{a})$ presented in Sect. 3, and $\mathbf{o}(\mathbf{a})$ denotes the linearized sequence of the subtrees obtained by an action sequence $\mathbf{a}$. $P(\mathbf{o}^j(\mathbf{a}) \mid \mathbf{w}; \theta^j)$ is an RNN model as defined in Eq. (2).

The parsing procedure with RNN models is similar to the baseline transition-based parser, except the incremental calculation of RNN models based on different linearization methods. Suppose that the current state is available by applying an action sequence a_1^{i-1}, and the next valid action is a_i. For the RNN model with "bo1" linearization, it is trivial to calculate the RNN-based score of a_1^i, i.e., $P(o_i \mid o_1^{i-1}; \theta)$ in Eq. (2). For the RNN model with "bo2" linearization, it is similar to the case of "bo1" if a_i is "**shift**" or "**finish**"; while if a_i is "**unary**" or "**reduce-L/R**", we have to lookup the child labels for the first subtree in the stack in order to get the corresponding token of a_i. For the RNN model with "nb" linearization, it is also similar to the case of "bo1" if a_i is "**shift**", "**finish**" or "**unary**", but it is non-trivial if a_i is "**reduce-L/R**". For example, in Fig. 2 (b), if a_i generates a non-temporary node "S", in order to obtain the number of its child nodes, we have to collapse this binarized subtree into non-binarized one, getting the token **reduce-S-3** for a_i. However, if a_i generates a temporary node "S*", there is no corresponding token in "nb" linearization method and thus we have to wait until a non-temporary node like "S" been generated. In all linearization methods, the "**idle**" action is ignored and the RNN models just set the score of the "idle" action as 0.

4.2 Training

Ideally, we can jointly train each element of the parameter vector $\hat{\theta}$ using an optimization algorithm such as the widely-used structured perceptron algorithm in traditional feature-rich constituent parsing [4]. Unfortunately, this joint training involves repeatedly decoding and this leads to exhaustive calculation of $P(\mathbf{o}^j(\mathbf{a}) \mid \mathbf{w}; \theta^j)$, which is very time-consuming in practice.

To improve the training efficiency, we train our proposed model in a pipeline manner instead of training jointly. In our training process, the parameter θ for the transition model ρ and parameters θ^j of each RNN model are firstly separately optimized. For θ, we use the standard structured perceptron. For θ^j, we employ the maximum log-likelihood algorithm, which does not require decoding. Then, we fix the optimized θ and θ^j (for all j) and then learn λ^j only.

In our experiments, for the first stage, we use the training data for optimization; while for the second stage we use the development set for tuning λ^j, because there are only a few parameters. In particular, for the second stage, we

employ a simple grid search procedure [19], which learns one parameter λ^j while fixing other $\lambda^{j'}$ ($j' \neq j$) each time[†]. Even though our seperate training may lead to suboptimal solution in theory, our experiments empirically show that it works well in practice.

5. Experiments

5.1 Settings

To demonstrate the effectiveness of our framework, we experimentally investigated the parsing performance for two languages: English and German (a morphologically rich language). Following previous works such as [7], for English data, we used the Wall Street Journal (WSJ) part of the Penn Treebank [21]. Sections 2-21, 22, and 23 were used for training, development, and testing, respectively. For German, we used the data from SPMRL 2014 shared task [22]. The SPMRL datasets are already divided into training, development, and testing sets. For both languages, the sentences were POS tagged using the Stanford NLP tools[††]. For German, we also give the experiment results for the case of golden POS-tags.

We use Zpar[†††] as the implementation of the feature-rich parser and the RNN model from GroundHog[††††] as the implementation of the sequence-to-sequence RNN parsing models. Following the default configuration of Zpar, the beam size is set to 16. For GroundHog, the dimension of word embedding is 620, the dimension of hidden units is 1000, and the batch size is 32. We used the most frequent 35K words and 50K words in the training data for English and German, respectively. The input vocabulary size is set manually, which is a trade-off between the word coverage on the corpus and the decoding speed.

5.2 Comparison of Models for Different Linearization

Our proposed models with different linearization methods were compared against the baseline parser [4][†††††]. Table 2 summarizes the parsing performance of each model on each test set. The best results are highlighted in bold font. The significance test was performed by the McNemar's test. The χ^2 test statistics were also reported in Table 2. The column "SI" shows whether the methods significantly outperform the corresponding baseline. Table 2 shows that, for both languages and all three models, the sequence-to-sequence RNN models improved the performance of the baseline parser.

Among the proposed three linearization methods, the "bo1" linearization method achieved the best result. As described in Sects. 3.2.2 and 3.2.3, we proposed the "nb" model and the "bo2" model to address two problems in the "bo1" model, respectively: (1) the sequence might be too long due to binarization, (2) the grandchildren information was not utilized. However, these results imply that new problems occurred in these two new models: the decision-delay problem (in the "nb" model) and the sparsity problem (in the "bo2" model). For the "nb" model, although the sequence is shorter since no binarization is needed, there exists some mismatch between the action sequence of the "nb" model and the action sequence implemented by the baseline parser, which makes the calculation of the score given by the "nb" model be delayed until all the child nodes of a node been generated. According to our statistics, for all 2,416 sentences of Section 23 of WSJ (the test set), if we convert the golden trees to the action sequences that should be implemented by the baseline parser, the total number of actions is 117,321. If we convert the golden trees to the action sequences of the "nb" model, the total number of actions is 100,866, which is only about 14.0% smaller. However, among these actions, 11,557 actions (about 11.5%) are mismatched. The poor result of the "nb" model could be caused by this reason. For the "bo2" model, although the grandchildren information is utilized, the vocabulary size of the action sequences is enlarged, causing the problem of sparsity. For Section 23 of WSJ, the vocabulary size increased from 92 ("bo1") to 1335 ("bo2"), about 15 times enlarged. This could be the reason that the performance of the "bo2" model is worse than that of the "bo1" model.

Our method did not perform very well on German. The improvement is less than the improvement on English. This is caused by the difference of the existence of OOV in these two languages. Compared with German, a typical fusional language, English has a more analytic structure, which makes the number of different words in German corpus much larger than that of English corpus (both training corpus and test corpus), causing the number of OOV in German test data larger. In our experiments, although we make the vocabulary size of German (50k) larger than that of English (35k), the percentage of OOV (6.2%) of German is still larger than that of English (1.7%). This limits the potential of RNNs.

Furthermore, to show the effectiveness of the proposed integration of the RNN models, i.e., integration into decoding, we compared to the parser where RNN models are used for reranking. Concretely, we use each RNN model to score the n-best list ($n = 16$ in our experiments) generated by the baseline parser, and generate the parsing result with the highest score, which is calculated by adding the score given

[†]We did try the sophisticated algorithm MERT [20] to optimize λ^j, which needs much more rounds of decoding, but we did not observe significant improvements over the simple grid search in our preliminary experiments.

[††]http://nlp.stanford.edu/software/tagger.html

[†††]https://github.com/frcchang/zpar/releases

[††††]https://github.com/lisa-groundhog/GroundHog

[†††††]If we only use the pure RNN model trained by GroundHog to parse using the bottom-up linearization (e.g., bo1), the F-score is only 84.03 on the development set of WSJ (with one bo1 model). The F-score might seem to be low when compared with the F-score 90.5 reported in [8]. However, as we will demonstrate in our experiments, even with this poor model, the performance of the baseline parser can still be improved. If better RNN models (such as the models in [7]) are used, we can expect that our framework can get higher F-score than what is reported in this paper, which implies the potential of our framework.

Table 2 Comparison between the baseline parser and the proposed parsers using one RNN model. "LR", "LP", and "F1" denote the recall, precision, and F-score of parsing, respectively. "gold" and "auto" represent the case of using gold POS-tags and automatically labelled POS-tags, respectively. "SI" means whether the improvement is "statistical significant improvement".

Data	Model	LR	LP	F1	χ^2	SI
English	baseline	90.20	90.70	90.40	-	-
	bo1	**91.56**	**91.05**	**91.30**	25.30	Yes ($p < 0.001$)
	nb	91.25	90.84	91.05	20.27	Yes ($p < 0.001$)
	bo2	91.41	91.05	91.23	18.52	Yes ($p < 0.001$)
German (gold)	baseline	86.11	82.59	84.31	-	-
	bo1	**87.08**	**82.76**	**84.87**	5.27	Yes ($p < 0.025$)
	nb	86.86	82.67	84.71	1.46	No
	bo2	86.78	82.64	84.66	0.35	No
German (auto)	baseline	82.24	83.07	82.65	-	-
	bo1	**82.54**	83.12	**82.83**	3.92	Yes ($p < 0.05$)
	nb	82.51	**83.13**	82.82	1.21	No
	bo2	82.48	83.10	82.79	1.06	No

Table 3 Examples of parsing results for short and long tree-span length. Here, we only show the labels of the root nodes of the subtrees. The low-frequency words are highlighted in bold.

Tree-span	Zpar	RNN parser	Our parser	Gold
very **satisfactorily**	ADVP	ADJP	ADVP	ADVP
See related story: "And **Bills** to Make Wishes Come True" – WSJ Oct. 17, 1989.	NP	S	S	S

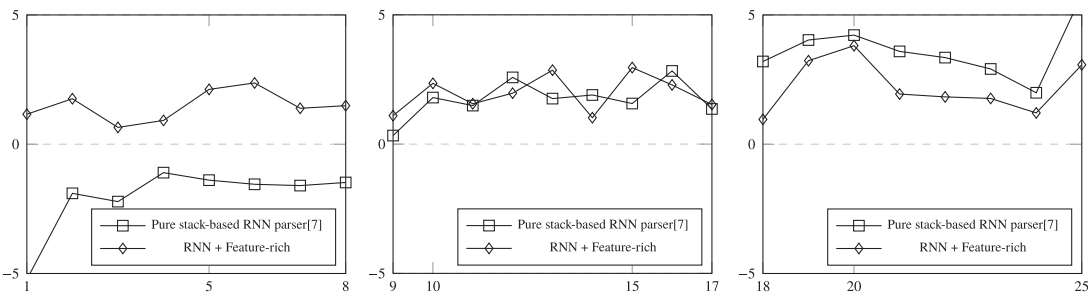


Fig. 3 Difference of F-score over baseline (y−axis) w.r.t. tree-span length (x−axis).

Table 4 Results of reranking. "TD" is the RNN model using the top-down linearization method of [8].

Data	Model	LR	LP	F1
English	baseline	90.20	90.70	90.40
	rerank (TD)	90.25	90.73	90.49
	rerank (bo1)	90.30	90.75	90.52
German	baseline	82.24	83.07	82.65
	rerank (TD)	82.27	83.00	82.63
	rerank (bo1)	82.30	83.10	82.70

by the RNN model. Table 4 shows the results[†]. We can see that the reranking methods can only improve the parsing performance slightly, while our method outperforms the reranking method.

Additionally, we evaluated the performance of combining different models by ensemble. The experiment results are listed in Table 5, where the numbers in the parentheses denote the numbers of combined models. For example, the "bo1(5)" model combines five "bo1" models with different initializations. Again, we reported the result of McNemar's test. The baseline is the same as the baseline in Table 2.

Comparing the English results in Table 2 with the first three rows in Table 5 (e.g. row 2 in Table 2 vs. row 1 in Table 5), we observe that combining several models with the same linearization methods improves the performance of all linearization methods. In addition, the fourth row in Table 5 shows that averaging the models using different linearization methods also improves the performance. On German dataset, similar phenomenon can be observed. Furthermore, by combining models with different linearization methods, the performance can be improved further. The final results (91.56 F1 for English and 84.97 F1 for German) outperform the best results in Table 2 (91.30 F1 for English and 84.87 for German).

5.3 Analysis on Different Length of Tree-Span

We examined the performance on different sized constituent trees. For a subtree with the root node X and leaf nodes $\{w_1 \dots w_n\}$, we define $\{w_1 \dots w_n\}$ as the tree-span of X, and regard the length of this tree-span, i.e., n, as the size of the tree.

[†]For simplicity, we report only rerank (TD), the RNN-model with the top-down linearization method, and rerank (bo1), which achieves best performance among the proposed bottom-up linearization methods.

Table 5 Results of the proposed parsers using several RNN models.

Data	Models	LR	LP	F1	χ^2	SI
English	bo1(5)	**91.80**	**91.32**	**91.56**	78.93	Yes ($p < 0.001$)
	nb(5)	91.57	91.03	91.30	32.11	Yes ($p < 0.001$)
	bo2(5)	91.71	91.30	91.51	80.93	Yes ($p < 0.001$)
	bo1(1)+nb(1)+bo2(1)	91.68	91.26	91.47	70.06	Yes ($p < 0.001$)
	bo1(5)+nb(5)	91.79	91.25	91.52	60.63	Yes ($p < 0.001$)
	bo1(5)+bo2(5)	91.72	91.18	91.45	54.48	Yes ($p < 0.001$)
German (gold)	bo1(5)	87.04	82.68	84.81	1.85	No
	nb(5)	86.80	82.69	84.70	2.52	No
	bo2(5)	86.97	**82.76**	84.81	5.53	Yes ($p < 0.025$)
	bo1(1)+nb(1)+bo2(1)	87.11	82.70	84.85	2.73	Yes ($p < 0.1$)
	bo1(5)+nb(5)	87.13	82.62	84.81	0.21	No
	bo1(5)+bo2(5)	**87.34**	82.72	**84.97**	5.49	Yes ($p < 0.025$)
German (auto)	bo1(5)	82.55	83.14	82.84	1.92	No
	nb(5)	82.50	83.10	82.80	1.31	No
	bo2(5)	82.50	83.12	82.81	2.71	Yes ($p < 0.1$)
	bo1(1)+nb(1)+bo2(1)	**82.82**	83.21	83.01	2.01	No
	bo1(5)+nb(5)	82.14	83.31	82.72	1.52	No
	bo1(5)+bo2(5)	81.66	**84.51**	**83.06**	6.72	Yes ($p < 0.01$)

Table 6 Comparison with other parsers. TB: using treebank only, AD: using additional unlabelled data, star symbol (*): parsers based on neural networks. "Gold": using golden POS-tags.

English (WSJ)				German (SPMRL 2014)	
TB		AD		TB & AD	
[2]	90.1	[23]	90.4	[24]	77.15
[4]	90.4	[25]	90.7	[26]	78.43
[8]*	90.5	[27]*	91.1	[28]	81.66
[7]*	90.7	[4] + Semi	91.3	[27]*	80.95
[29] (rerank)	91.0	[8]* + Semi	92.1	[30]*	82.00
[3]	91.1	[31]	92.3	Baseline [4]	82.65
[32]*	91.3	[11]*	**93.8**	This work*	**83.06**
This work*	**91.5**			[30]* (Gold)	84.60
[9]* (rerank)	92.4			This work* (Gold)	**84.97**
[11]* (rerank)	**92.6**				

Figure 3 shows how parsing performance changes with respect to the length of tree-span. We evaluate the F-score of the baseline parser, the RNN-based parser, and our parser ("bo1(5)") on the subtrees with different tree-span length on the test set of WSJ. Here, the baseline parser is Zpar (a state-of-the-art feature-rich parser), the RNN-based parser is the parser proposed by [7] (a state-of-the-art RNN-based parser). For the RNN-based parser, we did not use our direct baseline, i.e., the pure RNN model without rich features, because the pure RNN model performed quite bad for parsing[†]. We did not use the parser proposed by [8], either, because we failed to reproduce their outstanding result, which may caused by something unreported in their paper about the experiment configuration. However, we succeed to reproduce the result reported by [7], another outstanding RNN-based parser, so we chose this for comparison. Note that we evaluated only subtrees whose leaf nodes contain low-frequency words (specifically, frequency less than 5 in the training corpus) in order to demonstrate the influence of them. Figure 3 is split to three sub-figures based on tree-span length (short, middle, long). To make the illustrations clear, we plot the difference of the F-score from the baseline parser, rather than plotting the F-scores of each parser directly. In addition, Table 3 gives the examples of the parsing results of each parser for short and long tree-span length.

As can be seen in Fig. 3 and Table 3, the RNN-based parser cannot parse short tree-spans accurately. This indicates that, for short tree spans containing low-frequency words, the feature-rich parser can parse them with the help of simple back-off features, while RNN-based parsers cannot get enough data to learn. On the other hand, for middle-length and long-length tree spans, the RNN-based parser outperforms the baseline parser. This indicates that the RNN-based parser can utilize enough history information to make correct decision while the feature-rich parser cannot use a feature template to capture this global information due to the problem of sparseness. Figure 3 and Table 3 also shows that our parser performs well on both short and long tree-spans. This indicates that our parser combines the advantages of the RNN-based parser and the feature-rich parser, which coincides our motivation.

5.4 Comparison with State-of-the-Art Constituent Parsers

Table 6 compares the parsing performance (F1) of our

[†]Although the performance was bad, the experiments showed that when using the pure RNN model without rich features, the tendency was similar to Fig. 3, i.e., performing worse for shorter tree-spans, and performing better for longer tree-spans.

framework and other state-of-the-art parsers on the test set of the German SPMRL dataset and on Section 23 (test set) of the English WSJ treebank.

On the English dataset, our parser is competitive to the state-of-the-art parsers trained on WSJ data only. As reported in [9], reranking with the generative RNN models deliver gains over end-to-end discriminative RNN models, our framework thereby has the potential to advance the reranking models [9], [11] by integrating these generative RNN models into decoding[†], which remains a future work. On the German dataset, particularly, our parser outperforms other state-of-the-art parsers both when using auto-labelled POS tags and gold tags.

6. Related Works

There have been several notable works on RNN-based constituent parsing. For example, [7] proposed an end-to-end RNN parser whose performance is comparable to the state-of-the-art feature-rich parsers, and [9] proposed a variant RNN model for parsing. Unlike their pure neural network models, ours is a hybrid model consisting of neural networks and feature-rich models, which can combine their merits. Furthermore, our framework is general and can be applied on top of their RNN models.

There have been some other efforts about using NNs to improve the feature-rich parsers. [25] used feedforward neural networks (FFNNs) to automatically learn the optimized features for parsing, but their work cannot capture the information of the whole parsing history. [27] integrated FFNNs into CRF-based parsing, but their parser relied on additional datasets in order to match the state-of-the-art. Compared with these works, our parser uses RNNs, which can encode the information of the whole parsing history, and can yield quite good results without using any additional datasets.

7. Conclusions

We proposed a novel framework that improves the performance of the feature-rich transition-based constituent parser by using RNN models. The proposed framework combines the merits of feature-rich parsers and RNNs: it can make reliable parsing decisions for both short tree-spans by informative features and long tree-spans by encoding the parsing history with RNN models. Consequently, our parser achieves competitive performance on the standard English WSJ task without using any additional data. Furthermore, our parser set a new state-of-the-art result on German, a morphologically-rich language.

References

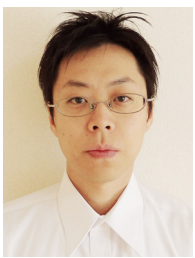
[1] M. Collins, "Head-driven statistical models for natural language parsing," Ph.D. thesis, University of Pennsylvania, 1999.

[†]In this sense, the RNN language model in [11] is considered as a generative model [33].

[2] S. Petrov and D. Klein, "Improved inference for unlexicalized parsing," HLT-NAACL, 1, vol.7, pp.404–411, 2007.
[3] X. Carreras, M. Collins, and T. Koo, "TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing," Proc. Twelfth Conference on Computational Natural Language Learning, pp.9–16, Association for Computational Linguistics, 2008.
[4] M. Zhu, Y. Zhang, W. Chen, M. Zhang, and J. Zhu, "Fast and accurate shift-reduce constituent parsing," Proc. 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Sofia, Bulgaria, pp.434–443, Aug. 2013.
[5] Z. Wang and N. Xue, "Joint POS tagging and transition-based constituent parsing in Chinese with non-local features," Proc. 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Baltimore, Maryland, pp.733–742, Association for Computational Linguistics, June 2014.
[6] J. Henderson, "Inducing history representations for broad coverage statistical parsing," Proc. 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1, pp.24–31, Association for Computational Linguistics, 2003.
[7] T. Watanabe and E. Sumita, "Transition-based neural constituent parsing," Proc. 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Beijing, China, pp.1169–1179, Association for Computational Linguistics, July 2015.
[8] O. Vinyals, Ł. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton, "Grammar as a foreign language," Advances in Neural Information Processing Systems, vol.28, pp.2773–2781, 2015.
[9] C. Dyer, A. Kuncoro, M. Ballesteros, and N.A. Smith, "Recurrent neural network grammars," Proc. 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, California, pp.199–209, Association for Computational Linguistics, June 2016.
[10] Y. Kim, Y. Jernite, D. Sontag, and A.M. Rush, "Character-aware neural language models," arXiv preprint arXiv:1508.06615, 2015.
[11] D.K. Choe and E. Charniak, "Parsing as language modeling," Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), pp.2331–2336, 2016.
[12] I. Sutskever, O. Vinyals, and Q.V. Le, "Sequence to sequence learning with neural networks," Advances in Neural Information Processing systems, vol.27, pp.3104–3112, 2014.
[13] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," arXiv preprint arXiv:1409.0473, 2014.
[14] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R.S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," arXiv preprint arXiv:1502.03044, vol.2, no.3, p.5, 2015.
[15] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," Proc. 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, pp.1724–1734, Association for Computational Linguistics, Oct. 2014.
[16] T. Koo and M. Collins, "Efficient third-order dependency parsers," Proc. 48th Annual Meeting of the Association for Computational Linguistics, pp.1–11, 2010.
[17] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," IEEE Trans. Neural Netw., vol.5, no.2, pp.157–166, 1994.
[18] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks.," ICML '13, Proc. 30th International Conference on Machine Learning, vol.28, no.3, pp.1310–1318, 2013.
[19] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," Journal of Machine Learning Research, vol.13, pp.281–

305, 2012.
[20] F.J. Och, "Minimum error rate training in statistical machine translation," Proc. 41st Annual Meeting on Association for Computational Linguistics-Volume 1, pp.160–167, Association for Computational Linguistics, 2003.
[21] M.P. Marcus, M.A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of English: The Penn treebank," Computational Linguistics, vol.19, no.2, pp.313–330, 1993.
[22] D. Seddah, S. Kübler, and R. Tsarfaty, "Introducing the SPMRL 2014 shared task on parsing morphologically-rich languages," Proc. First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages, pp.103–109, 2014.
[23] M. Zhu, J. Zhu, and H. Wang, "Exploiting lexical dependencies from large-scale data for better shift-reduce constituency parsing," Proc. COLING 2012, pp.3171–3186, 2012.
[24] B. Crabbé and D. Seddah, "Multilingual discriminative shift-reduce phrase structure parsing for the SPMRL 2014 shared task," Proc. First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages, 2014.
[25] Z. Wang, H. Mi, and N. Xue, "Feature optimization for constituent parsing via neural networks," Proc. 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Beijing, China, pp.1138–1147, Association for Computational Linguistics, July 2015.
[26] D. Hall, G. Durrett, and D. Klein, "Less grammar, more features," Proc. 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp.228–237, 2014.
[27] G. Durrett and D. Klein, "Neural CRF parsing," Proc. 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Beijing, China, pp.302–312, Association for Computational Linguistics, July 2015.
[28] A. Björkelund, Ö. Çetinoğlu, R. Farkas, T. Mueller, and W. Seeker, "(re) ranking meets morphosyntax: State-of-the-art results from the SPMRL 2013 shared task," Proc. Fourth Workshop on Statistical Parsing of Morphologically Rich Languages, pp.135–145, 2013.
[29] E. Charniak and M. Johnson, "Coarse-to-fine *n*-best parsing and MaxEnt discriminative reranking," Proc. 43rd Annual Meeting on Association for Computational Linguistics, pp.173–180, Association for Computational Linguistics, 2005.
[30] J. Legrand and R. Collobert, "Deep neural networks for syntactic parsing of morphologically rich languages," Proc. 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp.573–578, 2016.
[31] D. McClosky, E. Charniak, and M. Johnson, "Effective self-training for parsing," Proc. Main Conference on Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, pp.152–159, Association for Computational Linguistics, 2006.
[32] J. Cross and L. Huang, "Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles," Proc. 2016 Conference on Empirical Methods in Natural Language Processing, Austin, Texas, pp.1–11, Association for Computational Linguistics, November 2016.
[33] K.S. Jones, "Language modelling's generative model: Is it rational," master thesis, Computer Laboratory, University of Cambridge, 2004.

Chunpeng Ma is currently a Dr. student of Harbin Institute of Technology. His advisor is Prof. Tiejun Zhao. He received his B.E., M.E. from Harbin Institute of Technology in 2010 and 2014, respectively. From July 2013 to January 2014, he interned at Baidu Inc. with instruction from Xianchao Wu. From October 2015 to September 2016, he interned at National Institute of Information and Communications Technology, with the instruction from Akihiro Tamura, cooperating with Lemao Liu. His research intests include syntactic parsing and machine learning.



Akihiro Tamura received his B.E., M.E., and Dr. Eng. from the Tokyo Institute of Technology in 2005, 2007, and 2013, respectively. After serving as a researcher at NEC and NICT, he is currently an assistant professor at Ehime University. His research interests include natural language processing and machine learning. He is a member of the Association for Computational Linguistics, the Information Processing Society of Japan, the Japanese Society for Artificial Intelligence, and the Association for Natural Language Processing.

Lemao Liu received his Ph.D. degree from Harbin Institute of Technology, Harbin, China, in 2013. Then he started his career as a postdoc at the City University of New York in November 2013, with Prof. Liang Huang. During the research for this paper, he was a research in the National Institute of Information and Communication Technology, Japan. His research interests include natural language processing, machine translation and parsing.

Tiejun Zhao is a professor of Research Center of Language Technology, School of Computer Science and Technology, Harbin Institute of Technology. He is associate dean of SCST and director of Ministry of Education-Microsoft Key Laboratory of NLP & Speech in HIT. He is associate director of Machine Translation Subject Committee of Chinese Information Processing Society, the member of editorial board of Journal of Chinese Information Processing, Journal of Automation, the senior member and associate director of Chinese Computing Committee of China Computer Federation. His research fields include: natural language understanding, content-based web information processing, applied artificial intelligence.

Eiichiro Sumita received the M.S. degree in computer science from the University of Electro-Communications in 1982 and the Ph.D. degree in engineering from Kyoto University in 1999. He is the Associate Director General, ASTREC (Advanced Speech Translation Research and Development Promotion Center), NICT. Before joining NICT, he worked for ATR and IBM. His research interests include machine translation and e-Learning.