Improved Neural Machine Translation with a Syntax-Aware Encoder and Decoder

Huadong Chen[†], Shujian Huang[†][*], David Chiang[‡], Jiajun Chen[†]

[†]State Key Laboratory for Novel Software Technology, Nanjing University
{chenhd,huangsj,chenjj}@nlp.nju.edu.cn
[‡]Department of Computer Science and Engineering, University of Notre Dame
dchiang@nd.edu

Abstract

Most neural machine translation (NMT) models are based on the sequential encoder-decoder framework, which makes no use of syntactic information. In this paper, we improve this model by explicitly incorporating source-side syntactic trees. More specifically, we propose (1) a *bidirectional tree encoder* which learns both sequential and tree structured representations; (2) a *tree-coverage model* that lets the attention depend on the source-side syntax. Experiments on Chinese-English translation demonstrate that our proposed models outperform the sequential attentional model as well as a stronger baseline with a bottom-up tree encoder and word coverage.[1]

1 Introduction

Recently, neural machine translation (NMT) models (Sutskever et al., 2014; Bahdanau et al., 2015) have obtained state-of-the-art performance on many language pairs. Their success depends on the representation they use to bridge the source and target language sentences. However, this representation, a sequence of fixed-dimensional vectors, differs considerably from most theories about mental representations of sentences, and from traditional natural language processing pipelines, in which semantics is built up compositionally using a recursive syntactic structure.
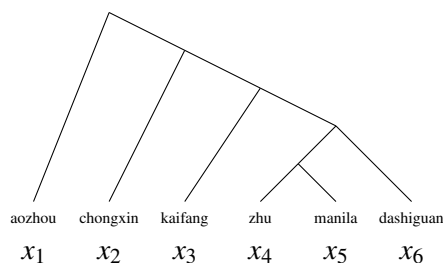
Perhaps as evidence of this, current NMT models still suffer from syntactic errors such as attachment (Shi et al., 2016). We argue that instead of letting the NMT model rely solely on the implicit structure it learns during training (Cho et al.,

(a) example sentence pair with alignments

(b) binarized source side tree

Figure 1: An example sentence pair (a), with its binarized source side tree (b). We use x_i to represent the i-th word in the source sentence. We will use this sentence pairs as the running example throughout this paper.

2014a), we can improve its performance by augmenting it with explicit structural information and using this information throughout the model. This has two benefits.

First, the explicit syntactic information will help the encoder generate better source side representations. Li et al. (2015) show that for tasks in which long-distance semantic dependencies matter, representations learned from recursive models using syntactic structures may be more powerful than those from sequential recurrent models. In the NMT case, given syntactic information, it will be easier for the encoder to incorporate long distance dependencies into better representations, which is especially important for the translation of long sentences.

Second, it becomes possible for the decoder to

---

use syntactic information to guide its reordering decisions better (especially for language pairs with significant reordering, like Chinese-English). Although the attention model (Bahdanau et al., 2015) and the coverage model (Tu et al., 2016; Mi et al., 2016) provide effective mechanisms to control the generation of translation, these mechanisms work at the word level and cannot capture phrasal cohesion between the two languages (Fox, 2002; Kim et al., 2017). With explicit syntactic structure, the decoder can generate the translation more in line with the source syntactic structure. For example, when translating the phrase *zhu manila dashiguan* in Figure 1, the tree structure indicates that *zhu* 'in' and *manila* form a syntactic unit, so that the model can avoid breaking this unit up to make an incorrect translation like "in embassy of manila" [2].

In this paper, we propose a novel encoder-decoder model that makes use of a precomputed source-side syntactic tree in both the encoder and decoder. In the encoder (§3.3), we improve the tree encoder of Eriguchi et al. (2016) by introducing a *bidirectional tree encoder*. For each source tree node (including the source words), we generate a representation containing information both from below (as with the original bottom-up encoder) and from above (using a top-down encoder). Thus, the annotation of each node summarizes the surrounding sequential context, as well as the entire syntactic context.

In the decoder (§3.4), we incorporate source syntactic tree structure into the attention model via an extension of the coverage model of Tu et al. (2016). With this *tree-coverage model*, we can better guide the generation phase of translation, for example, to learn a preference for phrasal cohesion (Fox, 2002). Moreover, with a tree encoder, the decoder may try to translate both a parent and a child node, even though they overlap; the tree-coverage model enables the decoder to learn to avoid this problem.

To demonstrate the effectiveness of the proposed model, we carry out experiments on Chinese-English translation. Our experiments show that: (1) our bidirectional tree encoder based NMT system achieves significant improvements over the standard attention-based NMT system, and (2) incorporating source tree structure into the attention model yields a further improvement.

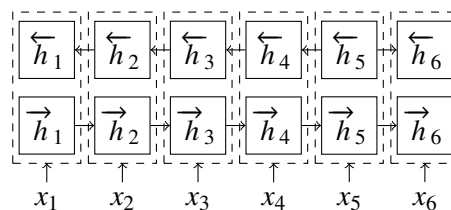[2]According to the source sentence, "embassy" belongs to "australia", not "manila".

Figure 2: Illustration of the bidirectional sequential encoder. The dashed rectangle represents the annotation of word x_i.

In all, we demonstrate an improvement of +3.54 BLEU over a standard attentional NMT system, and +1.90 BLEU over a stronger NMT system with a Tree-LSTM encoder (Eriguchi et al., 2016) and a coverage model (Tu et al., 2016). To the best of our knowledge, this is the first work that uses source-side syntax in both the encoder and decoder of an NMT system.

2 Neural Machine Translation

Most NMT systems follow the encoder-decoder framework with attention, first proposed by Bahdanau et al. (2015). Given a source sentence $\mathbf{x} = x_1 \cdots x_i \cdots x_I$ and a target sentence $\mathbf{y} = y_1 \cdots y_j \cdots y_J$, NMT aims to directly model the translation probability:

$$P(\mathbf{y} \mid \mathbf{x}; \theta) = \prod_1^J P(y_j \mid \mathbf{y}_{<\mathbf{j}}, \mathbf{x}; \theta), \qquad (1)$$

where θ is a set of parameters and $y_{<j}$ is the sequence of previously generated target words. Here, we briefly describe the underlying framework of the encoder-decoder NMT system.

2.1 Encoder Model

Following Bahdanau et al. (2015), we use a bidirectional gated recurrent unit (GRU) (Cho et al., 2014b) to encode the source sentence, so that the annotation of each word contains a summary of both the preceding and following words. The bidirectional GRU consists of a forward and a backward GRU, as shown in Figure 2. The forward GRU reads the source sentence from left to right and calculates a sequence of forward hidden states $(\overrightarrow{h_1}, \ldots, \overrightarrow{h_I})$. The backward GRU scans the source sentence from right to left, resulting in a sequence of backward hidden states $(\overleftarrow{h_1}, \ldots, \overleftarrow{h_I})$. Thus

$$\overrightarrow{h_i} = \text{GRU}(\overrightarrow{h_{i-1}}, s_i)$$
$$\overleftarrow{h_i} = \text{GRU}(\overleftarrow{h_{i-1}}, s_i) \qquad (2)$$

1937

where s_i is the i-th source word's word embedding, and GRU is a gated recurrent unit; see the paper by Cho et al. (2014b) for a definition.

The *annotation* of each source word x_i is obtained by concatenating the forward and backward hidden states:

$$\overleftrightarrow{h_i} = \begin{bmatrix} \overrightarrow{h_i} \\ \overleftarrow{h_i} \end{bmatrix}.$$

The whole sequence of these annotations is used by the decoder.

2.2 Decoder Model

The decoder is a forward GRU predicting the translation **y** word by word. The probability of generating the j-th word y_j is:

$$P(y_j \mid \mathbf{y}_{<\mathbf{j}}, \mathbf{x}; \theta) = \text{softmax}(t_{j-1}, d_j, c_j) \quad (3)$$

where t_{j-1} is the word embedding of the $(j-1)$-th target word, d_j is the decoder's hidden state of time j, and c_j is the *context vector* at time j. The state d_j is computed as

$$d_j = \text{GRU}(d_{j-1}, t_{j-1}, c_j), \quad (4)$$

where GRU($\cdot$) is extended to more than two arguments by first concatenating all arguments except the first.

The attention mechanism computes the context vector c_i as a weighted sum of the source annotations,

$$c_j = \sum_{i=1}^{I} \alpha_{j,i} \overleftrightarrow{h_i} \quad (5)$$

where the attention weight $\alpha_{j,i}$ is

$$\alpha_{j,i} = \frac{\exp(e_{j,i})}{\sum_{i'=1}^{I} \exp(e_{j,i'})} \quad (6)$$

and

$$e_{j,i} = v_a^{\text{T}} \tanh(W_a d_{j-1} + U_a \overleftrightarrow{h_i}) \quad (7)$$

where v_a, W_a and U_a are the weight matrices of the attention model, and $e_{j,i}$ is an attention model that scores how well d_{j-1} and $\overleftrightarrow{h_i}$ match.

With this strategy, the decoder can attend to the source annotations that are most relevant at a given time.

3 Tree Structure Enhanced Neural Machine Translation

Although syntax has shown its effectiveness in non-neural statistical machine translation (SMT) systems (Yamada and Knight, 2001; Koehn et al., 2003; Liu et al., 2006; Chiang, 2007), most proposed NMT models (a notable exception being that of Eriguchi et al. (2016)) process a sentence only as a sequence of words, and do not explicitly exploit the inherent structure of natural language sentences. In this section, we present models which directly incorporate source syntactic trees into the encoder-decoder framework.

3.1 Preliminaries

Like Eriguchi et al. (2016), we currently focus on source side syntactic trees, which can be computed prior to translation. Whereas Eriguchi et al. (2016) use HPSG trees, we use phrase-structure trees as in the Penn Chinese Treebank (Xue et al., 2005). Currently, we are only using the structure information from the tree without the syntactic labels. Thus our approach should be applicable to any syntactic grammar that provides such a tree structure (Figure 1(b)).

More formally, the encoder is given a source sentence $\mathbf{x} = x_1 \cdots x_I$ as well as a source tree whose leaves are labeled $x_1, \ldots, x_I$. We assume that this tree is strictly binary branching. For convenience, each node is assigned an index. The leaf nodes get indices $1, \ldots, I$, which is the same as their word indices. For any node with index k, let $p(k)$ denote the index of the node's parent (if it exists), and $L(k)$ and $R(k)$ denote the indices of the node's left and right children (if they exist).

3.2 Tree-GRU Encoder

We first describe tree encoders (Tai et al., 2015; Eriguchi et al., 2016), and then discuss our improvements.

Following Eriguchi et al. (2016), we build a tree encoder on top of the sequential encoder (as shown in Figure 3(a)). If node k is a leaf node, its hidden state is the annotation produced by the sequential encoder:

$$h_k^{\uparrow} = \overleftrightarrow{h_k}.$$

Thus, the encoder is able to capture both sequential context and syntactic context.

If node k is an interior node, its hidden state is the combination of its previously calculated left

child hidden state $h_{L(k)}$ and right child hidden state $h_{R(k)}$:

$$h_k^\uparrow = f(h_{L(k)}^\uparrow, h_{R(k)}^\uparrow) \qquad (8)$$

where $f(\cdot)$ is a nonlinear function, originally a Tree-LSTM (Tai et al., 2015; Eriguchi et al., 2016).

The first improvement we make to the above tree encoder is that, to be consistent with the sequential encoder model, we use Tree-GRU units instead of Tree-LSTM units. Similar to Tree-LSTMs, the Tree-GRU has gating mechanisms to control the information flow inside the unit for every node without separate memory cells. Then, Eq. 8 is calculated by a Tree-GRU as follows:

$$r_L = \sigma(U_L^{(rL)} h_{L(k)}^\uparrow + U_R^{(rL)} h_{R(k)}^\uparrow + b^{(rL)})$$

$$r_R = \sigma(U_L^{(rR)} h_{L(k)}^\uparrow + U_R^{(rR)} h_{R(k)}^\uparrow + b^{(rR)})$$

$$z_L = \sigma(U_L^{(zL)} h_{L(k)}^\uparrow + U_R^{(zL)} h_{R(k)}^\uparrow + b^{(zL)})$$

$$z_R = \sigma(U_L^{(zR)} h_{L(k)}^\uparrow + U_R^{(zR)} h_{R(k)}^\uparrow + b^{(zR)})$$

$$z = \sigma(U_L^{(z)} h_{L(k)}^\uparrow + U_R^{(z)} h_{R(k)}^\uparrow + b^{(z)})$$

$$\tilde{h}_k^\uparrow = \tanh\left(U_L(r_L \odot h_{L(k)}^\uparrow) + U_R(r_R \odot h_{R(k)}^\uparrow)\right)$$

$$h_k^\uparrow = z_L \odot h_{L(k)}^\uparrow + z_R \odot h_{R(k)}^\uparrow + z \odot \tilde{h}_k^\uparrow$$

where r_L, r_R are the reset gates and z_L, z_R are the update gates for the left and right children, and z is the update gate for the internal hidden state $\tilde{h}_k^\uparrow$. The $U^{(\cdot)}$ and $b^{(\cdot)}$ are the weight matrices and bias vectors.
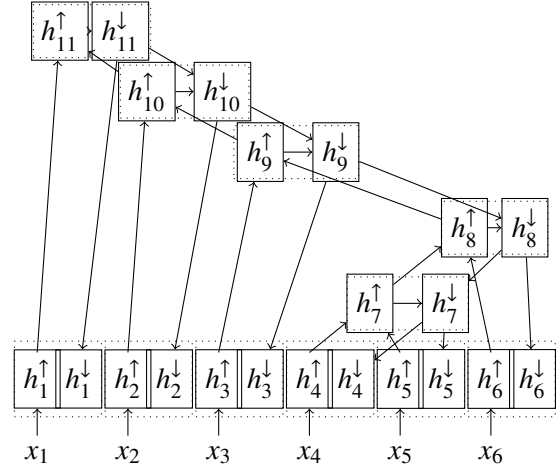
3.3 Bidirectional Tree Encoder

Although the bottom-up tree encoder can take advantage of syntactic structure, the learned representation of a node is based on its subtree only; it contains no information from higher up in the tree. In particular, the representation of leaf nodes is still the sequential one. Thus no syntactic information is fed into words. By analogy with the bidirectional sequential encoder, we propose a natural extension of the bottom-up tree encoder: the bidirectional tree encoder (Figure 3(b)).

Unlike the bottom-up tree encoder or the right-to-left sequential encoder, the top-down encoder by itself would have no lexical information as input. To address this issue, we feed the hidden states of the bottom-up encoder to the top-down encoder. In this way, the information of the whole syntactic tree is handed to the root node and propagated to its offspring by the top-down encoder.
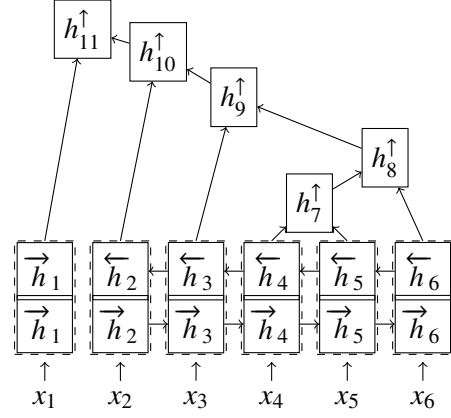
(a) Tree-GRU Encoder

(b) Bidirectional Tree Encoder

Figure 3: Illustration of the proposed encoder models for the running example. The non-leaf nodes are assigned with index 7-11. The annotations $h_i^\uparrow$ of leaf nodes in (b) are identical to the annotations (dashed rectangles) of leaf nodes in (a). The dotted rectangles in (b) indicate the annotation produced by the bidirectional tree encoder.

In the top-down encoder, each hidden state has only one predecessor. In fact, the top-down path from root of a tree to any node can be viewed as a sequential recurrent neural network. We can calculate the hidden states of each node top-down using a standard sequential GRU.

First, the hidden state of the root node ρ is simply computed as follows:

$$h_\rho^\downarrow = \tanh(W h_\rho^\uparrow + b) \qquad (9)$$

where W and b are a weight matrix and bias vector.

Then, other nodes are calculated by a GRU. For hidden state $h_k^\downarrow$:

$$h_k^\downarrow = \text{GRU}(h_{p(k)}^\downarrow, h_k^\uparrow) \qquad (10)$$

where $p(k)$ is the parent index of k. We replace the weight matrices W^r, U^r, W^z, U^z, W and U in the standard GRU with P_D^r, Q_D^r, P_D^z, Q_D^z, P_D, and Q_D, respectively. The subscript D is either L or R depending on whether node k is a left or right child, respectively.

Finally, the annotation of each node is obtained by concatenating its bottom-up hidden state and top-down hidden state:

$$h_k^{\updownarrow} = \begin{bmatrix} h_k^{\uparrow} \\ h_k^{\downarrow} \end{bmatrix}.$$

This allows the tree structure information flow from the root to the leaves (words). Thus, all the annotations are based on the full context of word sequence and syntactic tree structure.

Kokkinos and Potamianos (2017) propose a similar bidirectional Tree-GRU for sentiment analysis, which differs from ours in several respects: in the bottom-up encoder, we use separate reset/update gates for left and right children, analogous to Tree-LSTMs (Tai et al., 2015); in the top-down encoder, we use separate weights for left and right children.

Teng and Zhang (2016) also propose a bidirectional Tree-LSTM encoder for classification tasks. They use a more complex head-lexicalization scheme to feed the top-down encoder. We will compare their model with ours in the experiments.
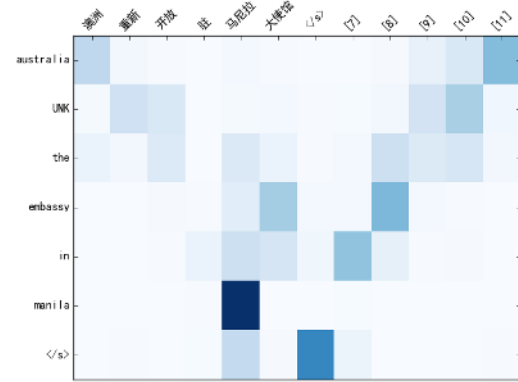
3.4 Tree-Coverage Model

We also extend the decoder to incorporate information about the source syntax into the attention model. We have observed two issues in translations produced using the tree encoder. First, a syntactic phrase in the source sentence is often incorrectly translated into discontinuous words in the output. Second, since the non-leaf node annotations contain more information than the leaf node annotations, the attention model prefers to attend to the non-leaf nodes, which may aggravate the over-translation problem (translating the same part of the sentence more than once).

As shown in Figure 4(a), almost all the non-leaf nodes are attended too many times during decoding. As a result, the Chinese phrase *zhu manila* is translated twice because the model attends to the node spanning *zhu manila* even though both words have already been translated; there is no mechanism to prevent this.
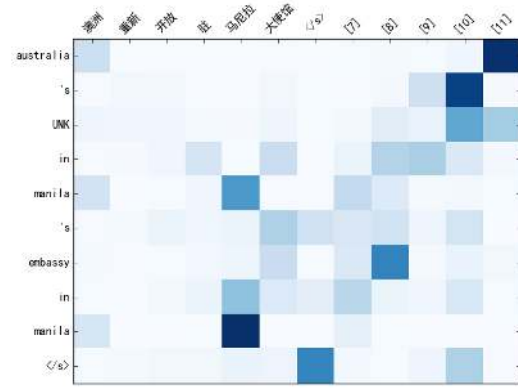
(a) Tree-GRU Encoder

(b) + Tree-Coverage Model

Figure 4: The attention heapmap plotting the attention weights during different translation steps, for translating the sentence in Figure 1(a). The nodes [7]-[11] correspond to non-leaf nodes indexed in Figure 3. Incorporating Tree-Coverage Model produces more concentrated alignments and alleviates the over-translation problem.

Inspired by the approaches of Cohn et al. (2016), Feng et al. (2016), Tu et al. (2016) and Mi et al. (2016), we propose to use prior knowledge to control the attention mechanism. In our case, the prior knowledge is the source syntactic information.

In particular, we build our model on top of the word coverage model proposed by Tu et al. (2016), which alleviate the problems of over-translation and under-translation (failing to translate part of a sentence). The word coverage model makes the attention at a given time step j dependent on the attention at previous time steps via *coverage vectors*:

$$C_{j,i} = \text{GRU}(C_{j-1,i}, \alpha_{j,i}, d_{j-1}, h_i). \quad (11)$$

1940

The coverage vectors are, in turn, used to update the attention at the next time step, by a small modification to the calculation of $e_{j,i}$ in Eq. (7):

$$e_{j,i} = v_a^{\mathrm{T}} \tanh\left(W_a d_{j-1} + U_a h_i + V_a C_{j-1,i}\right). \quad (12)$$

The word coverage model could be interpreted as a control mechanism for the attention model. Like the standard attention model, this coverage model sees the source-sentence annotations as a bag of vectors; it knows nothing about word order, still less about syntactic structure.

For our model, we extend the word coverage model to coverage on the tree structure by adding a coverage vector for each node in the tree. We further incorporate source tree structure information into the calculation of the coverage vector by requiring each node's coverage vector to depend on its children's coverage vectors and attentions at the previous time step:

$$\begin{aligned} C_{j,i} = \mathrm{GRU}(&C_{j-1,i}, \alpha_{j,i}, d_{j-1}, h_i, \\ &C_{j-1,L(i)}, \alpha_{j,L(i)}, \\ &C_{j-1,R(i)}, \alpha_{j,R(i)}). \end{aligned} \quad (13)$$

Although both child and parent nodes of a subtree are helpful for translation, they may supply redundant information. With our mechanism, when the child node is used to produce a translation, the coverage vector of its parent node will reflect this fact, so that the decoder may avoid using the redundant information in the parent node. Figure 4(b) shows a heatmap of the attention of our tree structure enhanced attention model. The attention of non-leaf nodes becomes more concentrated and the over-translation of *zhu manila* is corrected.

4 Experiments

4.1 Data

We conduct experiments on the NIST Chinese-English translation task. The parallel training data consists of 1.6M sentence pairs extracted from LDC corpora,[3] with 46.6M Chinese words and 52.5M English words, respectively. We use NIST MT02 as development data, and NIST MT03–06 as test data. These data are mostly in the same genre (newswire), avoiding the extra consideration of domain adaptation. Table 1 shows the statistics of the data sets. The Chinese side of the corpora is word segmented using ICTCLAS.[4] We

Data	Usage	Sents.
LDC	train	1.6M
MT02	dev	878
MT03	test	919
MT04	test	1,597
MT05	test	1,082
MT06	test	1,664

Table 1: Experiment data and statistics.

parse the Chinese sentences with the Berkeley Parser[5] (Petrov and Klein, 2007) and binarize the resulting trees following Zhang and Clark (2009). The English side of the corpora is lowercased and tokenized.

We filter out any translation pairs whose source sentences fail to be parsed. For efficient training, we also filter out the sentence pairs whose source or target lengths are longer than 50. We use a shortlist of the 30,000 most frequent words in each language to train our models, covering approximately 98.2% and 99.5% of the Chinese and English tokens, respectively. All out-of-vocabulary words are mapped to a special symbol UNK.

4.2 Model and Training Details

We compare our proposed models with several state-of-the-art NMT systems and techniques:

- **NMT**: the standard attentional NMT model (Bahdanau et al., 2015).

- **Tree-LSTM**: the attentional NMT model extended with the Tree-LSTM encoder (Eriguchi et al., 2016).

- **Coverage**: the attentional NMT model extended with word coverage (Tu et al., 2016).

We used the dl4mt implementation of the attentional model,[6] reimplementing the tree encoder and word coverage models. The word embedding dimension is 512. The hidden layer sizes of both forward and backward sequential encoder are 1024 (except where indicated). Since our Tree-GRU encoders are built on top of the bidirectional sequential encoder, the size of the hidden layer (in each direction) is 2048. For the coverage model, we set the size of coverage vectors to 50.

---

[3]LDC2002E18, LDC2003E14, the Hansards portion of LDC2004T08, and LDC2005T06.
[4]http://ictclas.nlpir.org

[5]https://github.com/slavpetrov/berkeleyparser
[6]https://github.com/nyu-dl/dl4mt-tutorial

#	Encoder	Coverage	MT02	MT03	MT04	MT05	MT06	Average
1	Sequential	no	33.76	31.88	33.15	30.55	27.47	30.76
2	Tree-LSTM	no	33.83	33.15	33.81	31.22	27.86	31.51(+0.75)
3	Tree-GRU	no	35.39	33.62	35.1	32.55	28.26	32.38(+1.62)
4	Bidirectional	no	35.52	33.91	35.51	33.34	29.91	33.17(+2.41)
5	Sequential	word	34.21	32.73	34.17	31.64	28.29	31.71(+0.95)
6	Tree-LSTM	word	35.81	33.62	34.84	32.6	28.52	32.40(+1.64)
7	Tree-GRU	word	35.91	33.71	35.46	33.02	29.14	32.84(+2.08)
8	Bidirectional	word	36.14	35.00	36.07	33.74	30.40	33.80(+3.04)
9	Tree-LSTM	tree	34.97	33.91	35.21	33.08	29.38	32.90(+2.14)
10	Tree-GRU	tree	35.67	34.25	35.72	33.47	29.95	33.35(+2.59)
11	Bidirectional	tree	**36.57**	**35.64**	**36.63**	**34.35**	**30.57**	**34.30(+3.54)**

Table 2: BLEU scores of different systems. "Sequential", "Tree-LSTM", "Tree-GRU" and "Bidirectional" denote the encoder part for the standard sequential encoder, Tree-LSTM encoder, Tree-GRU encoder and the bidirectional tree encoder, respectively. "no", "word" and "tree" in column "Coverage" represents the decoder part for using no coverage (standard attention), word coverage (Tu et al., 2016) and our proposed tree-coverage model, respectively.

#	System	Coverage	MT02	MT03	MT04	MT05	MT06	Average
12′	Seq-LSTM	no	34.98	32.81	34.08	31.39	28.03	31.58(+0.82)
13′	SeqTree-LSTM	no	35.28	33.56	34.94	32.64	29.26	32.60(+1.84)

Table 3: BLEU scores of different systems based on LSTM. "Seq-LSTM" denotes both the encoder and decoder parts for the sequential model are based on LSTM; "SeqTree-LSTM" means using Tree-LSTM encoder on top of "Seq-LSTM".

We use Adadelta (Zeiler, 2012) for optimization using a mini-batch size of 32. All other settings are the same as in Bahdanau et al. (2015).

We use case insensitive 4-gram BLEU (Papineni et al., 2002) for evaluation, as calculated by `multi-bleu.perl` in the Moses toolkit.[7]

4.3 Tree Encoders

This set of experiments evaluates the effectiveness of our proposed tree encoders. Table 2, row 2 confirms the finding of Eriguchi et al. (2016) that a Tree-LSTM encoder helps, and row 3 shows that our Tree-GRU encoder gets a better result (+0.87 BLEU, v.s. row 2). To verify our assumption that model consistency is important for performance, we also conduct experiments to compare Tree-LSTM and Tree-GRU on top of LSTM-based encoder-decoder settings. Tree-Lstm with LSTM based sequential model can obtain 1.02 BLEU improvement(Table 3, row 13′), while Tree-LSTM with GRU based sequential model only gets 0.75 BLEU improvement. Although Tree-Lstm with LSTM based sequential model obtain a slightly better result(+0.22 BLEU, v.s. Table 2, row 3), it

has more parameters(+1.6M) and takes 1.3 times longer for training.

Since the annotation size of our bidirectional tree encoder is twice of the Tree-LSTM encoder, we halved the size of the hidden layers in the sequential encoder to 512 in each direction, to make fair comparison. These results are shown in Table 4. Row 4′ shows that, even with the same annotation size, our bidirectional tree encoder works better than the original Tree-LSTM encoder (row 2). In fact, our halved-sized unidirectional Tree-GRU encoder (row 3′) also works better than the Tree-LSTM encoder (row 2) with half of its annotation size.

We also compared our bidirectional tree encoder with the head-lexicalization based bidirectional tree encoder proposed by Teng and Zhang (2016), which forms the input vector for each non-leaf node by a bottom-up head propagation mechanism (Table 4, row 14′). Our bidirectional tree encoder gives a better result, suggesting that head word information may not be as helpful for machine translation as it is for syntactic parsing.

When we set the hidden size back to 1024, we found that training the bidirectional tree encoder

---

[7] http://www.statmt.org/moses

#	Encoder	Coverage	MT02	MT03	MT04	MT05	MT06	Average
3′	Tree-GRU	no	34.92	32.79	34.16	32.03	28.75	31.93(+1.17)
4′	Bidirectional	no	35.02	32.64	35.04	32.50	29.72	32.48(+1.72)
14′	Bidirectional-head	no	34.66	33.17	34.78	31.70	28.47	32.03(+1.27)

Table 4: Experiments with 512 hidden units in each direction of the sequential encoder. The bidirectional tree encoder using head-lexicalization (Bidirectional-head), proposed by (Teng and Zhang, 2016), does not work as well as our simpler bidirectional tree encoder (Bidirectional).

was more difficult. Therefore, we adopted a two-phase training strategy: first, we train the parameters of the bottom-up encoder based NMT system; then, with the initialization of bottom-up encoder and random initialization of the top-down part and decoder, we train the bidirectional tree encoder based NMT system. Table 2, row 4 shows the results of this two-phase training: the bidirectional model (row 4) is 0.79 BLEU better than our unidirectional Tree-GRU (row 3).

4.4 Tree-Coverage Model

Rows 5–8 in Table 2 show that the word coverage model of Tu et al. (2016) consistently helps when used with our proposed tree encoders, with the bidirectional tree encoder remaining the best. However, the improvements of the tree encoder models are smaller than that of the baseline system. This may be caused by the fact that the word coverage model neglects the relationship among the trees, e.g. the relationship between children and parent nodes. Our tree-coverage model consistently improves performance further (rows 9–11).

Our best model combines our bidirectional tree encoder with our tree-coverage model (row 11), yielding a net improvement of +3.54 BLEU over the standard attentional model (row 1), and +1.90 BLEU over the stronger baseline that implements both the bottom-up tree encoder and coverage model from previous work (row 6).

As noted before, the original coverage model does not take word order into account. For comparison, we also implement an extension of the coverage model that lets each coverage vector also depend on those of its left and right neighbors at the previous time step. This model does not help; in fact, it reduces BLEU by about 0.2.

4.5 Analysis By Sentence Length

Following Bahdanau et al. (2015), we bin the development and test sentences by length and show BLEU scores for each bin in Figure 5. The proposed bidirectional tree encoder outperforms the
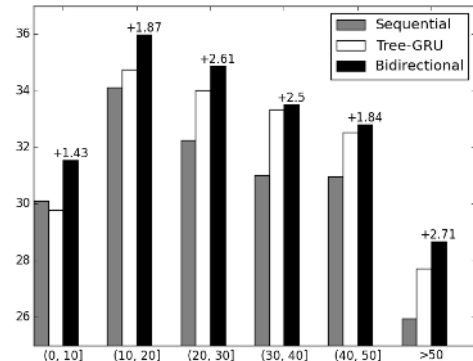

Figure 5: Performance of translations with respect to the lengths of the source sentences. "+" indicates the improvement over the baseline sequential model.

sequential NMT system and the Tree-GRU encoder across all lengths. The improvements become larger for sentences longer than 20 words, and the biggest improvement is for sentences longer than 50 words. This provides some evidence for the importance of syntactic information for long sentences.

5 Related Work

Recently, many studies have focused on using explicit syntactic tree structure to help learn sentence representations for various sentence classification tasks. For example, Teng and Zhang (2016) and Kokkinos and Potamianos (2017) extend the bottom-up model to a bidirectional model for classification tasks, using Tree-LSTMs with head lexicalization and Tree-GRUs, respectively. We draw on some of these ideas and apply them to machine translation. We use the representation learnt from tree structures to enhance the original sequential model, and make use of these syntactic information during the generation phase.

In NMT systems, the attention model (Bahdanau et al., 2015) becomes a crucial part of the

decoder model. Cohn et al. (2016) and Feng et al. (2016) extend the attentional model to include structural biases from word based alignment models. Kim et al. (2017) incorporate richer structural distributions within deep networks to extend the attention model. Our contribution to the decoder model is to directly exploit structural information in the attention model combined with a coverage mechanism.

6 Conclusion

We have investigated the potential of using explicit source-side syntactic trees in NMT by proposing a novel syntax-aware encoder-decoder model. Our experiments have demonstrated that a top-down encoder is a useful enhancement for the original bottom-up tree encoder (Eriguchi et al., 2016); and incorporating syntactic structure information into the decoder can better control the translation. Our analysis suggests that the benefit of source-side syntax is especially strong for long sentences.

Our current work only uses the structure part of the syntactic tree, without the labels. For future work, it will be interesting to make use of node labels from the tree, or to use syntactic information on the target side, as well.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*. http://arxiv.org/abs/1409.0473.

David Chiang. 2007. Hierarchical phrase-based translation. *Compututational Linguistics* 33(2):201–228. https://doi.org/10.1162/coli.2007.33.2.201.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proc. Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. pages 103–111. http://www.aclweb.org/anthology/W14-4012.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. EMNLP*. pages 1724–1734. http://www.aclweb.org/anthology/D14-1179.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proc. NAACL HLT*. pages 876–885. http://www.aclweb.org/anthology/N16-1102.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proc. ACL*. pages 823–833. http://www.aclweb.org/anthology/P16-1078.

Shi Feng, Shujie Liu, Nan Yang, Mu Li, Ming Zhou, and Kenny Q. Zhu. 2016. Improving attention modeling with implicit distortion and fertility for machine translation. In *Proc. COLING*. pages 3082–3092. http://aclweb.org/anthology/C16-1290.

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proc. EMNLP*. pages 304–3111. https://doi.org/10.3115/1118693.1118732.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. In *Proc. ICLR*. http://arxiv.org/abs/1702.00887.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. NAACL HLT*. pages 48–54. https://doi.org/10.3115/1073445.1073462.

Filippos Kokkinos and Alexandros Potamianos. 2017. Structural attention neural networks for improved sentiment analysis. In *Proc. EACL*. pages 586–591. http://www.aclweb.org/anthology/E17-2093.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proc. EMNLP*. pages 2304–2314. http://aclweb.org/anthology/D15-1278.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. ACL*. pages 609–616. https://doi.org/10.3115/1220175.1220252.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation. In *Proc. EMNLP*. pages 955–960. https://aclweb.org/anthology/D16-1096.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. ACL*. pages 311–318. https://doi.org/10.3115/1073083.1073135.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proc. NAACL HLT*. pages 404–411. http://www.aclweb.org/anthology/N/N07/N07-1051.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proc. EMNLP*. pages 1526–1534. https://aclweb.org/anthology/D16-1159.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proc. ACL-IJCNLP*. pages 1556–1566. http://www.aclweb.org/anthology/P15-1150.

Zhiyang Teng and Yue Zhang. 2016. Bidirectional tree-structured LSTM with head lexicalization. arXiv:1611.06788. http://arxiv.org/abs/1611.06788.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proc. ACL*. pages 76–85. http://www.aclweb.org/anthology/P16-1008.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Nat. Lang. Eng.* 11(2):207–238. https://doi.org/10.1017/S135132490400364X.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proc. ACL*. pages 523–530. https://doi.org/10.3115/1073012.1073079.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR* abs/1212.5701. http://arxiv.org/abs/1212.5701.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the Chinese Treebank using a global discriminative model. In *Proc. IWPT*. pages 162–171. http://www.aclweb.org/anthology/W09-3825.