Forest-Based Neural Machine Translation

Chunpeng Ma[1,3*] **Akihiro Tamura**[2] **Masao Utiyama**[3] **Tiejun Zhao**[1†] **Eiichiro Sumita**[3]
[1]Harbin Institute of Technology, Harbin, China
[2]Ehime University, Matsuyama, Japan
[3]National Institute of Information and Communications Technology, Kyoto, Japan
{cpma, tjzhao}@hit.edu.cn tamura@cs.ehime-u.ac.jp
{mutiyama, eiichiro.sumita}@nict.go.jp

Abstract

Tree-based neural machine translation (NMT) approaches, although achieved impressive performance, suffer from a major drawback: they only use the 1-best parse tree to direct the translation, which potentially introduces translation mistakes due to parsing errors. For statistical machine translation (SMT), forest-based methods have been proven to be effective for solving this problem, while for NMT this kind of approach has not been attempted. This paper proposes a forest-based NMT method that translates a linearized packed forest under a simple sequence-to-sequence framework (i.e., a forest-to-string NMT model). The BLEU score of the proposed method is higher than that of the string-to-string NMT, tree-based NMT, and forest-based SMT systems.

1 Introduction

NMT has witnessed promising improvements recently. Depending on the types of input and output, these efforts can be divided into three categories: *string-to-string* systems (Sutskever et al., 2014; Bahdanau et al., 2014); *tree-to-string* systems (Eriguchi et al., 2016, 2017); and *string-to-tree* systems (Aharoni and Goldberg, 2017; Nadejde et al., 2017). Compared with string-to-string systems, tree-to-string and string-to-tree systems (henceforth, *tree-based systems*) offer some attractive features. They can use more syntactic information (Li et al., 2017), and can conveniently incorporate prior knowledge (Zhang et al., 2017). Because of these advantages, tree-based methods become the focus of many researches of NMT nowadays.

Based on how to represent trees, there are two main categories of tree-based NMT methods: representing trees by a tree-structured neural network (Eriguchi et al., 2016; Zaremoodi and Haffari, 2017), representing trees by linearization (Vinyals et al., 2015; Dyer et al., 2016; Ma et al., 2017). Compared with the former, the latter method has a relatively simple model structure, so that a larger corpus can be used for training and the model can be trained within reasonable time, hence is preferred from the viewpoint of computation. Therefore we focus on this kind of methods in this paper.

In spite of impressive performance of tree-based NMT systems, they suffer from a major drawback: they only use the 1-best parse tree to direct the translation, which potentially introduces translation mistakes due to parsing errors (Quirk and Corston-Oliver, 2006). For SMT, forest-based methods have employed a *packed forest* to address this problem (Huang, 2008), which represents exponentially many parse trees rather than just the 1-best one (Mi et al., 2008; Mi and Huang, 2008). But for NMT, (computationally efficient) forest-based methods are still being explored[1].

Because of the structural complexity of forests, the inexistence of appropriate topological ordering, and the hyperedge-attachment nature of weights (see Section 3.1 for details), it is not trivial to linearize a forest. This hinders the development of forest-based NMT to some extent.

Inspired by the tree-based NMT methods based on linearization, we propose an efficient forest-based NMT approach (Section 3), which can en-

[*] Contribution during internship at National Institute of Information and Communications Technology.

[†] Corresponding author

[1] Zaremoodi and Haffari (2017) have proposed a forest-based NMT method based on a forest-structured neural network recently, but it is computationally *in*efficient (see Section 5).

code the syntactic information of a *packed forest* on the basis of a novel *weighted* linearization method for a packed forest (Section 3.1), and can decode the linearized packed forest under the simple sequence-to-sequence framework (Section 3.2). Experiments demonstrate the effectiveness of our method (Section 4).

2 Preliminaries

We first review the general sequence-to-sequence model (Section 2.1), then describe tree-based NMT systems based on linearization (Section 2.2), and finally introduce the packed forest, through which exponentially many trees can be represented in a compact manner (Section 2.3).

2.1 Sequence-to-sequence model

Current NMT systems usually resort to a simple framework, i.e., the sequence-to-sequence model (Cho et al., 2014; Sutskever et al., 2014). Given a source sequence $(x_0, \ldots, x_T)$, in order to find a target sequence $(y_0, \ldots, y_{T'})$ that maximizes the conditional probability $p(y_0, \ldots, y_{T'} \mid x_0, \ldots, x_T)$, the sequence-to-sequence model uses one RNN to encode the source sequence into a fixed-length context vector c and a second RNN to decode this vector and generate the target sequence. Formally, the probability of the target sequence can be calculated as follows:

$$p(y_0, \ldots, y_{T'} \mid x_0, \ldots, x_T) = \prod_{t=0}^{T'} p(y_t \mid c, y_0, \ldots, y_{t-1}), \quad (1)$$

where

$$p(y_t \mid c, y_0, \ldots, y_{t-1}) = g(y_{t-1}, s_t, c), \quad (2)$$

$$s_t = f(s_{t-1}, y_{t-1}, c), \quad (3)$$

$$c = q(h_0, \ldots, h_T), \quad (4)$$

$$h_t = f(e_t, h_{t-1}). \quad (5)$$

Here, g, f, and q are nonlinear functions; h_t and s_t are the hidden states of the source-side RNN and target-side RNN, respectively, c is the context vector, and e_t is the embedding of x_t.

Bahdanau et al. (2014) introduced an attention mechanism to deal with the issues related to long sequences (Cho et al., 2014). Instead of encoding the source sequence into a fixed vector c, the attention model uses different c_i-s when calculating the target-side output y_i at time step i:

$$c_i = \sum_{j=0}^{T} \alpha_{ij} h_j, \quad (6)$$

$$\alpha_{ij} = \frac{\exp(a(s_{i-1}, h_j))}{\sum_{k=0}^{T} \exp(a(s_{i-1}, h_k))}. \quad (7)$$

The function $a(s_{i-1}, h_j)$ can be regarded as representing the soft alignment between the target-side RNN hidden state s_{i-1} and the source-side RNN hidden state h_j.

By changing the format of the source/target sequences, this framework can be regarded as a string-to-string NMT system (Sutskever et al., 2014), a tree-to-string NMT system (Li et al., 2017), or a string-to-tree NMT system (Aharoni and Goldberg, 2017).

2.2 Linear-structured tree-based NMT systems

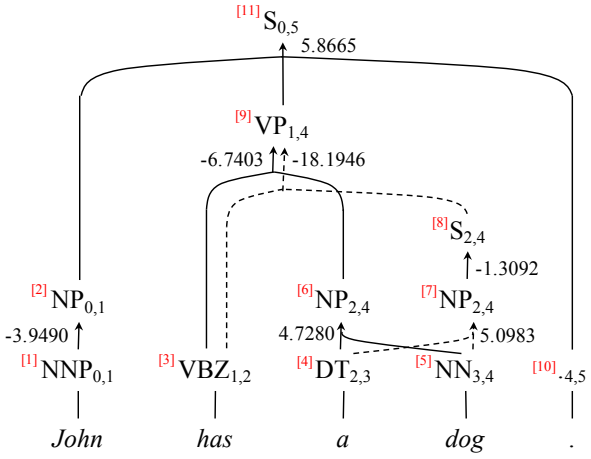
Regarding the linearization adopted for tree-to-string NMT (i.e., linearization of the source side), Sennrich and Haddow (2016) encoded the sequence of dependency labels and the sequence of words simultaneously, partially utilizing the syntax information, while Li et al. (2017) traversed the constituent tree of the source sentence and combined this with the word sequence, utilizing the syntax information completely.

Regarding the linearization used for string-to-tree NMT (i.e., linearization of the target side), Nadejde et al. (2017) used a CCG supertag sequence as the target sequence, while Aharoni and Goldberg (2017) applied a linearization method in a top-down manner, generating a sequence ensemble for the annotated tree in the Penn Treebank (Marcus et al., 1993). Wu et al. (2017) used transition actions to linearize a dependency tree, and employed the sequence-to-sequence framework for NMT.

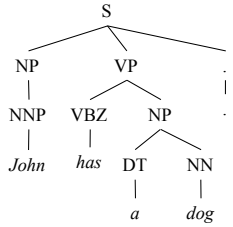
It can be seen all current tree-based NMT systems use only one tree for encoding or decoding. In contrast, we hope to utilize multiple trees (i.e., a forest). This is not trivial, on account of the lack of a fixed traversal order and the need for a compact representation.

2.3 Packed forest

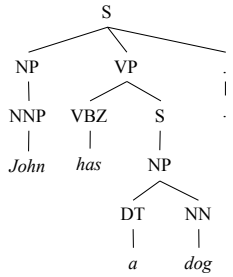
The *packed forest* gives a representation of exponentially many parsing trees, and can compactly encode many more candidates than the n-best list



(a) Packed forest



(b) Correct constituent tree, $score = -46.2389$



(c) Incorrect constituent tree, $score = -58.6321$

Figure 1: An example of (a) a packed forest. The numbers in the brackets located at the upper-left corner of each node in the packed forest show one correct topological ordering of the nodes. The packed forest is a compact representation of two trees: (b) the correct constituent tree, and (c) an incorrect constituent tree. Note that the terminal nodes (i.e., words in the sentence) in the packed forest are shown only for illustration, and they do not belong to the packed forest.

(Huang, 2008). Figure 1a shows a packed forest, which can be unpacked into two constituent trees (Figure 1b and Figure 1c).

Formally, a packed forest is a pair $\langle V, E \rangle$, where V is the set of nodes and E is the set of hyperedges. Each $v \in V$ can be represented as $X_{i,j}$, where X is a constituent label and $i, j \in [0, n]$ are indices of words, showing that the node spans the words ranging from i (inclusive) to j (exclusive). Here, n is the length of the input sentence. Each $e \in E$ is a three-tuple $\langle head(e), tails(e), score(e) \rangle$, where $head(e) \in V$ is similar to the head node in a constituent tree, and $tails(e) \in V^*$ is similar to the set of child nodes in a constituent tree. $score(e) \in \mathbb{R}$ is the logarithm of the probability that $tails(e)$ represents the tails of $head(e)$ calculated by the parser. Based on $score(e)$, the score of a constituent tree T can be calculated as follows:

$$score(T) = -\lambda n + \sum_{e \in E(T)} score(e), \quad (8)$$

where $E(T)$ is the set of hyperedges appearing in tree T, and λ is a regularization coefficient for the sentence length[2].

[2]Following the configuration of Charniak and Johnson (2005), for all the experiments in this paper, we fixed λ to $\log_2 600$.

3 Forest-based NMT

We first propose a linearization method for the packed forest (Section 3.1), then describe how to encode the linearized forest (Section 3.2), which can then be translated by the conventional decoder (see Section 2.1).

3.1 Forest linearization

Recently, several studies have focused on the linearization methods of a syntax tree, both in the area of tree-based NMT (Section 2.2) and in the area of parsing (Vinyals et al., 2015; Dyer et al., 2016; Ma et al., 2017). Basically, these methods follow a fixed traversal order (e.g., depth-first), which does not exist for the packed forest (a directed acyclic graph (DAG)). Furthermore, the weights are attached to edges of a packed forest instead of the nodes, which further increase the difficulty.

Topological ordering algorithms for DAG (Kahn, 1962; Tarjan, 1976) are not good solutions, because the outputted ordering is not always optimal for machine translation. In particular, a topo-

Algorithm 1 Linearization of a packed forest

1: **function** LINEARIZEFOREST($\langle V, E\rangle$, $\mathbf{w}$)
2: $v \leftarrow$ FINDROOT(V)
3: $\mathbf{r} \leftarrow []$
4: EXPANDSEQ(v, $\mathbf{r}$, $\langle V, E\rangle$, $\mathbf{w}$)
5: **return** $\mathbf{r}$
6: **function** FINDROOT(V)
7: **for** $v \in V$ **do**
8: **if** v has no parent **then**
9: **return** v
10: **procedure** EXPANDSEQ(v, $\mathbf{r}$, $\langle V, E\rangle$, $\mathbf{w}$)
11: **for** $e \in E$ **do**
12: **if** $head(e) = v$ **then**
13: **if** $tails(e) \neq \emptyset$ **then**
14: **for** $t \in$ SORT($tails(e)$) **do** ▷ Sort $tails(e)$ by word indices.
15: EXPANDSEQ(t, $\mathbf{r}$, $\langle V, E\rangle$, $\mathbf{w}$)
16: $l \leftarrow$ LINEARIZEEDGE($head(e)$, $\mathbf{w}$)
17: $\mathbf{r}$.append($\langle l, \sigma(0.0)\rangle$) ▷ σ is the sigmoid function, i.e., $\sigma(x) = \frac{1}{1+e^{-x}}, x \in \mathbb{R}$.
18: $l \leftarrow$ ©LINEARIZEEDGES($tails(e)$, $\mathbf{w}$) ▷ © is a unary operator.
19: $\mathbf{r}$.append($\langle l, \sigma(score(e))\rangle$)
20: **else**
21: $l \leftarrow$ LINEARIZEEDGE($head(e)$, $\mathbf{w}$)
22: $\mathbf{r}$.append($\langle l, \sigma(0.0)\rangle$)
23: **function** LINEARIZEEDGE($X_{i,j}$, $\mathbf{w}$)
24: **return** $X \otimes (\odot_{k=i}^{j-1} w_k)$
25: **function** LINEARIZEEDGES($\mathbf{v}$, $\mathbf{w}$)
26: **return** $\oplus_{v \in \mathbf{v}}$LINEARIZEEDGE($v$, $\mathbf{w}$)

logical ordering could ignore "word sequential information" and "parent-child information" in the sentences.

For example, for the packed forest in Figure 1a, although "[10]→[1]→[2]→ ··· →[9]→[11]" is a valid topological ordering, the *word sequential information* of the words (e.g., "John" should be located ahead of the period), which is fairly crucial for translation of languages with fixed pragmatic word order such as Chinese or English, is lost.

As another example, for the packed forest in Figure 1a, nodes [2], [9], and [10] are all the children of node [11]. However, in the topological order "[1]→[2]→ ··· →[9]→[10]→[11]," node [2] is quite far from node [11], while nodes [9] and [10] are both close to node [11]. The *parent-child information* cannot be reflected in this topological order, which is not what we would expect.

To address the above two problems, we propose a novel linearization algorithm for a packed forest (Algorithm 1). The algorithm linearizes the packed forest from the root node (Line 2) to leaf nodes by calling the EXPANDSEQ procedure (Line 15) recursively, while preserving the word order in the sentence (Line 14). In this way, *word sequential information* is preserved. Within the

NNP⊗*John* / NP⊗*John* / ©NNP⊗*John* / VBZ⊗*has* / DT⊗*a* / NN⊗*dog* / NP⊗*a*⊙*dog* / ©DT⊗*a*⊕NN⊗*dog* / NP⊗*a*⊙*dog* / ©DT⊗*a*⊕NN⊗*dog* / S⊗*a*⊙*dog* / ©NP⊗*a*⊙*dog* / VP⊗*has*⊙*a*⊙*dog* / ©VBZ⊗*has*⊕NP⊗*a*⊙*dog* / ©VBZ⊗*has*⊕S⊗*a*⊙*dog* / .⊗. / S⊗*John*⊙*has*⊙*a*⊙*dog*⊙. / ©NP⊗*John*⊕VP⊗*has*⊙*a*⊙*dog*⊕.⊗.

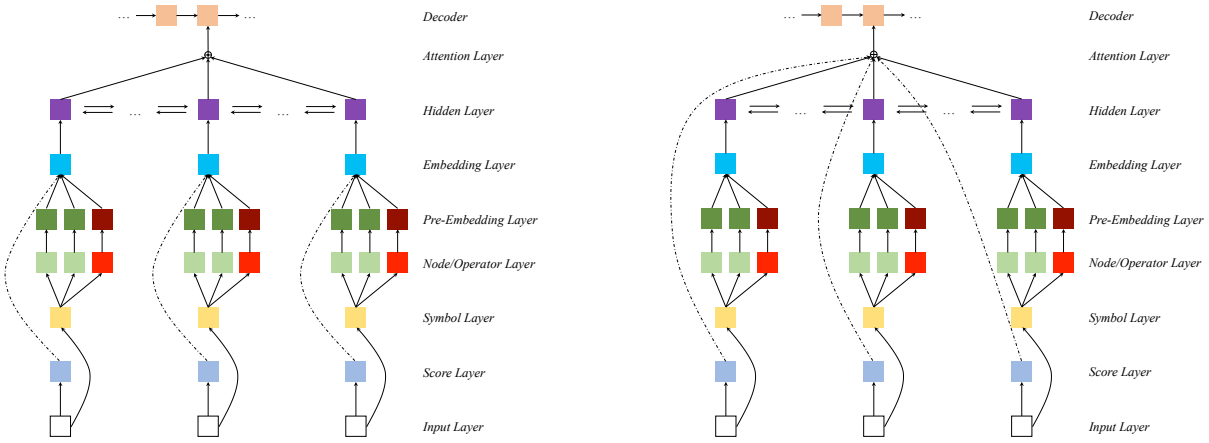
Figure 2: Linearization result of the packed forest in Figure 1a

EXPANDSEQ procedure, once a hyperedge is linearized (Line 16), the tails are also linearized immediately (Line 18). In this way, *parent-child information* is preserved. Intuitively, different parts of constituent trees should be combined in different ways, therefore we define different operators (©, ⊗, ⊕, or ⊙) to represent the relationships between different parts, so that the representations of these parts can be combined in different ways (see Section 3.2 for details). Words are concatenated by the operator "⊙" with each other, a word and a constituent label is concatenated by the operator "⊗", the linearization results of child nodes are concatenated by the operator "⊕" with each other, while the unary operator "©" is used to indicate that the node is the child node of the previous part. Furthermore, each token in the linearized sequence is related to a score, representing the confidence of the parser.

The linearization result of the packed forest in Figure 1a is shown in Figure 2. Tokens in the linearized sequence are separated by slashes. Each token in the sequence is composed of different types of symbols and combined by different operators. We can see that word sequential information is preserved. For example, "NNP⊗*John*" (linearization result of node [1]) is in front of "VBZ⊗*has*" (linearization result of node [3]), which is in front of "DT⊗*a*" (linearization result of node [4]). Moreover, parent-child information is also preserved. For example, "NP⊗*John*" (linearization result of node [2]) is followed by "©NNP⊗*John*" (linearization result of node [1], the child of node [2]).

Note that our linearization method cannot fully recover packed forest. What we want to do is *not* to propose a fully recoverable linearization method. What we *actually* want to do is to encode syntax information as much as possible, so that we can improve the performance of NMT. As will be shown in Section 4, this goal is achieved.

Also note that there is one more advantage of our linearization method: the linearized sequence



(a) Score-on-Embedding (SoE) (b) Score-on-Attention (SoA)

Figure 3: The framework of the forest-based NMT system.

is a weighted sequence, while all the previous studies ignored the weights during linearization. As will be shown in Section 4, the weights are actually important not only for the linearization of a packed forest, but also for the linearization of a single tree.

By preserving only the nodes and hyperedges in the 1-best tree and removing all others, our linearization method can be regarded as a tree-linearization method. Compared with other tree-linearization methods, our method combines several different kinds of information within one symbol, retaining the parent-child information, and incorporating the confidence of the parser in the sequence. We examine whether the weights can be useful not only for linear structured tree-based NMT but also for our forest-based NMT.

Furthermore, although our method is nonreversible for packed forests, it is reversible for constituent trees, in that the linearization is processed exactly in the depth-first traversal order and all necessary information in the tree nodes has been encoded. As far as we know, there is no previous work on linearization of packed forests.

3.2 Encoding the linearized forest

The linearized packed forest forms the input of the encoder, which has two major differences from the input of a sequence-to-sequence NMT system. First, the input sequence of the encoder consists of two parts: the symbol sequence and the score sequence. Second, each symbol in the symbol sequence consists of several parts (words and constituent labels), which are combined by certain operators (©, $\otimes$, $\oplus$, or $\odot$). Based on these observations, we propose two new frameworks, which are illustrated in Figure 3.

Formally, the input layer receives the sequence $(\langle l_0, \xi_0\rangle, \ldots, \langle l_T, \xi_T\rangle)$, where l_i denotes the i-th symbol and ξ_i its score. Then, the sequence is fed into the score layer and the symbol layer. The score and symbol layers receive the sequence and output the score sequence $\xi = (\xi_0, \ldots, \xi_T)$ and symbol sequence $\mathbf{l} = (l_0, \ldots, l_T)$, respectively, from the input. Any item $l \in \mathbf{l}$ in the symbol layer has the form

$$l = o_0 x_1 o_1 \ldots x_{m-1} o_{m-1} x_m, \tag{9}$$

where each x_k $(k = 1, \ldots, m)$ is a word or a constituent label, m is the total number of words and constituent labels in a symbol, o_0 is "©" or empty, and each o_k $(k = 1, \ldots, m-1)$ is either "$\otimes$", "$\oplus$", or "$\odot$". Then, in the node/operator layer, the x-s and o-s are separated and rearranged as $\mathbf{x} = (x_1, \ldots, x_m, o_0, \ldots, o_{m-1})$, which is fed to the pre-embedding layer. The pre-embedding layer generates a sequence $\mathbf{p} = (p_1, \ldots, p_m, \ldots, p_{2m})$, which is calculated as follows:

$$\mathbf{p} = W_{emb}[I(\mathbf{x})]. \tag{10}$$

Here, the function $I(\mathbf{x})$ returns a list of the indices in the dictionary for all the elements in $\mathbf{x}$, which consist of words, constituent labels, or operators. In addition, W_{emb} is the embedding matrix of size $(|w_{word}| + |w_{label}| + 4) \times d_{word}$, where $|w_{word}|$ and $|w_{label}|$ are the total number of words and constituent labels, respectively, d_{word} is the dimension of the word embedding, and there are four possible operators: "©," "$\otimes$," "$\oplus$," and "$\odot$." Note

that $\mathbf{p}$ is a list of $2m$ vectors, and the dimension of each vector is d_{word}.

Because the length of the sequence of the input layer is $T+1$, there are $T+1$ different $\mathbf{p}$-s in the pre-embedding layer, which we denote by $\mathbf{P} = (\mathbf{p}_0, \ldots, \mathbf{p}_T)$. Depending on where the score layer is incorporated, we propose two frameworks: Score-on-Embedding (SoE) and Score-on-Attention (SoA). In SoE, the k-th element of the embedding layer is calculated as follows:

$$e_k = \xi_k \sum_{p \in \mathbf{p}_k} p, \tag{11}$$

while in SoA, the k-th element of the embedding layer is calculated as

$$e_k = \sum_{p \in \mathbf{p}_k} p, \tag{12}$$

where $k = 0, \ldots, T$. Note that $e_k \in \mathbb{R}^{d_{word}}$. In this manner, the proposed forest-to-string NMT framework is connected with the conventional sequence-to-sequence NMT framework.

After calculating the embedding vectors in the embedding layer, the hidden vectors are calculated using Equation 5. When calculating the context vector c_i-s, SoE and SoA differ from each other. For SoE, the c_i-s are calculated using Equation 6 and 7, while for SoA, the α_{ij}-s used to calculate the c_i-s are determined as follows:

$$\alpha_{ij} = \frac{\exp(\xi_j a(s_{i-1}, h_j))}{\sum_{k=0}^{T} \exp(\xi_k a(s_{i-1}, h_k))}. \tag{13}$$

Then, using the decoder of the sequence-to-sequence framework, the sentence of the target language can be generated.

4 Experiments

4.1 Setup

We evaluate the effectiveness of our forest-based NMT systems on English-to-Chinese and English-to-Japanese translation tasks[3]. The statistics of the corpora used in our experiments are summarized in Table 1.

The packed forests of English sentences are obtained by the constituent parser proposed by Huang (2008)[4]. We filtered out the sentences for which the parser cannot generate the packed forest successfully and the sentences longer than 80 words. For NIST datasets, we simply choose the first reference among the four English references of NIST corpora, because all of them are independent with each other, according to the documents of NIST datasets. For Chinese sentences, we used Stanford segmenter[5] for segmentation. For Japanese sentences, we followed the preprocessing steps recommended in WAT 2017[6].

Language	Corpus	Usage	#Sent.
English-Japanese	ASPEC	train	100,000
		dev.	1790
		test	1812
English-Chinese	LDC[7]	train	1,423,695
	FBIS		233,510
	NIST MT 02	dev.	876
	NIST MT 03	test	919
	NIST MT 04		1,788
	NIST MT 05		1,082

Table 1: Statistics of the corpora.

We implemented our framework based on `nematus`[8] (Sennrich et al., 2017). For optimization, we used the Adadelta algorithm (Zeiler, 2012). In order to avoid overfitting, we used dropout (Srivastava et al., 2014) on the embedding layer and hidden layer, with the dropout probability set to 0.2. We used the gated recurrent unit (Cho et al., 2014) as the recurrent unit of RNNs, which are bi-directional, with one hidden layer.

Based on the tuning result, we set the maximum length of the input sequence to 300, the hidden layer size as 512, the dimension of word embedding as 620, and the batch size for training as 40. We pruned the packed forest using the algorithm of Huang (2008), with a threshold of 5. If the linearization of the pruned forest is still longer than 300, then we linearize the 1-best parsing tree instead of the forest. During decoding, we used beam search, and fixed the beam size to 12. For the case of Forest (SoA), with 1 core of Tesla K80 GPU and LDC corpus as the training data, training spent about 10 days, and decoding speed is about 10 sentences per second.

[3] English is commonly chosen as the *target* language. We chose English as the *source* language because a high-performance forest parser is not available for other languages.

[4] http://web.engr.oregonstate.edu/~huanlian/software/forest-reranker/forest-charniak-v0.8.tar.bz2

[5] https://nlp.stanford.edu/software/stanford-segmenter-2017-06-09.zip

[6] http://lotus.kuee.kyoto-u.ac.jp/WAT/WAT2017/baseline/dataPreparationJE.html

[7] LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08, and LDC2005T06

[8] https://github.com/EdinburghNLP/nematus

Types	Systems & Configurations	MT 03		MT 04		MT 05		p value
		FBIS	LDC	FBIS	LDC	FBIS	LDC	
FS	Mi et al. (2008)	27.10	28.21	28.67	30.09	26.57	28.36	-
TN	Eriguchi et al. (2016)	29.00	29.71	30.24	31.56	28.38	30.33	-
	Chen et al. (2017)	28.34	29.64	30.00	31.25	28.14	29.59	-
	Li et al. (2017)	28.40	29.60	29.66	31.96	27.74	29.84	-
FN	s2s	27.44	29.18	29.73	30.53	27.32	28.80	-
	1-best (No score)	28.61	29.38	30.07	31.58	28.59	30.01	< 0.01
	1-best (SoE)	28.78	30.65	30.36	32.22	29.31	30.16	< 0.05
	1-best (SoA)	29.39	30.80	30.25	32.39	29.30	30.61	< 0.005
	Forest (No score)	28.06	29.63	29.51	31.41	28.48	29.75	< 0.01
	Forest (SoE)	29.58	31.07	**30.67**	32.69	29.26	30.41	< 0.001
	Forest (SoA)	**29.63**	**31.35**	30.31	**33.14**	**29.87**	**31.23**	< 0.001

Table 2: English-Chinese experimental results (character-level BLEU). "FS," "TN," and "FN" denote forest-based SMT, tree-based NMT, and forest-based NMT systems, respectively. We performed the paired bootstrap resampling significance test (Koehn, 2004) over the NIST MT 03 to 05 corpus, with respect to the s2s baseline, and list the p values in the table.

Types	Systems & Configurations	BLEU (test)	p value
FS	Mi et al. (2008)	34.13	-
TN	Eriguchi et al. (2016)	37.52	-
	Chen et al. (2017)	36.94	-
	Li et al. (2017)	36.21	-
FN	s2s	37.10	-
	1-best (No score)	38.01	< 0.05
	1-best (SoE)	38.53	< 0.01
	1-best (SoA)	39.42	< 0.001
	Forest (No score)	37.92	< 0.1
	Forest (SoE)	41.35	< 0.01
	Forest (SoA)	**42.17**	< 0.005

Table 3: English-Japanese experimental results (character-level BLEU).

4.2 Experimental results

Table 2 and 3 summarize the experimental results. To avoid the affect of segmentation errors, the performance were evaluated by character-level BLEU (Papineni et al., 2002). We compare our proposed models (i.e., Forest (SoE) and Forest (SoA)) with three types of baseline: a string-to-string model (s2s), forest-based models that do not use score sequences (Forest (No score)), and tree-based models that use the 1-best parsing tree (1-best (No score, SoE, SoA)). For the 1-best models, we preserve the nodes and hyperedges that are used in the 1-best constituent tree in the packed forest, and remove all other nodes and hyperedges, yielding a pruned forest that contains only the 1-best constituent tree. For the "No score" configurations, we force the input score sequence to be a sequence of 1.0 with the same length as the input symbol sequence, so that neither the embedding layer nor the attention layer are affected by the score sequence.

In addition, we also perform a comparison with some state-of-the-art tree-based systems that are publicly available, including an SMT system (Mi et al., 2008) and the NMT systems (Eriguchi et al. (2016)[9], Chen et al. (2017)[10], and Li et al. (2017)). For Mi et al. (2008), we use the implementation of `cicada`[11]. For Li et al. (2017), we reimplemented the "Mixed RNN Encoder" model, because of its outstanding performance on the NIST MT corpus.

We can see that for both English-Chinese and English-Japanese, compared with the s2s baseline system, both the 1-best and forest-based configurations yield better results. This indicates syntactic information contained in the constituent trees or forests is indeed useful for machine translation. Specifically, we observe the following facts.

First, among the three different frameworks SoE, SoA, and No-score, the SoA framework performs the best, while the No-score framework per-

[9] https://github.com/tempra28/tree2seq

[10] https://github.com/howardchenhd/Syntax-awared-NMT

[11] https://github.com/tarowatanabe/cicada

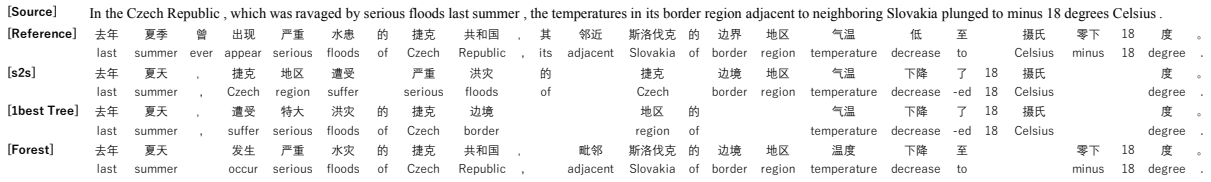


Figure 4: Chinese translation results of an English sentence.

forms the worst. This indicates that the scores of the edges in constituent trees or packed forests, which reflect the confidence of the correctness of the edges, are indeed useful. In fact, for the 1-best constituent parsing tree, the score of the edge reflects the confidence of the parser. By using this information, the NMT system succeed to learn a better attention, paying much attention to the confident structure and not paying attention to the unconfident structure, which improved the translation performance. This fact is ignored by previous studies on tree-based NMT. Furthermore, it is better to use the scores to modify the values of attention instead of rescaling the word embeddings, because modifying word embeddings carelessly may change the semantic meanings of words.

Second, compared with the cases that only using the 1-best constituent trees, using packed forests yields statistical significantly better results for the SoE and SoA frameworks. This shows the effectiveness of using more syntactic information. Compared with one constituent tree, the packed forest, which contains multiple different trees, describes the syntactic structure of the sentence in different aspects, which together increase the accuracy of machine translation. However, without using the scores, the 1-best constituent tree is preferred. This is because without using the scores, all trees in the packed forest are treated equally, which makes it easy to import noise into the encoder.

Compared with other types of state-of-the-art systems, our systems using only the 1-best tree (1-best(SoE, SoA)) are better than the other tree-based systems. Moreover, our NMT systems using the packed forests achieve the best performance. These results also support the usefulness of the scores of the edges and packed forests in NMT.

As for the efficiency, the training time of the SoA system was slightly longer than that of the SoE system, which was about twice of the s2s baseline. The training time of the tree-based system was about 1.5 times of the baseline. For the case of Forest (SoA), with 1 core of Tesla P100 GPU and LDC corpus as the training data, training spent about 10 days, and decoding speed was about 10 sentences per second. The reason for the relatively low efficiency is that the linearized sequences of packed forests were much longer than word sequences, enlarging the scale of the inputs. Despite this, the training process ended within reasonable time.

4.3 Qualitative analysis

Figure 4 illustrates the translation results of an English sentence using several different configurations: the s2s baseline, using only the 1-best tree (SoE), and using the packed forest (SoE). This is a sentence from NIST MT 03, and the training corpus is the LDC corpus.

For the s2s case, no syntactic information is utilized, and therefore the output of the system is not a grammatical Chinese sentence. The attributive phrase of "Czech border region" is a complete sentence. However, the attributive is not allowed to be a complete sentence in Chinese.

For the case of using 1-best constituent tree, the output is a grammatical Chinese sentence. However, the phrase "adjacent to neighboring Slovakia" is completely ignored in the translation result. After analyzing the constituent tree, we found that this phrase was incorrectly parsed as an "adverb phrase", so that the NMT system paid little attention to it, because of the low confidence given by the parser.

In contrast, for the case of the packed forest, we can see this phrase was not ignored and was translated correctly. Actually, besides "adverb phrase", this phrase was also correctly parsed as an "adjective phrase", and covered by multiple different nodes in the forest, making it difficult for the encoder to ignore the phrase.

We also noticed that our method performed better on learning attention. For the example in Figure 4, we observed that for s2s model, the decoder paid attention to the word "Czech" twice, which

causes the output sentence contains the Chinese translation of Czech twice. On the other hand, for our forest model, by using the syntax information, the decoder paid attention to the phrase "In the Czech Republic" only once, making the decoder generates the correct output.

5 Related work

Incorporating syntactic information into NMT systems is attracting widespread attention nowadays. Compared with conventional string-to-string NMT systems, tree-based systems demonstrate a better performance with the help of constituent trees or dependency trees.

The first noteworthy study is Eriguchi et al. (2016), which used Tree-structured LSTM (Tai et al., 2015) to encode the HPSG syntax tree of the sentence in the source-side in a bottom-up manner. Then, Chen et al. (2017) enhanced the encoder with a top-down tree encoder.

As a simple extension of Eriguchi et al. (2016), very recently, Zaremoodi and Haffari (2017) proposed a forest-based NMT method by representing the packed forest with a forest-structured neural network. However, their method was evaluated in small-scale MT settings (each training dataset consists of under 10k parallel sentences). In contrast, our proposed method is effective in a large-scale MT setting, and we present qualitative analysis regarding the effectiveness of using forests in NMT.

Although these methods obtained good results, the tree-structured network used by the encoder made the training and decoding relatively slow, therefore restricts the scope of application.

Other attempts at encoding syntactic trees have also been proposed. Eriguchi et al. (2017) combined the Recurrent Neural Network Grammar (Dyer et al., 2016) with NMT systems, while Li et al. (2017) linearized the constituent tree and encoded it using RNNs. The training of these methods is fast, because of the linear structures of RNNs. However, all these syntax-based NMT systems used only the 1-best parsing tree, making the systems sensitive to parsing errors.

Instead of using trees to represent syntactic information, some studies use other data structures to represent the latent syntax of the input sentence. For example, Hashimoto and Tsuruoka (2017) proposed translating using a latent graph. However, such systems do not enjoy the benefit of handcrafted syntactic knowledge, because they do not use a parser trained from a large treebank with human annotations.

Compared with these related studies, our framework utilizes a linearized packed forest, meaning the encoder can encode exponentially many trees in an efficient manner. The experimental results demonstrated these advantages.

6 Conclusion and future work

We proposed a new NMT framework, which encodes a packed forest for the source sentence using linear-structured neural networks, such as RNN. Compared with conventional string-to-string NMT systems and tree-to-string NMT systems, our framework can utilize exponentially many linearized parsing trees during encoding, without significantly decreasing the efficiency. This represents the first attempt at using a forest under the string-to-string NMT framework. The experimental results demonstrate the effectiveness of our framework.

As future work, we plan to design some more elaborate structures to incorporate the score layer in the encoder. Further improvement in the translation performance is expected to be achieved for the forest-based NMT system. We will also apply the proposed linearization method to other tasks.

Acknowledgements

We are grateful to the anonymous reviewers for their insightful comments and suggestions. We thank Lemao Liu from Tencent AI Lab for his suggestions about the experiments. We thank Atsushi Fujita whose suggestions greatly improve the readability and the logical soundness of this paper. This work was done during the internship of Chunpeng Ma at NICT. Akihiro Tamura is supported by JSPS KAKENHI Grant Number JP18K18110. Tiejun Zhao is supported by the National Natural Science Foundation of China (NSFC) via grant 91520204 and State High-Tech Development Plan of China (863 program) via grant 2015AA015405.

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 132–140.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1936–1945.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. *arXiv preprint arXiv:1602.07776*.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 823–833.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 72–78.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2017. Neural machine translation with source-side latent graph parsing. *arXiv preprint arXiv:1702.02265*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594.

Arthur B Kahn. 1962. Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*, pages 388–395.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 688–697.

Chunpeng Ma, Lemao Liu, Akihiro Tamura, Tiejun Zhao, and Sumita Eiichiro. 2017. Deterministic attention for sequence-to-sequence constituent parsing. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3237–3243.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 206–214.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL-08: HLT*, pages 192–199.

Maria Nadejde, Siva Reddy, Rico Sennrich, Tomasz Dwojak, Marcin Junczys-Dowmunt, Philipp Koehn, and Alexandra Brich. 2017. Syntax-aware neural machine translation using ccg. *arXiv preprint arXiv:1702.01147*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Chris Quirk and Simon Corston-Oliver. 2006. The impact of parse quality on syntactically-informed statistical machine translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 62–69.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. *arXiv preprint arXiv:1606.02892*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations

from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566.

Robert Endre Tarjan. 1976. Edge-disjoint spanning trees and depth-first search. *Acta Informatica*, 6(2):171–185.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 698–707.

Poorya Zaremoodi and Gholamreza Haffari. 2017. Incorporating syntactic uncertainty in neural machine translation with forest-to-sequence model. *arXiv preprint arXiv:1711.07019*.

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

Jiacheng Zhang, Yang Liu, Huanbo Luan, Jingfang Xu, and Maosong Sun. 2017. Prior knowledge integration for neural machine translation using posterior regularization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1514–1523.