Towards Neural Machine Translation with Latent Tree Attention

James Bradbury and **Richard Socher**
james.bradbury@salesforce.com
rsocher@salesforce.com

Abstract

Building models that take advantage of the hierarchical structure of language without *a priori* annotation is a longstanding goal in natural language processing. We introduce such a model for the task of machine translation, pairing a recurrent neural network grammar encoder with a novel attentional RNNG decoder and applying policy gradient reinforcement learning to induce unsupervised tree structures on both the source and target. When trained on character-level datasets with no explicit segmentation or parse annotation, the model learns a plausible segmentation and shallow parse, obtaining performance close to an attentional baseline.

1 Introduction

Many efforts to exploit linguistic hierarchy in NLP tasks make use of the output of a self-contained parser system trained from a human-annotated treebank (Huang et al., 2006). An alternative approach aims to jointly learn the task at hand and relevant aspects of linguistic hierarchy, inducing from an unannotated training dataset parse trees that may or may not correspond to treebank annotation practices (Wu, 1997; Chiang, 2005).

Most deep learning models for NLP that aim to make use of linguistic hierarchy integrate an external parser, either to prescribe the recursive structure of the neural network (Pollack, 1990; Goller and Küchler, 1996; Socher et al., 2013) or to provide a supervision signal or training data for a network that predicts its own structure (Socher et al., 2010; Bowman et al., 2016; Dyer et al., 2016b). But some recently described neural network models take the second approach and treat hierarchical structure as a latent variable, applying inference over graph-based conditional random fields (Kim et al., 2017), the straight-through estimator (Chung et al., 2017), or policy gradient reinforcement learning (Yogatama et al., 2017) to work around the inapplicability of gradient-based learning to problems with discrete latent states.

For the task of machine translation, syntactically-informed models have shown promise both inside and outside the deep learning context, with hierarchical phrase-based models frequently outperforming traditional ones (Chiang, 2005) and neural MT models augmented with morphosyntactic input features (Sennrich and Haddow, 2016; Nadejde et al., 2017), a tree-structured encoder (Eriguchi et al., 2016; Hashimoto and Tsuruoka, 2017), and a jointly trained parser (Eriguchi et al., 2017) each outperforming purely-sequential baselines.

Drawing on many of these precedents, we introduce an attentional neural machine translation model whose encoder and decoder components are both tree-structured neural networks that predict their own constituency structure as they consume or emit text. The encoder and decoder networks are variants of the RNNG model introduced by Dyer et al. (2016b), allowing tree structures of unconstrained arity, while text is ingested at the character level, allowing the model to discover and make use of structure within words.

The parsing decisions of the encoder and decoder RNNGs are parameterized by a stochastic policy trained using a weighted sum of two objectives: a language model loss term that rewards predicting the next character with high likelihood, and a tree attention term that rewards one-to-one attentional correspondence between constituents in the encoder and decoder.

We evaluate this model on the German-English language pair of the `flickr30k` dataset, where it obtains similar performance to a strong character-

Proceedings of the 2nd Workshop on Structured Prediction for Natural Language Processing, pages 12–16
Copenhagen, Denmark, September 7–11, 2017.

level baseline. Analysis of the latent trees produced by the encoder and decoder shows that the model learns a reasonable segmentation and shallow parse, and most phrase-level constituents constructed while ingesting the German input sentence correspond meaningfully to constituents built while generating the English output.

2 Model

2.1 Encoder/Decoder Architecture

The model consists of a coupled encoder and decoder, where the encoder is a modified stack-only recurrent neural network grammar (Kuncoro et al., 2017) and the decoder is a stack-only RNNG augmented with constituent-level attention. An RNNG is a top-down transition-based model that jointly builds a sentence representation and parse tree, representing the parser state with a StackLSTM and using a bidirectional LSTM as a constituent composition function. Our implementation is detailed in Figure 1, and differs from Dyer et al. (2016b) in that it lacks separate new-nonterminal tokens for different phrase types, and thus does not include the phrase type as an input to the composition function. Instead, the values of $\mathbf{x}_i$ for the encoder are fixed to a constant $\mathbf{x}^{\text{enc}}$ while the values of $\mathbf{x}_j$ for the decoder are determined through an attention procedure (Section 2.2).

As originally described, the RNNG predicts parser transitions using a one-layer $\tanh$ perceptron with three concatenated inputs: the last state of a unidirectional LSTM over the stack contents ($\mathbf{s}$), the last state of a unidirectional LSTM over the reversed buffer of unparsed tokens ($\mathbf{b}$), and the result of an LSTM over the past transitions ($\mathbf{a}$). All three of these states can be computed with at most one LSTM step per parser transition using the StackLSTM algorithm (Dyer et al., 2016a). But such a baseline RNNG is actually outperformed by one which conditions the parser transitions only on the stack representation (Kuncoro et al., 2017). Restricting our model to this stack-only case allows both the encoder and decoder to be supervised using a language model loss, while allowing the model access to $\mathbf{b}$ would give it a trivial way to predict the next character and obtain zero loss.

2.2 Attention

With the exception of the attention mechanism, the encoder and decoder are identical. While the encoder uses a single token to represent a new nonterminal, the decoder represents a new nonterminal on the stack as a sum weighted by *structural attention* of the *phrase representations* of all nonterminal tree nodes produced by the encoder. In particular, we use the normalized dot products between the decoder stack representation $\mathbf{s}_j^{\text{dec}}$ and the stack representation at each encoder node $\mathbf{s}_{\text{enc}}^i$ (that is, the hidden state of the StackLSTM up to and including $\mathbf{x}_j^{\text{enc}}$ but not $\mathbf{h}_j^{\text{enc}}$) as coefficients in a weighted sum of the phrase embeddings $\mathbf{h}_{\text{enc}}^i$ corresponding to the encoder nodes:

$$\begin{aligned} \alpha_{ij} &= \operatorname*{softmax}_{\text{all } i}(\mathbf{s}_i^{\text{enc}} \cdot \mathbf{s}_j^{\text{dec}}) \\ \mathbf{x}_j^{\text{dec}} &= \sum_i \alpha_{ij} \mathbf{h}_i^{\text{enc}}. \end{aligned} \tag{1}$$

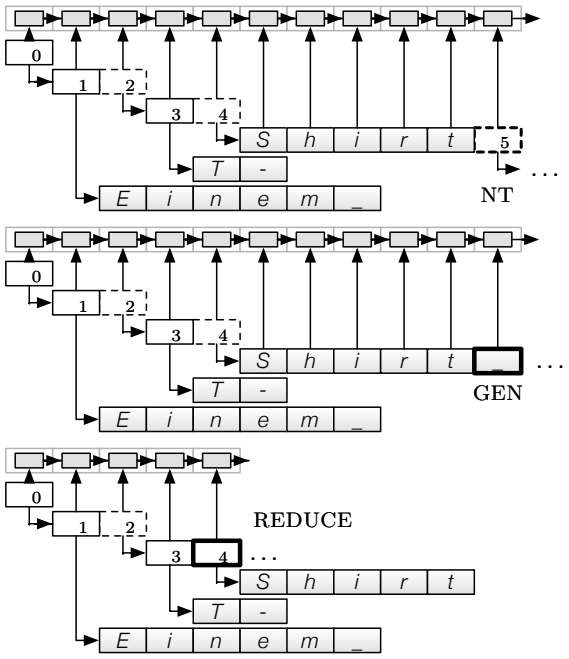

Figure 1: At a given timestep during either encoding or decoding there are three possible transitions (although one or more may be forbidden): begin a new nonterminal constituent (NT), predict and ingest a terminal (GEN), or end the current nonterminal (REDUCE). If the chosen transition is NT, the RNNG adds a new-nonterminal token $\mathbf{x}_{i+1}$ to the active constituent and begins a new nonterminal constituent (**1a**). If the transition is GEN, the RNNG predicts the next token (Section 2.3) and adds the ground-truth next token $\mathbf{e}$ from the context buffer at the cursor location (**1b**). If the transition is REDUCE, the contents of the active nonterminal are passed to the composition function, the new-nonterminal token $\mathbf{x}_i$ is replaced with the result of the composition $\mathbf{h}_i$, and the StackLSTM rolls back to the previously active constituent (**1c**). In all three cases, the StackLSTM then advances one step with the newly added token as input ($\mathbf{x}_{i+1}$, $\mathbf{e}$, or $\mathbf{h}_i$).

Since the dot products between encoder and decoder stack representations are a measure of structural similarity between the (left context of) the current decoder state and the encoder state. Within a particular decoder nonterminal, the model reduces to ordinary sequence-to-sequence transduction. Starting from the encoder's representation of the corresponding nonterminal or a weighted combination of such representations, the decoder will emit a translated sequence of child constituents (both nonterminal and terminal) one by one—applying attention only when emitting nonterminal children.

2.3 Training

We formulate our model as a stochastic computation graph (Schulman et al., 2015), leading to a training paradigm that combines backpropagation (which provides the exact gradient through deterministic nodes) and vanilla policy gradient (which provides a Monte Carlo estimator for the gradient through stochastic nodes).

There are several kinds of training signals in our model. First, when the encoder or decoder chooses the GEN action it passes the current stack state $\mathbf{s}$ through a one-layer softmax perceptron, giving the probability that the next token is each of the characters in the vocabulary. The language model loss $\mathcal{L}_k$ for each generated token is the negative log probability assigned to the ground-truth next token. The other differentiable training signal is the coverage loss $\mathcal{L}_c$, which is a measure of how much the attention weights diverge from the ideal of a one-to-one mapping. This penalty is computed as a sum of three MSE terms:

$$\begin{aligned}\mathcal{L}_c = &\operatorname*{mean}_{\text{all } i}(1 - \sum_{\text{all } j} \alpha_{ij})^2 \\ &+ \operatorname*{mean}_{\text{all } i}(1 - \max_{\text{all } j} \alpha_{ij})^2 \\ &+ \operatorname*{mean}_{\text{all } j}(1 - \max_{\text{all } i} \alpha_{ij})^2\end{aligned} \tag{2}$$

Backpropagation using the differentiable losses affects only the weights of the output softmax perceptron. The overall loss function for these weights is a weighted sum of all $\mathcal{L}_k$ terms and $\mathcal{L}_c$:

$$\mathcal{L} = 100\mathcal{L}_c + 10 \sum_{\text{all } k} \mathcal{L}_k \tag{3}$$

There are additionally nondifferentiable rewards r that bias the model towards or away from certain kinds of tree structures. Here, negative numbers correspond to penalties. We assign a tree reward of -1 when the model predicts a REDUCE with only one child constituent (REDUCE with zero child constituents is forbidden) or predicts two REDUCE or NT transitions in a row. This biases the model against unary branching and reduces its likelihood of producing an exclusively left- or right-branching tree structure. In addition, for all constituents except the root, we assign a tree reward based on the size and type of its children. If n and t are the number of nonterminal and terminal children, this reward is $4t$ if all children are terminal and $9\sqrt{n}$ otherwise. A reward structure like this biases the model against freely mixing terminals and nonterminals within the same constituent and provides incentive to build substantial tree structures early on in training so the model doesn't get stuck in trivial local minima.

Within both the encoder and decoder, each stochastic action node has a corresponding tree reward r_k if the action was REDUCE (otherwise zero) and a corresponding language model loss $\mathcal{L}_k$ if the action was GEN (otherwise zero). We subtract an exponential moving average baseline from each tree reward and additional exponential moving average baselines—computed independently for each character z in the vocabulary, because we want to reduce the effect of character frequency—from the language model losses. If $\text{GEN}(k)$ is the number of GEN transitions among actions one through k, and γ is a decay constant, the final reward $\mathcal{R}_k^m$ for action k with $m \in \{\text{enc}, \text{dec}\}$ is:

$$\begin{aligned}\hat{r}_k &= r_k - r_{\text{baseline}} \\ \hat{\mathcal{L}}_k &= \mathcal{L}_k - \mathcal{L}_{\text{baseline}}(z_k) \\ \hat{\mathcal{R}}_k &= \sum_{\kappa=k}^{K_m} \gamma^{\text{GEN}(\kappa)-\text{GEN}(k)}(\hat{r}_\kappa - \hat{\mathcal{L}}_\kappa^m) \\ \mathcal{R}_k^{\text{enc}} &= \hat{\mathcal{R}}_k - \mathcal{L}_c - (m = \text{enc}) \sum_{\kappa=1}^{K_{\text{dec}}} \mathcal{L}_k^{\text{dec}}.\end{aligned} \tag{4}$$

These rewards define the gradient that each stochastic node (with normalized action probabilities p_k^a and chosen action a_k) produces during backpropagation according to the standard multinomial score function estimator (REINFORCE):

$$\nabla_\theta p_k^a = \operatorname*{mean}_{\text{a}_\text{k}=\text{a}} \mathcal{R}_k \nabla_\theta \log p_k^{a_k} = \operatorname*{mean}_{\text{a}_\text{k}=\text{a}} \frac{-\mathcal{R}_k}{p_k^{a_k}} \tag{5}$$

3 Results

We evaluated our model on the German-English language pair of the `flickr30k` data, the tex-

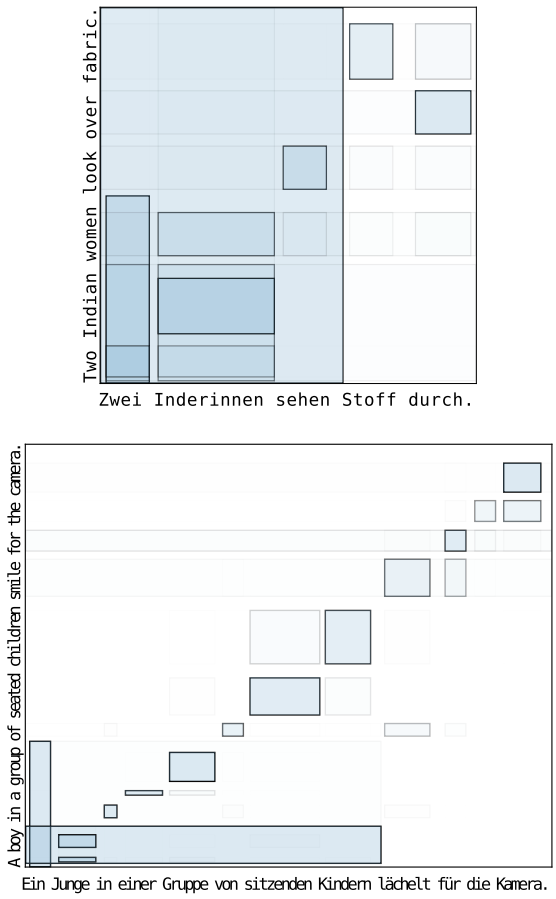

Figure 2: Attention visualizations for two sentences from the development set. Attention between two constituents is represented by a shaded rectangle whose projections on the x and y axes cover the encoder and decoder constituents respectively.

tual component of the WMT Multimodal Translation shared task (Specia et al., 2016). An attentional sequence-to-sequence model with two layers and 384 hidden units from the OpenNMT project (Klein et al., 2017) was run at the character level as a baseline, obtaining 32.0 test BLEU with greedy inference. Our model with the same hidden size and greedy inference achieves test BLEU of 28.5 after removing repeated bigrams.

We implemented the model in PyTorch, benefiting from its strong support for dynamic and stochastic computation graphs, and trained with batch size 10 and the Adam optimizer (Kingma and Ba, 2015) with early stopping after 12 epochs. Character embeddings and the encoder's $\mathbf{x}^{\text{enc}}$ embedding were initialized to random 384-dimensional vectors. The value of γ and the decay constant for the baselines' exponential moving average were both set to 0.95. A random selection of translations is included in the supplemental material, while two attention plots are shown in Figure 2. Figure 2b demonstrates a common pathology of the model, where a phrasal encoder constituent would be attended to during decoding of the head word of the corresponding decoder constituent, while the head word of the encoder constituent would be attended to during decoding of the decoder constituent corresponding to the whole phrase. Another common pathology is repeated sentence fragments in the translation, which are likely generated because the model cannot condition future attention directly on past attention weights (the "input feeding" approach introduced by Luong et al. (2015)).

Translation quality also suffers because of our use of a stack-only RNNG, which we chose because an RNNG with both stack and buffer inputs is incompatible with a language model loss. During encoding, the model must decide at the very beginning of the sentence how deeply to embed the first character. But with a stack-only RNNG, it must make this decision randomly, since it isn't able to use the buffer representation—which contains the entire sentence.

4 Conclusion

We introduce a new approach to leveraging unsupervised tree structures in NLP tasks like machine translation. Our experiments demonstrate that a small-scale MT dataset contains sufficient training signal to infer latent linguistic structure, and we are excited to learn what models like the one presented here can discover in full-size translation corpora. One particularly promising avenue of research is to leverage the inherently compositional phrase representations $\mathbf{h}_i^{\text{enc}}$ produced by the encoder for other NLP tasks.

There are also many possible directions for improving the model itself and the training process. Value function baselines can replace exponential moving averages, pure reinforcement learning can replace teacher forcing, and beam search can be used in place of greedy inference. Solutions to the translation pathologies presented in Section 3 are likely more complex, although one possible approach would leverage variational inference using a teacher model that can see the buffer and helps train a stack-only student model.

References

Samuel Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *ACL*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *ACL*.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2017. Hierarchical multiscale recurrent neural networks. In *ICLR*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2016a. Transition-based dependency parsing with stack long short-term memory. In *EMNLP*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah Smith. 2016b. Recurrent neural network grammars. In *NAACL*.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *ACL*.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. *arXiv preprint arXiv:1702.03525* .

Christoph Goller and Andreas Küchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *IEEE International Conference on Neural Networks*. IEEE, volume 1, pages 347–352.

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

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. A syntax-directed translator with extended domain of locality. In *CHPJI-NLP*. Association for Computational Linguistics.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander Rush. 2017. Structured attention networks. In *ICLR*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. *ArXiv preprint arXiv:1701.02810* .

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah Smith. 2017. What do recurrent neural network grammars learn about syntax? In *EACL*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

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

Jordan B Pollack. 1990. Recursive distributed representations. *Artificial Intelligence* 46(1):77–105.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. 2015. Gradient estimation using stochastic computation graphs. In *NIPS*.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *WMT*.

Richard Socher, Christopher Manning, and Andrew Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Lucia Specia, Stella Frank, Khalil Simaan, and Desmond Elliott. 2016. A shared task on multimodal machine translation and crosslingual image description. In *WMT*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational linguistics* 23(3):377–403.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to compose words into sentences with reinforcement learning. In *ICLR*.