Incorporating Syntactic Uncertainty in Neural Machine Translation with a Forest-to-Sequence Model

Poorya Zaremoodi **Gholamreza Haffari**
Faculty of Information Technology, Monash University
firstname.lastname@monash.edu

Abstract

Incorporating syntactic information in Neural Machine Translation (NMT) can lead to better reorderings, particularly useful when the language pairs are syntactically highly divergent or when the training bitext is not large. Previous work on using syntactic information, provided by top-1 parse trees generated by (inevitably error-prone) parsers, has been promising. In this paper, we propose a *forest*-to-sequence NMT model to make use of exponentially many parse trees of the source sentence to compensate for the parser errors. Our method represents the collection of parse trees as a packed forest, and learns a neural transducer to translate from the input forest to the target sentence. Experiments on English to German, Chinese and Farsi translation tasks show the superiority of our approach over the sequence-to-sequence and tree-to-sequence neural translation models.

1 Introduction

The neural approach is revolutionising machine translation (MT). The main neural approach to MT is based on the encoder-decoder architecture (Cho et al., 2014; Sutskever et al., 2014), where an encoder (e.g a recurrent neural network) reads the source sentences *sequentially* to produce a fixed-length vector representation. Then, a decoder generates the translation from the encoded vector, which can dynamically change using the *attention* mechanism.

One of the main premises about natural language is that words of a sentence are inter-related according to a (latent) hierarchical structure, i.e. a syntactic tree. Therefore, it is expected that modeling the syntactic structure should improve the performance of NMT, especially in low-resource or linguistically divergent scenarios, such as English-Farsi. In this direction, (Li et al., 2017) uses a sequence-to-sequence model, making use of *linearised* parse trees. (Chen et al., 2017b) has proposed a model which uses syntax to constrain the dynamic encoding of the source sentence via structurally constrained attention. (Bastings et al., 2017; Shuangzhi Wu, 2017) have incorporated syntactic information provided by the dependency tree of the source sentence. (Marcheggiani et al., 2018) has proposed a model to inject semantic bias into the encoder of NMT model.

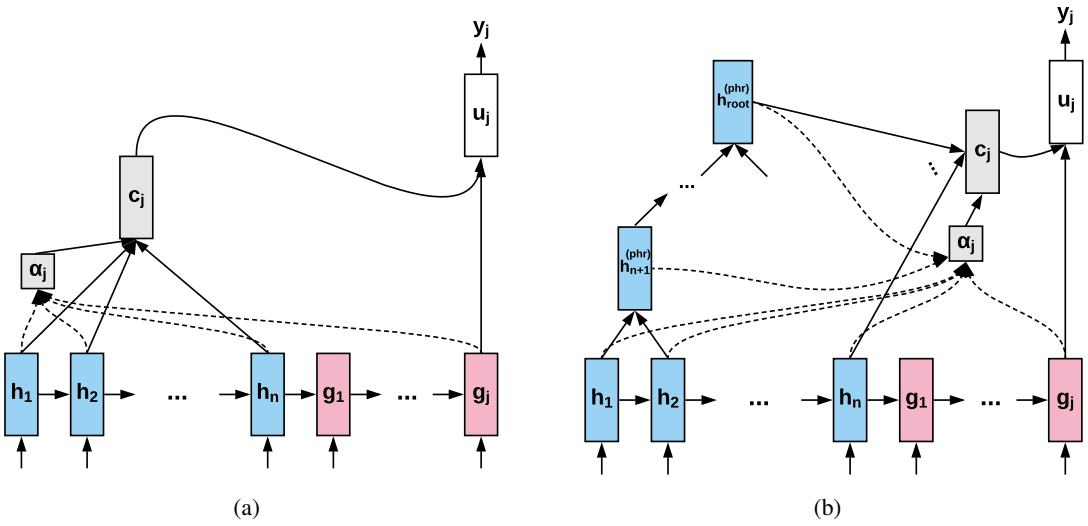
Recently, (Eriguchi et al., 2016; Chen et al., 2017a) have proposed methods to incorporate the hierarchical syntactic constituency information of the source sentence. In addition to the embedding of words, computed using the vanilla sequential encoder, they compute the embeddings of phrases recursively, directed by the top-1 parse tree of the source sentence generated by a parser. Though the results are promising, the top-1 trees are prone to parser error, and furthermore cannot capture semantic ambiguities of the source sentence.

In this paper, we address the aforementioned issues by using exponentially many trees encoded in a forest instead of a single top-1 parse tree. We capture the parser uncertainty by considering many parse trees and their probabilities. The encoding of each source sentence is guided by the forest, and includes the forest nodes whose representations are computed in a bottom-up fashion using our ForestLSTM architecture (§3). Thus, in the encoding stage of this approach, different ways of constructing a phrase

Proceedings of the 27th International Conference on Computational Linguistics, pages 1421–1429
Santa Fe, New Mexico, USA, August 20-26, 2018.

are taken into consideration along with the probability of rules in the corresponding trees. We evaluate our approach on English to Chinese, Farsi and German translation tasks, showing that forests lead to better performance compared to top-1 tree and sequential encoders (§4).

2 Neural Machine Translation



(a) (b)

Figure 1: (a) Attentional SEQ2SEQ model (b) Attentional TREE2SEQ model.

2.1 Attentional Sequence-to-Sequence model

This model is based on the attentional encoder-decoder architecture for SEQ2SEQ transduction. It contains a sequential encoder to read the input sentence and a sequential decoder to generate the output. An example is depicted in Figure 1(a).

Sequential Encoder The source sentence is encoded by a sequential LSTM:

$$\boldsymbol{h}_i = \text{SeqLSTM}(\boldsymbol{h}_{i-1}, \boldsymbol{E}_x[x_i])$$

where $\boldsymbol{E}_x[x_i]$ is the embedding of the word x_i in the embedding table $\boldsymbol{E}_x$ of the source language, and $\boldsymbol{h}_i$ is the context-dependent embedding of x_i.

Sequential Decoder For the generation of each target word j, a dynamic context $\boldsymbol{c}_j$ is produced to summarise the relevant parts of the source sentence. Then, the decoder generates the j-th word as follows:

$$\begin{aligned} \boldsymbol{g}_j &= \tanh(\boldsymbol{W}^{gh}\boldsymbol{g}_{j-1} + \boldsymbol{W}^{gi}\boldsymbol{E}_y[\boldsymbol{y}_{j-1}] + \boldsymbol{W}^{ga}\boldsymbol{c}_j + \boldsymbol{W}^{gu}\boldsymbol{u}_{j-1}) \\ \boldsymbol{u}_j &= \tanh(\boldsymbol{W}^{uc}\boldsymbol{c}_j + \boldsymbol{W}^{ui}\boldsymbol{E}_y[\boldsymbol{y}_{j-1}] + \boldsymbol{g_j}) \\ y_j|\boldsymbol{y}_{<j}, \boldsymbol{x} &\sim \text{softmax}(\boldsymbol{W}^{ou}\boldsymbol{u_j} + \boldsymbol{b}^o) \end{aligned}$$

where $\boldsymbol{E}_y[y_j]$ is the embedding of word y_j looked-up from target language embedding table $\boldsymbol{E}_y$, and $\boldsymbol{W}$ matrices and $\boldsymbol{b}^o$ are the parameters.

The attention mechanism provides relevant information from the source sentence with respect to the current state of the decoder. In each decoding step, the dynamic context is computed as follows:

$$\begin{aligned} a_{ij} &= \boldsymbol{v}^\top \tanh(\boldsymbol{W}^{ae}\boldsymbol{h}_i + \boldsymbol{W}^{ag}\boldsymbol{g}_{j-1}) \\ a_{i'j} &= \boldsymbol{v}^\top \tanh(\boldsymbol{W}^{ae}\boldsymbol{h}^{phr}_{i'} + \boldsymbol{W}^{ag}\boldsymbol{g}_{j-1}) \\ \boldsymbol{\alpha}_j &= \text{softmax}(\boldsymbol{a}_j) \end{aligned}$$

$$\boldsymbol{c}_j = \sum_{i=1}^{n} \alpha_{ji} \boldsymbol{h}_i$$

where a_{ij} and $a_{i'j}$ are the attention scores, and $\boldsymbol{h}_i$ refers to embedding of words. n is the length of the input sentence, and n_p is the number of forest nodes.

2.2 Attentional Tree-to-Sequence model

The majority of NMT models are based on sequential encoding of the source sentence. An exception is the TREE2SEQ model in (Eriguchi et al., 2016), where the encoding of the source sentence is directed by the top-1 parse tree. This model computes the embeddings of phrases in addition to the words, then attend on both words and phrases in the decoder. An example is depicted in Figure 1(b).

Tree Encoder It consists of sequential and recursive parts. The sequential part is the vanilla sequence encoder discussed in Section 2.1, which computes the embeddings of words. Then, the embeddings of phrases are computed using the embeddings of their constituent words in a recursive bottom-up fashion:

$$\boldsymbol{h}_k^{(phr)} = \text{TreeLSTM}(\boldsymbol{h}_k^l, \boldsymbol{h}_k^r).$$

where $\boldsymbol{h}_k^l$ and $\boldsymbol{h}_k^r$ are hidden states of left and right children respectively. This method uses TreeLSTM units (Tai et al., 2015) to calculate the embedding of a parent node using its two two children units as follow:

$$\begin{aligned}
\boldsymbol{i} &= \sigma(\boldsymbol{U}_l^{(i)} \boldsymbol{h}^l + \boldsymbol{U}_r^{(i)} \boldsymbol{h}^r + \boldsymbol{b}^{(i)}) \\
\boldsymbol{f}^l &= \sigma(\boldsymbol{U}_l^{(f_l)} \boldsymbol{h}^l + \boldsymbol{U}_r^{(f_l)} \boldsymbol{h}^r + \boldsymbol{b}^{(f_l)}) \\
\boldsymbol{f}^r &= \sigma(\boldsymbol{U}_l^{(f_r)} \boldsymbol{h}^r + \boldsymbol{U}_r^{(f_r)} \boldsymbol{h}^r + \boldsymbol{b}^{(f_r)}) \\
\boldsymbol{o} &= \sigma(\boldsymbol{U}_l^{(o)} \boldsymbol{h}^l + \boldsymbol{U}_r^{(o)} \boldsymbol{h}^r + \boldsymbol{b}^{(o)}) \\
\tilde{\boldsymbol{c}} &= \tanh(\boldsymbol{U}_l^{(\tilde{c})} \boldsymbol{h}^l + \boldsymbol{U}_r^{(\tilde{c})} \boldsymbol{h}^r + \boldsymbol{b}^{(\tilde{c})}) \\
\boldsymbol{c}^{(phr)} &= \boldsymbol{i} \odot \tilde{\boldsymbol{c}} + \boldsymbol{f}^l \odot \boldsymbol{c}^l + \boldsymbol{f}^r \odot \boldsymbol{c}^r \\
\boldsymbol{h}^{(phr)} &= \boldsymbol{o} \odot \tanh(\boldsymbol{c}^{(phr)})
\end{aligned}$$

where $\boldsymbol{i}$, $\boldsymbol{f}^l$, $\boldsymbol{f}^r$, $\boldsymbol{o}_j$,$\tilde{\boldsymbol{c}}_j$ are the input gate, left and right forget gates, output gate, and a state for updating memory cell; $\boldsymbol{c}^r$ and $\boldsymbol{c}^l$ are memory cells of the right and left units.

Sequential Decoder Eriguchi et al. set the initial state of the decoder by combining the final state of the sequential and tree encoders as follow:

$$\boldsymbol{g}_0 = \text{TreeLSTM}(\boldsymbol{h}_n, \boldsymbol{h}_{root}^{(phr)}),$$

The rest of the decoder is similar to the vanilla attentional decoder discussed in Section 2.1. The difference is that, in this model, the attention mechanism makes use of phrases as well as words. Thus, the dynamic context is computed as follows:

$$\boldsymbol{c}_j = \sum_{i=1}^{n} \alpha_{ji} \boldsymbol{h}_i + \sum_{i'=n+1}^{2n-1} \alpha_{ji'} \boldsymbol{h}_{i'}^{phr}$$

3 Neural Forest-to-Sequence Translation

The TREE2SEQ model uses the top-1 parse tree generated by a parser. Mistakes and uncertainty in parsing eventually affect the performance of the translation. To address these issues, we propose a method to consider exponentially many parse trees along with their corresponding probabilities. It consists of a *forest* encoder to encode a collection of packed parse trees, in order to reduce error propagation due to using only the top-1 parse tree. Our forest encoder computes representations for words and phrases of the source sentence with respect to its parse forest. A sequential decoder, then, generates output words one-by-one from left-to-right by attending to both words and phrases (i.e. forest nodes).

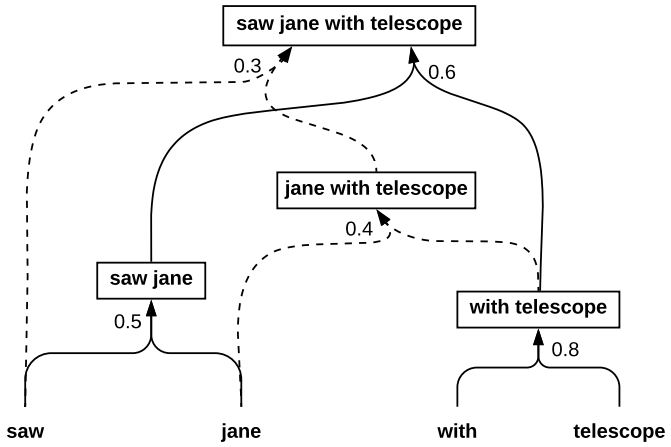


Figure 2: An example of generating a phrase from two different parse trees

3.1 Forest Encoder

The representation of words are computed using sequential encoder (Section 2.1). Then, the embeddings of phrases are computed with respect to the source sentence parse forest in a recursive bottom-up fashion.

Embedding of Phrases. We compute the embedding of the forest nodes (phrases) in a bottom-up fashion. For each hyperedge, we compute the embedding of the head with respect to its tails using a TreeLSTM unit (Tai et al., 2015). In a forest, however, a phrase can be constructed in multiple ways using the incoming hyperedges to a forest node, with different probabilities (see Figure 2). Our ForestLSTM combines the phrase embeddings resulted from these hyperedges, and takes into account their probabilities in order to obtain a unified embedding for the forest node and its corresponding phrase:

$$
\begin{aligned}
\boldsymbol{\gamma}^l &= \tanh\left(\boldsymbol{U}^{\gamma}\sum_{l'=1}^{N}\mathbf{1}_{l\neq l'}\boldsymbol{h}^{l'} + \boldsymbol{W}^{\gamma}\boldsymbol{h}^{l} + \boldsymbol{v}^{\gamma}p^{l} + \boldsymbol{b}^{\gamma}\right)\\
\boldsymbol{f}^l &= \sigma\left(\boldsymbol{U}^{f}\sum_{l'=1}^{N}\mathbf{1}_{l\neq l'}[\boldsymbol{h}^{l'};\boldsymbol{\gamma}^{l'}] + \boldsymbol{W}^{f}[\boldsymbol{h}^{l};\boldsymbol{\gamma}^{l}] + \boldsymbol{b}^{l}\right)\\
\boldsymbol{i} &= \sigma\left(\boldsymbol{U}^{i}\sum_{l=1}^{N}[\boldsymbol{h}^{l};\boldsymbol{\gamma}^{l}] + \boldsymbol{b}^{i}\right)\\
\boldsymbol{o} &= \sigma\left(\boldsymbol{U}^{o}\sum_{l=1}^{N}[\boldsymbol{h}^{l};\boldsymbol{\gamma}^{l}] + \boldsymbol{b}^{o}\right)
\end{aligned}
$$

where N is the number of incoming hyperedges, $\boldsymbol{h}^l$ is the embedding for the head of the l-th incoming hyperedge and p^l is its probability and $\boldsymbol{v}^{\gamma}$ is the learned weight for the probablity. $\boldsymbol{\gamma}^l$ is a probability-sensitive intermediate representation for the l-th incoming hyperedge, which is then used in the computations of the forget gate $\boldsymbol{f}^l$, the input gate $\boldsymbol{i}$, and the output gate $\boldsymbol{o}$. The representation of the phrase $\boldsymbol{h}^{phr}$ is then computed as

$$
\begin{aligned}
\tilde{\boldsymbol{c}} &= \tanh\left(\boldsymbol{U}^{\tilde{c}}\sum_{l=1}^{N}[\boldsymbol{h}^{l};\boldsymbol{\gamma}^{l}] + \boldsymbol{b}^{\tilde{c}}\right)\\
\boldsymbol{c}^{phr} &= \boldsymbol{i}\odot\tilde{\boldsymbol{c}} + \sum_{l=1}^{N}\boldsymbol{f}^{l}\odot\boldsymbol{c}^{l}\\
\boldsymbol{h}^{phr} &= \boldsymbol{o}\odot\tanh(\boldsymbol{c}^{phr})
\end{aligned}
$$

where $\boldsymbol{c}^l$ is the memory cell of the TreeLSTM unit used to compute the representation of the head for the l-th hyperedge from its tail nodes.

3.2 Sequential Decoder

We use a sequential attentional decoder similar to that of the TREE2SEQ model, where the attention mechanism attends to both words and phrases in the forest:

$$\boldsymbol{c}_j = \sum_{i=1}^{n} \alpha_{ji} \boldsymbol{h}_i + \sum_{i'=1+n}^{n_p+n} \alpha_{ji'} \boldsymbol{h}_{i'}^{phr}$$

where n is the length of the input sentence, and n_p is the number of forest nodes.

We initialize the decoder's first state by combining the embeddings of the last word in the source sentence and the root of the forest:

$$\boldsymbol{g}_0 = \text{TreeLSTM}(\boldsymbol{h}_n, \boldsymbol{h}_{root}^{phr}).$$

This provides a summary of phrases and words in the source sentence to the decoder.

3.3 Training

Training is done end-to-end by minimising the cross entropy objective:

$$J(\theta) = \sum_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{F}_{\boldsymbol{x}}) \in D} -\log p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{F}_{\boldsymbol{x}})$$

where D is the set of triples consists of the bilingual training sentences $(\boldsymbol{x}, \boldsymbol{y})$ paired with the parse forests of the source sentences $\boldsymbol{F}_{\boldsymbol{x}}$.

4 Computational Complexity Analysis

We now analyse the computational complexity of inference for SEQ2SEQ, TREE2SEQ and FOREST2SEQ models. We show that, interestingly, our method process exponentially many trees with only a small linear overhead.

Let $|x|$ denotes the length of the source sentence and $O(W_s)$ and $O(W_r)$ are computational complexity of forward-pass for the sequential and Tree/Forest LSTM units, respectively. Having N nodes in the tree/forest, the computational complexity of the encoding phase would be:

$$O(2W_s|x| + W_r N)$$

where the first term shows the computational complexity of a bidirectional sequential encoder to compute the embeddings of words, and the latter one is the time for computing the embeddings of phrases with respect to the corresponding tree/forest.

For generating each word in the target sentence, the attention mechanism performs soft attention on words and phrases of the source sentence. If $O(W_t)$ be the time for updating the decoder state and generating the next target word, for a target sentence with length $|y|$ the decoding phase computational complexity would be:

$$O(W_t|y| + |y|(N + |x|))$$

Hence, the total inference time for a sentence pair is:

$$O(2W_s|x| + W_r N + W_t|y| + |y|(N + x))$$

The difference among the three methods is N. For the SEQ2SEQ model N is 0. For the TREE2SEQ model the number of nodes in the tree is a constant function of the input size: $N = |x| - 1$. Since

Sentence Length	Avg. tree nodes	Avg. forest nodes	Avg. # of trees in forests
<10	7.94	9.77	6.13E+4
10-19	12.3	18.99	2.62E+16
20-29	21.18	41.79	2.76E+22
>30	31	78.72	2.21E+15
all	10.33	14.84	1.41E+20

Table 1: The average number of nodes in trees and forests along with average number of trees in forests for En→Fa bucketed dataset.

we used *pruned* forests obtained from the parser in (Huang, 2008), the number of nodes in the forest is variable. Table 1 shows the average value of N for trees/forests for different source lengths for one of the datasets we used in experiments. As seen, while forests contain exponentially many trees, on average, the number of nodes in parse forests is less than twice the number of nodes in the corresponding top-1 parse trees. It shows that our method considers exponentially many trees instead of top-1 tree using only a small linear overhead.

5 Experiments

5.1 The Setup

Datasets. We make use of three different language pairs: English (En) to Farsi (Fa), Chinese (Ch), and German (De). Our research focus is to tackle NMT issues for bilingually low-resource scenarios and En→Fa is intrinsically a low-resource language. Moreover, we used small datasets for En→Ch and En→De language pairs to simulate low-resource scenarios, where the source and target languages are linguistically divergent and close, respectively. For En→Fa, we use the TEP corpus (Tiedemann, 2009) which is extracted from movie subtitles. It has about 341K sentence pairs, where we split into 337K for training, 2K for development, and 2K for test. For En→Ch, we use BTEC where 'devset1_2' and 'devset_3' are used as the development and test sets, and training consists of 44,016 sentence pairs. For En→De, we use the first 100K sentences of Europarl[1] for training, 'newstest2013' for development, and 'newstest2014' for test.

We lowercase and tokenise the corpora using Moses scripts (Koehn et al., 2007). Sentences longer than 50 words are removed, and words with frequency less than 5 are replaced with <UNK>. Compact forests and trees for source English sentences are obtained from the parser in (Huang, 2008), where the forests are binarised, i.e. hyperedges with more than two tail nodes are converted to multiple hyperedges with two tail nodes. This is to ensure a fair comparison between our model and the TREE2SEQ model (Eriguchi et al., 2016) where they use binary HPSG parse trees. Furthermore, we prune the forests by removing low probability hyperedges, which significantly reduces the size of the forests. In all experiments, we use the development sets for setting the hyper-parameters, and the test sets for evaluation.

Implementation Details. We use Mantis implementation of attentional NMT (Cohn et al., 2016) to develop our code for FOREST2SEQ and TREE2SEQ with DyNet (Neubig et al., 2017). All neural models are trained end-to-end using Stochastic Gradient Descent, where the mini-batch size is set to 128. The maximum training epochs is set to 20, and we use early stopping on the development set as a stopping condition. We generate the translations using greedy decoding. The BLEU score is computed using 'multi-bleu.perl' script in Moses.

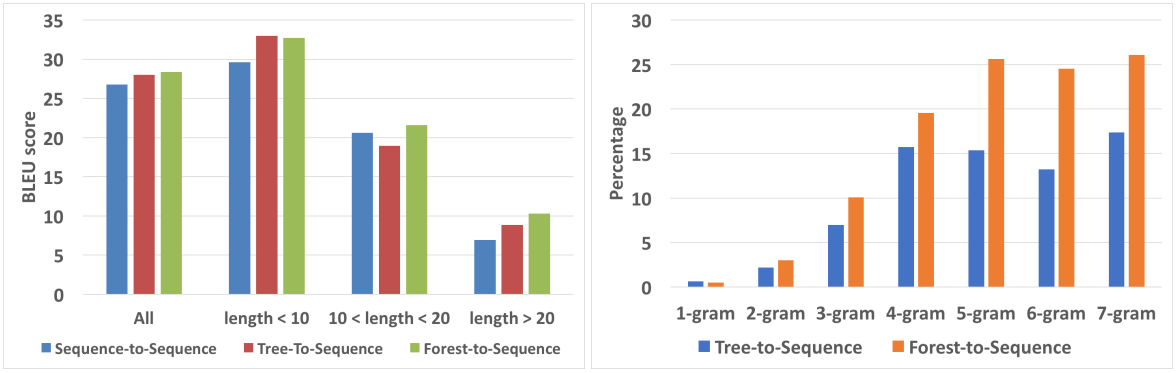
5.2 Results

The perplexity and BLEU scores of different models for all translation tasks are presented in Table 2. In all translation tasks, FOREST2SEQ outperforms TREE2SEQ as it reduces syntactic errors by using forests instead of top-1 parse trees. Our results confirm those in (Eriguchi et al., 2016), and show that using syntactic trees in TREE2SEQ improve the translation quality compared to the vanilla SEQ2SEQ.

[1] http://www.statmt.org/wmt14/translation-task.html

Method	H	En → De		En → Ch		En → Fa	
		Perplexity	BLEU	Perplexity	BLEU	Perplexity	BLEU
SEQ2SEQ	256	33.07	11.98	6.48	25.43	19.21	10.17
(Luong et al., 2015)	512	32.61	12.21	6.12	26.77	18.4	10.93
TREE2SEQ	256	30.13	13	6.17	26.85	17.94	11.32
(Eriguchi et al., 2016)	512	31.86	13.05	5.71	28	16.28	11.71
our	256	30.83	**13.54**	6.16	27.08	17.62	11.91
FOREST2SEQ	512	**29.25**	13.43	**5.49**	**28.39**	**16.66**	**12.38**

Table 2: Comparison of the methods together with different hidden dimension size (H) for all datasets.



(a) (b)

Figure 3: (a) BLEU scores for bucketed En→Ch dataset. (b) Percentage of more correct n-grams generated by the TREE2SEQ and FOREST2SEQ models compared to SEQ2SEQ model for En→Ch dataset.

Comparing BLEU scores of the forest-based and tree-based models, the largest increase is observed for En→Fa. This can be attributed to the syntactic divergence between English and Farsi (SVO vs SOV) as well as the reduction of significant errors in the top-1 parser trees for this translation task, resulted from the domain mismatch between the parser's training data (i.e. Penn Tree Bank) and the English source (i.e. informal movie subtitles).

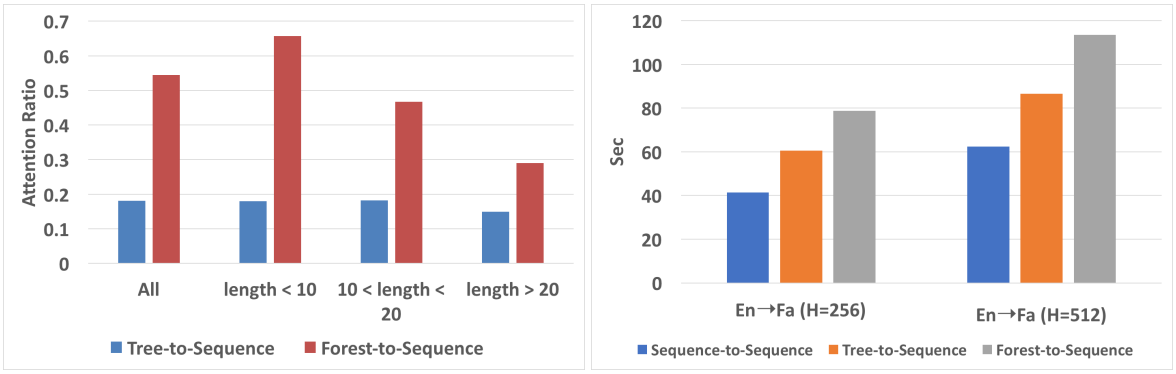
5.3 Analysis

The effect of sequential part in the forest encoder The forest encoder consists of sequential and recursive parts, where the former is the vanilla sequence encoder. We investigate the effect of the sequential part in the proposed forest encoder. Table 3 shows the results on the test set of En → Fa dataset. The results show that the sequential part in the forest encoder lead to improvement in results. Speculatively, the sequential part helps the forest encoder by providing the context-aware embeddings for words which then be used to construct phrase embeddings.

For which sentence lengths the forest-based model is more helpful? To investigate the effect of source sentence length, we divide En→Ch dataset into three buckets with respect to the length of source sentences. Figure 3(a) depicts the BLEU scores resulted from the models for different buckets. The FOREST2SEQ model performs better than vanilla SEQ2SEQ model on all buckets. Interestingly, while TREE2SEQ model has a lower BLEU compared to SEQ2SEQ model for the sentences whose lengths are between 10 and 20, the FOREST2SEQ model has a better BLEU possibly due to reducing parsing errors.

	Perplexity	BLEU
Forest encoder W/ sequential part	16.66	12.38
Forest encoder W/O sequential part	17.48	11.97

Table 3: The Effect of the sequential part in our proposed forest encoder on the En → Fa test set.



(a) (b)

Figure 4: (a) Attention ratios for En→Fa bucketed dataset.(b) Inference time (seconds) required for the test set of En→Fa dataset using trained models.

The forest-based model results in correct larger n-grams in translations We further analyse the effect of the incorporated syntactic knowledge on improving the number of generated gold n-grams in translations. For each sentence, we compute the number of n-grams in the generated translations which are common with those in the gold translation. Then, after aggregating the results over the entire test set, we compute the percentage of additional gold n-grams generated by syntax aware models, i.e. TREE2SEQ and FOREST2SEQ, compared to the SEQ2SEQ model. The results are depicted in Figure 3(b). Generating correct high order n-grams is hard, and results show that incorporating syntax is beneficial. As n increases, the FOREST2SEQ model performs significantly better than the TREE2SEQ model in generating gold n-grams, possibly due to better reorderings between the source and target.

How much attention the forest-based and tree-based models pay to the syntactic information? We next analyse the extent by which the syntactic information is used by the TREE2SEQ and FOREST2SEQ models. We compute the ratio of attention on phrases to words for both of the syntax-aware models in En→Fa translation task, where the source and target languages are highly syntactically divergent. For each triple in the test set, we calculate the sum of attention on words and phrases during decoding. Then, the ratio of attention on phrases to words are computed and is averaged for all triples. Figure 4(a) shows these attention ratios for bucketed En→Fa dataset. It shows that for all sentence lengths, the FOREST2SEQ model provides richer phrase embeddings compared to the TREE2SEQ model, leading to a more usage of the syntactic information.

Investigating the effect of using trees/forests on inference time We measured the inference time required for the test set of EN→FA dataset using the trained models. The results are depicted in Figure 4(b). As seen, while using one parse tree increases the inference time linearly, interestingly, our FOREST2SEQ model considers exponentially many trees also with a small linear overhead.

6 Conclusion

We have proposed a forest-to-sequence attentional NMT model, which uses a packed forest instead of the top-1 parse tree in the encoder.

Using a forest of parse trees, our method efficiently considers exponentially many constituency trees in order to take into account parser uncertainties and errors. Experimental results show our method is superior to the attentional tree-to-sequence model, which is more prone to the parsing errors.

Acknowledgments

This work was supported by the Multi-modal Australian ScienceS Imaging and Visualisation Environment (MASSIVE) (www.massive.org.au), and by the Australian Research Council through DP160102686. The first author was partly supported by CSIRO's Data61.

References

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. *arXiv preprint arXiv:1704.04675*.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017a. 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)*, volume 1, pages 1936–1945.

Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2017b. Syntax-directed attention for neural machine translation. *arXiv preprint arXiv:1711.04231*.

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. pages 1724–1734.

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. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*.

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*, pages 823–833. Association for Computational Linguistics.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. *arXiv preprint arXiv:1705.01020*.

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

Diego Marcheggiani, Joost Bastings, and Ivan Titov. 2018. Exploiting semantics in neural machine translation with graph convolutional networks. *arXiv preprint arXiv:1804.08313*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Dongdong Zhang Shuangzhi Wu, Ming Zhou. 2017. Improved neural machine translation with source syntax. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4179–4185.

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. *arXiv preprint arXiv:1503.00075*.

Jörg Tiedemann. 2009. News from OPUS-A collection of multilingual parallel corpora with tools and interfaces. In *Recent advances in natural language processing*, volume 5, pages 237–248.