An Empirical Study of Building a Strong Baseline for Constituency Parsing

Jun Suzuki, Sho Takase, Hidetaka Kamigaito, Makoto Morishita, and **Masaaki Nagata**
NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237 Japan
{suzuki.jun, takase.sho, kamigaito.hidetaka,
morishita.makoto, nagata.masaaki}@lab.ntt.co.jp

Abstract

This paper investigates the construction of a strong baseline based on general purpose sequence-to-sequence models for constituency parsing. We incorporate several techniques that were mainly developed in natural language generation tasks, e.g., machine translation and summarization, and demonstrate that the sequence-to-sequence model achieves the current top-notch parsers' performance without requiring explicit task-specific knowledge or architecture of constituent parsing.

1 Introduction

Sequence-to-sequence (Seq2seq) models have successfully improved many well-studied NLP tasks, especially for natural language generation (NLG) tasks, such as machine translation (MT) (Sutskever et al., 2014; Cho et al., 2014) and abstractive summarization (Rush et al., 2015). Seq2seq models have also been applied to constituency parsing (Vinyals et al., 2015) and provided a fairly good result. However one obvious, intuitive drawback of Seq2seq models when they are applied to constituency parsing is that they have no explicit architecture to model latent nested relationships among the words and phrases in constituency parse trees, Thus, models that directly model them, such as RNNG (Dyer et al., 2016), are an intuitively more promising approach. In fact, RNNG and its extensions (Kuncoro et al., 2017; Fried et al., 2017) provide the current state-of-the-art performance. Sec2seq models are currently considered a simple baseline of neural-based constituency parsing.

After the first proposal of an Seq2seq constituency parser, many task-independent techniques have been developed, mainly in the NLG research area. Our aim is to update the Seq2seq approach proposed in Vinyals et al. (2015) as a stronger baseline of constituency parsing. Our motivation is basically identical to that described in Denkowski and Neubig (2017). A strong baseline is crucial for reporting reliable experimental results. It offers a fair evaluation of promising new techniques if they solve new issues or simply resolve issues that have already been addressed by current generic technology. More specifically, it might become possible to analyze what types of implicit linguistic structures are easier or harder to capture for neural models by comparing the outputs of strong Seq2seq models and task-specific models, e.g., RNNG.

The contributions of this paper are summarized as follows: (1) a strong baseline for constituency parsing based on general purpose Seq2seq models[1], (2) an empirical investigation of several generic techniques that can (or cannot) contribute to improve the parser performance, (3) empirical evidence that Seq2seq models implicitly learn parse tree structures well without knowing task-specific and explicit tree structure information.

2 Constituency Parsing by Seq2seq

Our starting point is an RNN-based Seq2seq model with an attention mechanism that was applied to constituency parsing (Vinyals et al., 2015). We omit detailed descriptions due to space limitations, but note that our model architecture is identical to the one introduced in Luong et al. (2015a)[2].

A key trick for applying Seq2seq models to constituency parsing is the *linearization* of parse

[1] Our code and experimental configurations for reproducing our experiments are publicly available: https://github.com/nttcslab-nlp/strong_s2s_baseline_parser

[2] More specifically, our Seq2seq model follows the one implemented in seq2seq-attn (https://github.com/harvardnlp/seq2seq-attn), which is the alpha-version of the OpenNMT tool (http://opennmt.net).

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 1–7
Melbourne, Australia, July 15 - 20, 2018.

Original input	John has a dog .
Output: S-exp.	(S (NP NNP) (VP VBZ (NP DT NN)) .)
Linearized form	$(_{\mathrm{S}}\ (_{\mathrm{NP}}\ \mathrm{NNP}\ )_{\mathrm{NP}}\ (_{\mathrm{VP}}\ \mathrm{VBZ}\ (_{\mathrm{NP}}\ \mathrm{DT}\ \mathrm{NN}\ )_{\mathrm{NP}}\ )_{\mathrm{VP}}\ .\ )_{\mathrm{S}}$
w/ POS normalized	$(_{\mathrm{S}}\ (_{\mathrm{NP}}\ \mathrm{XX}\ )_{\mathrm{NP}}\ (_{\mathrm{VP}}\ \mathrm{XX}\ (_{\mathrm{NP}}\ \mathrm{XX}\ \mathrm{XX}\ )_{\mathrm{NP}}\ )_{\mathrm{VP}}\ .\ )_{\mathrm{S}}$

Table 1: Examples of linearization and POS-tag normalization (Vinyals et al., 2015)

trees (Vinyals et al., 2015). Roughly speaking, a *linearized parse tree* consists of open, close bracketing and POS-tags that correspond to a given input raw sentence. Since a one-to-one mapping exists between a parse tree and its linearized form (if the linearized form is a valid tree), we can recover parse trees from the predicted linearized parse tree. Vinyals et al. (2015) also introduced the *part-of-speech (POS) tag normalization* technique. They substituted each POS tag in a linearized parse tree to a single XX-tag[3], which allows Seq2seq models to achieve a more competitive performance range than the current state-of-the-art parses[4]. Table 1 shows an example of a parse tree to which linearization and POS-tag normalization was applied.

3 Task-independent Extensions

This section describes several generic techniques that improve Seq2seq performance[5]. Table 2 lists the notations used in this paper for a convenient reference.

3.1 Subword as input features

Applying subword decomposition has recently become a leading technique in NMT literature (Sennrich et al., 2016; Wu et al., 2016). Its primary advantage is a significant reduction of the serious out-of-vocabulary (OOV) problem. We incorporated subword information as an additional feature of the original input words. A similar usage of subword features was previously proposed in Bojanowski et al. (2017).

Formally, the encoder embedding vector at encoder position i, namely, $\boldsymbol{e}_i$, is calculated as follows:

$$\boldsymbol{e}_i = \boldsymbol{E}\boldsymbol{x}_k + \sum_{k' \in \psi(w_i)} \boldsymbol{F}\boldsymbol{s}_{k'}, \qquad (1)$$

[3]We did not substitute POS-tags for punctuation symbols such as ".", and ",".

[4]Several recently developed neural-based constituency parsers ignore POS tags since they are not evaluated in the standard evaluation metric of constituency parsing (Bracketing F-measure).

[5]Figure in the supplementary material shows the brief sketch of the method explained in the following section.

D	: dimension of the embeddings
H	: dimension of the hidden states
i	: index of the (token) position in input sentence
j	: index of the (token) position in output linearized format of parse tree
$\mathcal{V}^{(e)}$	: vocabulary of word for input (encoder) side
$\mathcal{V}^{(s)}$	: vocabulary of subword for input (encoder) side
$\boldsymbol{E}$	: encoder embedding matrix for $\mathcal{V}^{(e)}$, where $\boldsymbol{E} \in \mathbb{R}^{D \times \lvert\mathcal{V}^{(e)}\rvert}$
$\boldsymbol{F}$	: encoder embedding matrix for $\mathcal{V}^{(s)}$, where $\boldsymbol{F} \in \mathbb{R}^{D \times \lvert\mathcal{V}^{(s)}\rvert}$
w_i	: i-th word (token) in the input sentence, $w_i \in \mathcal{V}^{(e)}$
$\boldsymbol{x}_k$	: one-hot vector representation of the k-th word in $\mathcal{V}^{(e)}$
$\boldsymbol{s}_k$	: one-hot vector representation of the k-th subword in $\mathcal{V}^{(s)}$
$\boldsymbol{u}$	: encoder embedding vector of unknown token
$\phi(\cdot)$	: function that returns the index of given word in the vocabulary $\mathcal{V}^{(e)}$
$\psi(\cdot)$	: function that returns a set of indices in the subword vocabulary $\mathcal{V}^{(s)}$ generated from the given word. e.g., $k \in \psi(w_i)$
$\boldsymbol{e}_i$	: encoder embedding vector at position i in encoder
$\mathcal{V}^{(d)}$	: vocabulary of output with POS-tag normalization
$\mathcal{V}^{(q)}$	: vocabulary of output without POS-tag normalization
$\boldsymbol{W}^{(o)}$	: decoder output matrix for $\mathcal{V}^{(d)}$, where $\boldsymbol{W}^{(o)} \in \mathbb{R}^{\lvert\mathcal{V}^{(o)}\rvert \times H}$
$\boldsymbol{W}^{(q)}$	: decoder output matrix for $\mathcal{V}^{(q)}$, where $\boldsymbol{W}^{(q)} \in \mathbb{R}^{\lvert\mathcal{V}^{(q)}\rvert \times H}$
$\boldsymbol{z}_j$	: final hidden vector calculated at the decoder position j
$\boldsymbol{o}_j$	: final decoder output scores at decoder position j
$\boldsymbol{q}_j$	: output scores of auxiliary task at decoder position j
$\boldsymbol{b}$	: additional bias term in the decoder output layer for mask
$\boldsymbol{p}_j$	: vector format of output probability at decoder position j
A	: number of models for ensembling
C	: number of candidates generating for LM-reranking

Table 2: List of notations used in this paper.

where $k = \phi(w_i)$. Note that the second term of RHS indicates our additional subword features, and the first represents the standard word embedding extraction procedure. Among several choices, we used the byte-pair encoding (BPE) approach proposed in Sennrich et al. (2016) applying 1,000 merge operations[6].

3.2 Unknown token embedding as a bias

We generally replace rare words, e.g., those appearing less than five times in the training data, with *unknown* tokens in the Seq2seq approach. However, we suspect that embedding vectors, which correspond to unknown tokens, cannot be trained well for the following reasons: (1) the occurrence of unknown tokens remains relatively small in the training data since they are obvious replacements for rare words, and (2) Seq2seq is relatively ineffective for training infrequent words (Luong et al., 2015b). Based on these observations, we utilize the unknown embedding as a bias term $\boldsymbol{b}$ of linear layer ($\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$) when obtaining every encoder embeddings for overcoming infrequent word problem. Then, we modify Eq. 2 as follows:

$$\boldsymbol{e}_i = (\boldsymbol{E}\boldsymbol{x}_k + \boldsymbol{u}) + \sum_{k' \in \psi(w_i)} (\boldsymbol{F}\boldsymbol{s}_{k'} + \boldsymbol{u}). \qquad (2)$$

Note that if w_i is unknown token, then Eq. 2 becomes $\boldsymbol{e}_i = 2\boldsymbol{u} + \sum_{k' \in \psi(w_i)} (\boldsymbol{F}\boldsymbol{s}_{k'} + \boldsymbol{u})$.

[6]https://github.com/rsennrich/subword-nmt

3.3 Multi-task learning

Several papers on the Seq2seq approach (Luong et al., 2016) have reported that the multi-task learning extension often improves the task performance if we can find effective auxiliary tasks related to the target task. From this general knowledge, we re-consider jointly estimating POS-tags by incorporating the linearized forms without the POS-tag normalization as an auxiliary task. In detail, the linearized forms with and without the POS-tag normalization are independently and simultaneously estimated as $\boldsymbol{o}_j$ and $\boldsymbol{q}_j$, respectively, in the decoder output layer by following equation:

$$\boldsymbol{o}_j = \boldsymbol{W}^{(o)}\boldsymbol{z}_j, \quad \text{and} \quad \boldsymbol{q}_j = \boldsymbol{W}^{(q)}\boldsymbol{z}_j. \quad (3)$$

3.4 Output length controlling

As described in Vinyals et al. (2015), not all the outputs (predicted linearized parse trees) obtained from the Seq2seq parser are valid (well-formed) as a parse tree. Toward guaranteeing that every output is a valid tree, we introduce a simple extension of the method for controlling the Seq2seq output length (Kikuchi et al., 2016).

First, we introduce an additional bias term $\boldsymbol{b}$ in the decoder output layer to prevent the selection of certain output words:

$$\boldsymbol{p}_j = \texttt{softmax}(\boldsymbol{o}_j + \boldsymbol{b}). \quad (4)$$

If we set a large negative value at the m-th element in $\boldsymbol{b}$, namely $b_m \approx -\infty$, then the m-th element in $\boldsymbol{p}_j$ becomes approximately 0, namely $p_{j,m} \approx 0$, regardless of the value of the k-th element in $\boldsymbol{o}_j$. We refer to this operation to set value $-\infty$ in $\boldsymbol{b}$ as a mask. Since this naive masking approach is harmless to GPU-friendly processing, we can still exploit GPU parallelization.

We set $\boldsymbol{b}$ to always mask the EOS-tag and change $\boldsymbol{b}$ when at least one of the following conditions is satisfied: (1) if the number of open and closed brackets generated so far is the same, then we mask the XX-tags (or the POS-tags) and all the *closed* brackets. (2) if the number of predicted XX-tags (or POS-tags) is equivalent to that of the words in a given input sentence, then we mask the XX-tags (or all the POS-tags) and all the *open* brackets. If both conditions (1) and (2) are satisfied, then the decoding process is finished. The additional cost for controlling the mask is to count the number of XX-tags and the open and closed brackets so far generated in the decoding process.

Dim. of embedding D	300	Dim. of hidden state H	200
Encoder RNN unit	bi-LSTM	Num. of layers L	2
Decoder RNN unit	LSTM with attention	Dropout rate	0.3
Optimizer	SGD	Gradient clipping G	1.0
Learning rate decay	0.9 (after 50 epoch)	Initial learning rate	1.0
Mini-batch size M	16 (shuffled at each epoch)		
Stopping criterion	100 epochs (w/o early stopping)		
Beam size (at Test) B	5		

Table 3: List of model and optimization configurations (hyper-parameters) in our experiments

3.5 Pre-trained word embeddings

The pre-trained word embeddings obtained from a large external corpora often boost the final task performance even if they only initialize the input embedding layer. In constituency parsing, several systems also incorporate pre-trained word embeddings, such as Vinyals et al. (2015); Durrett and Klein (2015). To maintain as much reproducibility of our experiments as possible, we simply applied publicly available pre-trained word embeddings, i.e., `glove.840B.300d`[7], as initial values of the encoder embedding layer.

3.6 Model ensemble

Ensembling several independently trained models together significantly improves many NLP tasks. In the ensembling process, we predict the output tokens using the arithmetic mean of predicted probabilities computed by each model:

$$\boldsymbol{p}_j = \frac{1}{A}\sum_{a=1}^{A}\boldsymbol{p}_j^{(a)}, \quad (5)$$

where $\boldsymbol{p}_j^{(a)}$ represents the probability distribution at position j predicted by the a-th model.

3.7 Language model (LM) reranking

Choe and Charniak (2016) demonstrated that reranking the predicted parser output candidates with an RNN language model (LM) significantly improves performance. We refer to this reranking process as *LM-rerank*. Following their success, we also trained RNN-LMs on the PTB dataset with their published preprocessing code[8] to reproduce the experiments in Choe and Charniak (2016) for our LM-rerank. We selected the current state-of-the-art LM (Yang et al., 2018)[9] as our LM-reranker, which is a much stronger LM than was used in Choe and Charniak (2016).

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

[8] https://github.com/cdg720/emnlp2016

[9] We used the identical hyper-parameters introduced in their site: https://github.com/zihangdai/mos.

ID	Method	category	Development Data (PTB Sec.22) Bracketing F_1 (Bra.F) ave. ±stdev	min / max	Complete match (CM) ave. ±stdev	min / max	Test Data (PTB Sec.23) Bra.F ave. ±stdev	CM ave. ±stdev	Bra.F (dev.max	CM model)
(a)	Seq2seq w/ attn (+post-proc for valid parse tree)		88.08 ±0.41	87.27 / 88.72	35.80 ±0.78	34.88 / 37.41	88.13 ±0.22	35.05 ±0.79	88.39	35.97
(b)	(a) + Dec.control (§3.4)	dec. mask.	88.35 ±0.37	87.70 / 88.83	35.89 ±0.80	34.94 / 37.47	–	–	–	–
(c)	(b) + Subword (§3.1)	enc. feature	89.76 ±0.23	89.40 / 90.03	39.79 ±0.79	38.47 / 40.88	–	–	–	–
(d)	(c) + Unk bias (§3.2)	enc. featture	90.10 ±0.24	89.77 / 90.54	40.98 ±0.82	39.59 / 42.18	–	–	–	–
(e)	(d) + Pos (§3.3)	dec. multitask	**90.21 ±0.20**	89.85 / 90.48	**41.09 ±0.98**	39.35 / 42.82	90.38 ±0.28	40.76 ±0.74	**90.62**	**41.39**
(f)	(a) + Pre-trained emb. (§3.5)	enc. initialization	89.99 ±0.17	89.75 / 90.34	40.69 ±0.83	39.41 / 41.76	90.14 ±0.12	40.40 ±0.44	90.32	40.89
(g)	(f) + Dec.control (§3.4)	dec. mask.	90.28 ±0.15	90.10 / 90.55	40.78 ±0.84	39.53 / 41.88	–	–	–	–
(h)	(g) + Subword (§3.1)	enc. feature	90.34 ±0.10	90.20 / 90.53	41.19 ±0.64	40.12 / 42.06	–	–	–	–
(i)	(h) + Unk bias (§3.2)	enc. feature	90.92 ±0.17	90.67 / 91.17	**43.38 ±0.57**	42.47 / 44.29	–	–	–	–
(j)	(i) + Pos (§3.3)	dec. multitask	**90.93 ±0.14**	90.68 / 91.07	42.76 ±0.38	42.00 / 43.18	91.18 ±0.12	42.39 ±0.68	**91.36**	**43.50**

Table 4: Results on English PTB data: Results were average (ave), worst (min), and best (max) performance of ten models independently trained with distinct random initial values. Test data was only evaluated on baseline and our best setting ((a), (e), (f) and (j)) to prevent over-tuning to the test data. We confirmed that all our results contained no malformed parse trees.

ID	Method	Dev. Bra.F	Dev. CM	Test Bra.F	Test CM
(k)	(e) + ensemble $A = 8$ (§3.6)	92.32	45.76	**92.18**	**45.90**
(l)	(k) + LM-rerank $C = 80$ (§3.7)	94.31	53.59	**94.14**	**52.69**
(m)	(j) + ensemble $A = 8$ (§3.6)	92.90	47.85	**92.74**	**47.27**
(n)	(m) + LM-rerank $C = 80$ (§3.7)	94.30	54.12	**94.32**	**52.81**

Table 5: Ensembling and reranking results

(a) Mini-batch size M

method	Bra.F	CM
(j) $M = 16$	**90.93**	**42.76**
$M = 64$	89.85	40.94
$M = 256$	89.41	40.41

(b) Gradient clipping G

method	Bra.F	CM
(j) $G = 1$	**90.93**	42.76
$G = 5$	87.36	36.71

(c) Hidden dim H and layer L

method	Bra.F	CM
(j) $H = 200, L = 2$	**90.93**	42.76
$H = 200, L = 3$	90.75	43.00
$H = 200, L = 4$	90.55	42.84
$H = 512, L = 2$	90.59	**43.38**

(d) Beam size B

method	Bra.F	CM
$B = 1$	90.55	42.49
(j) $B = 5$	**90.93**	**42.76**
$B = 20$	90.98	**42.76**
$B = 50$	91.01	**42.76**

(e) usage of subword information (feature or split)

method	Bra.F	CM
(h) word split with $1K$ subword feature	90.93	42.76
$8K$ subword split	87.39	33.62
$16K$ subword split	87.20	31.21

Table 6: Impact of hyper-parameter selections. We only evaluated the development data (PTB Sec. 22) to prevent over-tuning to the test data.

System (Brief description)	Bra.F
[Trained (strictly) from PTB only, no additional resources]	
(Kamigaito et al., 2017) Seq2seq, sup.attention	89.5
(Cross and Huang, 2016a) Shift-reduce	89.95
Ours; Seq2seq	**90.62**
(Watanabe and Sumita, 2015) Shift-reduce	90.68
(Shindo et al., 2012)	91.1
(Cross and Huang, 2016b) Shift-reduce	91.3
(Kamigaito et al., 2017) Seq2seq, sup.attention, ensemble	91.5
(Dyer et al., 2016) Shift-reduce, discriminative	91.7
(Liu and Zhang, 2017) Shift-reduce	91.7
(Stern et al., 2017a) Top-down	91.79
Ours; Seq2seq, ensemble	**92.18**
(Shindo et al., 2012) ensemble	92.4
(Stern et al., 2017b) Top-down, rerank	92.56
(Choe and Charniak, 2016) CKY, LM-rerank	92.6
(Dyer et al., 2016) Shift-reduce, generative	93.3
(Kuncoro et al., 2017) Shift-reduce, rerank	93.6
Ours; Seq2seq, ensemble, LM-rerank(80)	**94.14**
(Fried et al., 2017) Shift-reduce, ensemble, rerank	94.25
[PTB only, but utilizing pre-trained emb. from external corpus for init.]	
(Vinyals et al., 2015) Seq2seq	88.3
(Vinyals et al., 2015) Seq2seq, ensemble	90.5
(Durrett and Klein, 2015) CKY	91.1
Ours; Seq2seq	**91.36**
Ours; Seq2seq, ensemble	**92.74**
Ours best; Seq2seq, ensemble, LM-rerank(80)	**94.32**
[Trained from PTB and other external silver data]	
(Choe and Charniak, 2016) CKY, LM-rerank	93.8
(Fried et al., 2017) Shift-reduce, ensemble, rerank	94.66

Table 7: List of bracketing F-measures on test data (PTB Sec.23) reported in recent top-notch systems: scores with bold font represent our scores.

4 Experiments

Our experiments used the English Penn Treebank data (Marcus et al., 1994), which are the most widely used benchmark data in the literature. We used the standard split of training (Sec.02–21), development (Sec.22), and test data (Sec.23) and strictly followed the instructions for the evaluation settings explained in Vinyals et al. (2015). For data pre-processing, all the parse trees were transformed into linearized forms, which include standard *UNK* replacement for OOV words and *POS-tag normalization* by XX-tags. As explained in Vinyals et al. (2015), we did not apply any parse tree binarization or special unary treatment, which were used as common techniques in the literature.

Table 3 summarizes the model configurations and the optimization settings used in our experiments unless otherwise specified.

4.1 Results

Table 4 shows the main results of our experiments. We reported the Bracketing F-measures (Bra.F) and the complete match scores (CM) evaluated by the EVALB tool[10]. The averages (ave), standard deviations (stdev), lowest (min), and highest (max) scores were calculated from ten independent runs of each setting trained with different random initialization values. This table empirically reveals the effectiveness of individual techniques. Each technique gradually improved the performance, and the best result (j) achieved ap-

[10] http://nlp.cs.nyu.edu/evalb/

proximately 3 point gain from the baseline conventional Seq2seq model (a) on test data Bra.F.

One drawback of Seq2seq approach is that it seems sensitive to initialization. Comparing only with a single result for each setting may produce inaccurate conclusions. Therefore, we should evaluate the performances over several trials to improve the evaluation reliability.

The baseline Seq2seq models, (a) and (f), produced the malformed parse trees. We post-processed such malformed parse trees by simple rules introduced in (Vinyals et al., 2015). On the other hand, we confirmed that all the results applying the technique explained in Sec. 3.4 produced no malformed parse trees.

Ensembling and Reranking: Table 5 shows the results of our models with model ensembling and LM-reranking. For ensemble, we randomly selected eight of the ten Seq2seq models reported in Table 4. For LM-reranking, we first generated 80 candidates by the above eight ensemble models and selected the best parse tree for each input in terms of the LM-reranker. The results in Table 5 were taken from a single-shot evaluation, unlike the averages of ten independent runs in Table 4.

Hyper-parameter selection: We empirically investigated the impact of the hyper-parameter selections. Table 6 shows the results. The following observations appear informative for building strong baseline systems: (1) Smaller mini-batch size M and gradient clipping G provided the better performance. Such settings lead to slower and longer training, but higher performance. (2) Larger layer size, hidden state dimension, and beam size have little impact on the performance; our setting, $L = 2$, $H = 200$, and $B = 5$ looks adequate in terms of speed/performance trade-off.

Input unit selection: As often demonstrated in the NMT literature, using subword split as input token unit instead of standard tokenized word unit has potential to improve the performance. Table 6 (e) shows the results of utilizing subword splits. Clearly, $8K$ and $16K$ subword splits as input token units significantly degraded the performance. It seems that the numbers of XX-tags in output and tokens in input should keep consistent for better performance since Seq2seq models look to somehow learn such relationship, and used it during the decoding. Thus, using subword information as features is one promising approach for leveraging subword information into constituency parsing.

4.2 Comparison to current top systems

Table 7 lists the reported constituency parsing scores on PTB that were recently published in the literature. We split the results into three categories. The first category (top row) contains the results of the methods that were trained only from the pre-defined training data (PTB Sec.02–21), *without* any additional resources. The second category (middle row) consists of the results of methods that were trained from the pre-defined PTB training data as well as those listed in the top row, but incorporating word embeddings obtained from a large-scale external corpus to initialize the encoder embedding layer. The third category (bottom row) shows the performance of the methods that were trained using high-confidence, auto-parsed trees in addition to the pre-defined PTB training data.

Our Seq2seq approach successfully achieved the competitive level as the current top-notch methods: RNNG and its variants. Note here that, as described in Dyer et al. (2016), RNNG uses Berkeley parser's mapping rules for effectively handling singleton words in the training corpus. In contrast, we demonstrated that Seq2seq models have enough power to achieve a competitive state-of-the-art performance without leveraging such task-dependent knowledge. Moreover, they need no explicit information of parse tree structures, transition states, stacks, (Stanford or Berkeley) mapping rules, or external silver training data during the model training except general purpose word embeddings as initial values. These observations from our experiments imply that recently developed Seq2seq models have enough ability to implicitly learn parsing structures from linearized parse trees. Our results argue that Seq2seq models can be a strong baseline for constituency parsing.

5 Conclusion

This paper investigated how well general purpose Seq2seq models can achieve the higher performance of constituency parsing as a strong baseline method. We incorporated several generic techniques to enhance Seq2seq models, such as incorporating subword features, and output length controlling. We experimentally demonstrated that by applying ensemble and LM-reranking techniques, a general purpose Seq2seq model achieved almost the same performance level as the state-of-the-art constituency parser without any task-specific or explicit tree structure information.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*, 5:135–146.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2331–2336.

James Cross and Liang Huang. 2016a. Incremental parsing with minimal features using bi-directional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 32–37.

James Cross and Liang Huang. 2016b. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–11.

Michael Denkowski and Graham Neubig. 2017. Stronger baselines for trustable results in neural machine translation. In *Proceedings of the 1st Workshop on Neural Machine Translation (WNMT)*.

Greg Durrett and Dan Klein. 2015. Neural CRF parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL) and the 7th International Joint Conference on Natural Language Processing*, pages 302–312.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 North American Chapter of the Association for Computational Linguistics*, pages 199–209.

Daniel Fried, Mitchell Stern, and Dan Klein. 2017. Improving neural parsing by disentangling model combination and reranking effects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 161–166.

Hidetaka Kamigaito, Katsuhiko Hayashi, Tsutomu Hirao, Hiroya Takamura, Manabu Okumura, and Masaaki Nagata. 2017. Supervised attention for sequence-to-sequence constituency parsing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 7–12, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1328–1338.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proceedings of the 15th European Chapter of the Association for Computational Linguistics (EACL)*, pages 1249–1258.

Jiangming Liu and Yue Zhang. 2017. Shift-reduce constituent parsing with neural lookahead features. *Transactions of the Association for Computational Linguistics*, 5:45–58.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Minh-Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL) and the 7th International Joint Conference on Natural Language Processing*.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 379–389.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1715–1725.

Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. 2012. Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–448. Association for Computational Linguistics.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017a. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 818–827.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017b. Effective inference for generative neural parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1695–1700.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS)*.

Oriol Vinyals, Ł ukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a Foreign Language. *Advances in Neural Information Processing Systems 28*, pages 2773–2781.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based neural constituent parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL) and the 7th International Joint Conference on Natural Language Processing*, pages 1169–1179.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.