Efficient Bilingual Generalization from Neural Transduction Grammar Induction

Yuchen YAN, Dekai WU, Serkan KUMYOL

Department of Computer Science and Engineering,
Human Language Technology Center,
The Hong Kong University of Science and Technology

{yyanaa|dekai|skumyol}@cs.ust.hk

Abstract

We introduce (1) a novel neural network structure for bilingual modeling of sentence pairs that allows efficient capturing of bilingual relationship via biconstituent composition, (2) the concept of neural network biparsing, which applies to not only machine translation (MT) but also to a variety of other bilingual research areas, and (3) the concept of a biparsing-backpropagation training loop, which we hypothesize that can efficiently learn complex biparse tree patterns. Our work distinguishes from sequential attention-based models, which are more traditionally found in neural machine translation (NMT) in three aspects. First, our model enforces compositional constraints. Second, our model has a smaller search space in terms of discovering bilingual relationships from bilingual sentence pairs. Third, our model produces explicit biparse trees, which enable transparent error analysis during evaluation and external tree constraints during training.

1. Introduction

In this paper, we introduce a neural network structure for modeling of bilingual sentence pairs that features efficient capturing of bilingual relationships by learning explicit compositional biconstituent structures, as opposed to conventional attention-based NMT models, which learn flat token-to-token bilingual relationships requiring numerous parallel corpora. Token-to-token bilingual relationship formalism is inefficient in two aspects. First, it lacks compositional structure, which is the key to generalizing biphrases from bitokens and generalizing bisentences from biphrases. Second, there are no constraints on the space of all possible token alignments, resulting in the attention layer inefficiently exploring strategys for such a huge search space. Our model skips directly to the compositional structure, representing bilingual relationships by recursively composing biconstituents with one extra degree of ordering flexibility.

We propose a new training strategy based on what we call a **biparsing-backpropagation training loop**, inspired by our hypothesis that good biparse trees lead to better models, and better models will compute better biparse trees, forming a faster feedback loop to efficiently capture bilingual relationships in low resource scenarios. When biparsing a corpus, desirable tree patterns tend to show up repeatedly because they explain more bilingual phenomenon, which will be learned during backpropagation. In the next epoch, these learned patterns will help compute more accurate biparse trees, revealing more complex and desirable tree patterns.

The paper is divided into two main parts. We begin in the first part below by laying out the basic formalism of **soft transduction grammars** and **soft biparse trees**, which are the neural network analogous to symbolic transduction grammars and symbolic biparse trees. We then introduce several competitive neural network designs and explains how the design decisions we make are suitable in terms of the generalizablity of the neural network structure and the expressiveness of the transduction grammar. In the second part, we explain how our neural network model implements the formalism of a soft transduction grammar, together with a pipeline for the biparsing-backpropagation training loop. Afterwards, a small experiment is presented demonstrating how this feedback loop discovers biparse tree patterns by showing how biparse trees evolve over time.

2. Related works

To our knowledge there is no published research in NMT that works with biparse trees besides the basic TRAAM model (Addanki and Wu, 2014), which is an incomplete model that can perform neither MT nor biparsing on its own. However, there have been many attempts to incorporate monolingual trees (mostly syntactic trees) into MT systems.

Most of the related work started from the seq2seq architecture, linearizing a syntactic parse tree into a flat sequence in depth first search order, which we hypothesize is a non-optimal representation since linearizing inevitably separates related sibling tree nodes apart, resulting in unnecessary distant dependencies. Vinyals *et al.* (2015) trained a seq2seq model translating from monolingual sentences to their linearized parse trees (without tokens), effectively building a neural network syntactic parser. Aharoni and Goldberg (2017) proposed linearizing the target parse tree with tokens, resulting in a model that can translate from a source sentence to a target parse tree (with tokens). Furthermore, Ma *et al.* (2018) proposed a way to linearize an entire weighted parse forest into a sequence. Another variation of this was proposed by Wang *et al.* (2018) introducing additional connections to LSTM, so that when generating a tree node, an LSTM unit has direct access to output from its parent node.

Another approach is to use a recursive unit naturally following the tree structure, so that linearization is no longer required at the encoder side, which is an improvement but still requiring syntactic parse trees as additional input. Eriguchi *et al.* (2017) proposed using Tree-LSTM (Tai *et al.*, 2015) to encode source sentences along the topology of a syntactic parse tree.

3. Soft transduction grammars

We propose a new concept called a **soft transduction grammar**, which uses **soft biparse trees** to explain sentence pairs, in contrast to traditional **transduction grammars** which use **biparse trees**. Our new soft transduction grammar has the advantage of not having to keep track of a combinatorically exploding number of nonterminal categories and rules, thus significantly reducing computational complexity while retaining its expressiveness of bilingual relationships.

Formally, a soft transduction grammar consists of:

- An output language vocabulary and an input language vocabulary.
- A function **bi_lexicon_embed** that takes a biconstituent as input, and returns a biconstituent embedding.
- A function **bi_lexicon_readout** that takes a biconstituent embedding as input, and returns a bilexicon.
- A function **bi_lexicon_evaluate** that takes a biconstituent as input, and returns a degree of goodness based on whether the given biconstituent is a valid bilexicon.
- A function **bi_compose** that takes (1) a list of biconstituent embeddings in output language order, (2) the list of the same biconstituent embeddings in input language order as input, and returns a composed biconstituent embedding.
- A function **bi_decompose** that takes a biconstituent embedding as input, and returns (1) a list of biconstituent embeddings in the output language order, (2) a list of the same biconstituent embeddings in the input language order.
- A function **bi_compose_evaluate** that takes (1) a list of biconstituent embeddings in the output language order, (2) a list of the same biconstituent embeddings in the input language order as input, and returns a degree of goodness based on whether the given biconstituents ”do compose nicely.”

A soft transduction grammar is capable of performing a variety of bilingual tasks using algorithms similar to those of a symbolic transduction grammar. These bilingual tasks include:

- **Parallel sentence embedding**: Takes a bilingual sentence pair as input, and returns an biconstituent embedding.
- **Parallel sentence generation**: Takes a biconstituent embedding as input, and generates a bilingual sentence pair. Note that getting back the exact original sentence is unlikely for a long sentence if biconstituent composition is lossy. However, we hypothesize that a good soft transduction grammar will try to preserve the syntactic/semantic structure of the original sentence pair.
- **Tree recognition**: Takes a biparse tree as input, and calculates a degree of goodness.
- **Biparsing**: Takes a bilingual sentence pair as input, and finds the best biparse tree.
- **Transduction**: Takes an input language sentence as input, and returns a sentence in output langauge (or vise versa).

3.1. Inversion transduction grammar

We are working with a special type of transduction grammar, called ITGs, or inversion transduction grammars

(Wu, 1997) which has an empirically appropriate ordering flexibility, small enough to retain efficient computation and general enough to explain almost every alignment in natural language transductions. When an ITG composes child biconstituents into a parent biconstituent, the input language constituents may read in either the same order as the output language constituents or the reverse order with the output language constituents. When working with soft transduction grammars, all biparse trees will have their nonterminal categories and preterminal categories be replaced with biconstituent embeddings. We call such biparse trees **soft biparse trees**.

To keep the model simple, we work with parse/biparse trees in **2 Normal Form**, which means a biconstituent has exactly 2 child biconstituents. In addition, the size of a bilexicon must be restricted to avoid overfitting. In our work, a bilexicon must satisfy $\min(l_e, l_f) \leq 2$ where l_e is output language phrase length and l_f is input language phrase length. Note that in this paper the letter **e** and **f** always refer to the output language and the input language respectively.

4. Gated RAAM: composing and decomposing embedding vectors

Our improvement on recursive auto-associative memory, which we call **Gated RAAM** or **GRAAM**, can recursively compose two embeddings into one (of the same dimension) while being more specialized to recursively unfold it back to the original embeddings when compared with conventional RAAMs or recursive auto encoders, for two reasons: (1) the loss is directly defined as the reconstruction error of the recursively unfolded original embedding, and (2) both the composer and the decomposer networks apply gated connections similar to an LSTM (long short term memory) (Hochreiter and Schmidhuber, 1997) or a GRU (gated recurrent unit) (Cho *et al.*, 2014), thus being able to learn from error signals multiple composition and decomposition steps away.

Note that GRAAM is a monolingual concept. How to generalize it bilingually will be introduced in the next section.

We introduce the new concept of **decaying unfolding recursive loss**, that directly targets leaf node reconstruction while encouraging the RAAM or GRAAM network to prioritize remembering closer children over distant children. In contrast, the original unfolding recursive loss (Socher *et al.*, 2011) doesn't account for the fact that for a natural language parse tree, shallower terminals (often representing main sentence structures) should be prioritized over deeper ones (often representing supplementary modifiers or even nested clauses). To solve this problem, our new decaying unfolding loss has different weightings for leaves of different depths. At each level deeper into the tree, the reconstruction loss is scaled by a decaying factor γ. In our model, we choose $\gamma = 0.5$.

Along with the new loss metric, we propose a pair of new composer and decomposer designs, solving the vanishing gradient problem brought by the decaying unfolding recursive loss, borrowing the gated connection idea from LSTM and GRU.

First we introduce the decomposer. The decomposer takes an embedding $\mathbf{y}$ as input, and returns the decomposed children $\hat{\mathbf{x}_0}, \hat{\mathbf{x}_1}$, denoted as $\hat{\mathbf{x}}_0, \hat{\mathbf{x}}_1 = \text{decompose}(\mathbf{y})$. The decomposer consists of two sub-components of an identical structure (but having different sets of internal parameters), one for predicting the left children and one for predicting the right children. This sub-component is mathematically equivalent to a GRU having no other input but its cell state: $\hat{\mathbf{x}}_0 = \text{GRU}_0(\mathbf{y}, \mathbf{0})$ and $\hat{\mathbf{x}}_1 = \text{GRU}_1(\mathbf{y}, \mathbf{0})$ where $\mathbf{0}$ is the zero vector.

Next we introduce the composer. The composer takes two child embeddings $\mathbf{x_0}, \mathbf{x_1}$ and returns composed embedding $\mathbf{y}$, denoted as $\mathbf{y} = \text{compose}(\mathbf{x_0}, \mathbf{x_1})$. Note that a Tree-LSTM (Tai *et al.*, 2015) does not work, because it doesn't satisfy the requirement that backpropagation from either $\mathbf{y}$ to $\mathbf{x_0}$ or from $\mathbf{y}$ to $\mathbf{x_1}$ should not cause the vanishing gradient problem. Here is our new structure:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x_0} \\ \mathbf{x_1} \end{bmatrix} \tag{1}$$

$$\mathbf{p_0} = \mathbf{x} \otimes \sigma(\mathbf{W_0 x} + \mathbf{b_0}) \tag{2}$$

$$\mathbf{p_1} = \mathbf{x} \otimes \sigma(\mathbf{W_1 x} + \mathbf{b_1}) \tag{3}$$

$$\mathbf{c} = \tanh(\mathbf{W_2} \begin{bmatrix} \mathbf{p_0} \\ \mathbf{p_1} \end{bmatrix} + \mathbf{b_2}) \tag{4}$$

$$\mathbf{f_0} = \mathbf{W_3 x} + \mathbf{b_3} \tag{5}$$

$$\mathbf{f_1} = \mathbf{W_4 x} + \mathbf{b_4} \tag{6}$$

$$[\mathbf{y_0}; \mathbf{k_c}; \mathbf{y_1}] = \text{softmax}([\mathbf{f_0}; \mathbf{0}; \mathbf{f_1}]^T)^T \tag{7}$$

$$\mathbf{y} = \mathbf{x_0} \otimes \mathbf{k_0} + \mathbf{c} \otimes \mathbf{k_c} + \mathbf{x_1} \otimes \mathbf{k_1} \tag{8}$$

where $\mathbf{W_0}, \mathbf{W_1}, \mathbf{W_2}, \mathbf{W_3}, \mathbf{W_4}, \mathbf{b_0}, \mathbf{b_1}, \mathbf{b_2}, \mathbf{b_3}, \mathbf{b_4}$ are internal parameters.

The GRAAM composer is different from a GRU in the sense that it has two cell states to merge. In GRU, the forget gate ensures that the weight of the **update candidate** and the weight of the **original cell state** sum to 1. In GRAAM composer, however, the forget gate has to ensure that the weights of update candidate $\mathbf{k_c}$, and the weights of the two original cell states $\mathbf{k_0}, \mathbf{k_1}$ sum to 1. This is the reason that a softmax function is required.

5. Gated Parallel RAAM: representing bilingual relationships

In order to model bilingual translation relationships, we introduce a generalization of GRAAM called **Gated Parallel RAAM** or **GPRAAM**, equipped with two new concepts:

(1) **Parallel composer/decomposer** in contrast to **integrated composer/decomposer** (from TRAAM). Our new design ensures that the loss of mispredicting the output language order is of the same magnitude as the loss of mispredicting the input language order, which we hypothesize could prevent the network from prioritize learning from either language's ordering.

(2) **Parallel biconstituent embedding** in contrast to **integrated biconstituent embedding** (from TRAAM). Our new design enables learning from monolingual data, because the way that the two GRAAMs are loosely coupled makes it possible to break the transduction model into two monolingual language models (or vise versa put them back). Being able to decouple enables monolingual training as a bootstrap step, which we hypothesize could greatly reduce the number of bilingual training epochs required.

An integrated composer/decomposer contains one composer and one decomposer. The composer composes an additional permutation bit along with the two child biconstituent embeddings, whereas the decomposer decomposes the composed biconstituent embedding back into the two child biconstituent embeddings together with a permutation bit, as shown in Figure 1. It suffers from penalty unfairness: (1) when mispredicting both the input and the output language order, error signals are coming from the two predicted embeddings (but not the permutation bit); (2) when mispredicting only the input language order, error signal are coming from only the predicted permutation bit; (3) when mispredicting only the output language order, error signals are coming from all three predictions (which is totally undesirable).

A parallel composer/decomposer contains two composer/decomposer, one composing/decomposing child biconstituents in the output language order, and the other composing/decomposing in the input language order, as shown in Figure 2. The penalties for mispredicting biconstituent orderings are fair.

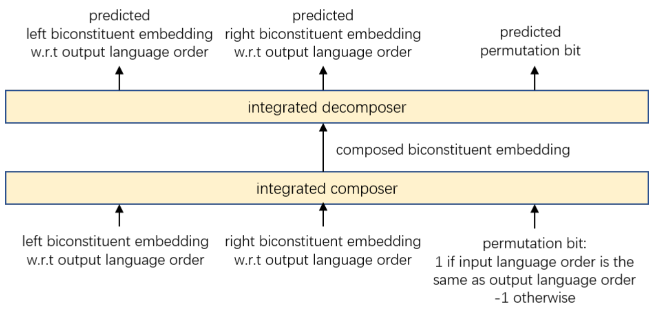
predicted
left biconstituent embedding
w.r.t output language order
predicted
right biconstituent embedding
w.r.t output language order
predicted
permutation bit
integrated decomposer
composed biconstituent embedding
integrated composer
left biconstituent embedding
w.r.t output language order
right biconstituent embedding
w.r.t output language order
permutation bit:
1 if input language order is the same as output language order
-1 otherwise

Figure 1: integrated composer/decomposer with integrated embedding

We also propose the new idea that a biconstituent embedding can also be decoupled into two separated embedding vectors: an **output language partition** and an **input language partition**. Such a design is called **parallel embedding**. This new design makes it possible to decouple the entire soft transduction grammar into two soft CFGs, and having them bootstrapped separately on monolingual corpora. The parallel embedding design requires a pair of bridges that can translate between an output language partition and an input language partition: (1) the **output language to input language bridging function** or **e_f_bridge**, and (2) the **input language to output language bridging function** or **f_e_bridge**, as shown in Figure 3. The bridges can be implemented by various neural network structures, and in our work, both e_f_bridge and f_e_bridge are implemented by a simple 3-layer feedforward network with residual connections.

6. Implementation details

Our complete model consists of (1) a GPRAAM, (2) an input language word embedding lookup table, (3) an output language word readout layer, and (4) an input language word readout layer. These components together implement all the functions required in the soft transduction grammar formalism, which will be explained below.

Note that with parallel embedding design, a **biconstituent embedding** consists of both an output language embedding and an input language embedding.

bi_lexicon_embed: implementing this function relies on the fact that our model can be broken down into

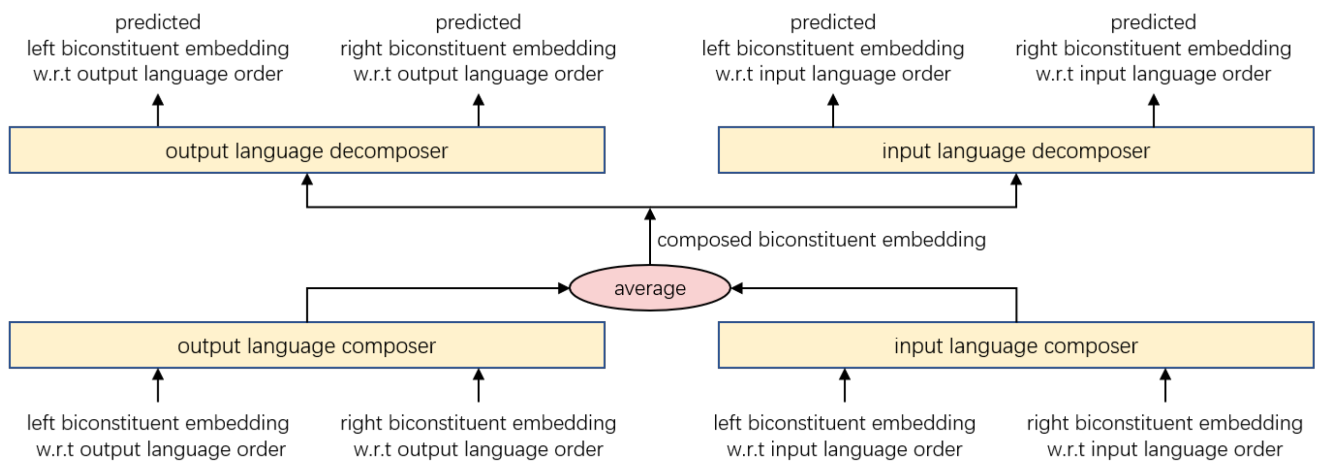
predicted
left biconstituent embedding
w.r.t output language order
predicted
right biconstituent embedding
w.r.t output language order
predicted
left biconstituent embedding
w.r.t input language order
predicted
right biconstituent embedding
w.r.t input language order
output language decomposer
input language decomposer
composed biconstituent embedding
average
output language composer
input language composer
left biconstituent embedding
w.r.t output language order
right biconstituent embedding
w.r.t output language order
left biconstituent embedding
w.r.t input language order
right biconstituent embedding
w.r.t input language order

Figure 2: parallel composer/decomposer with integrated embedding

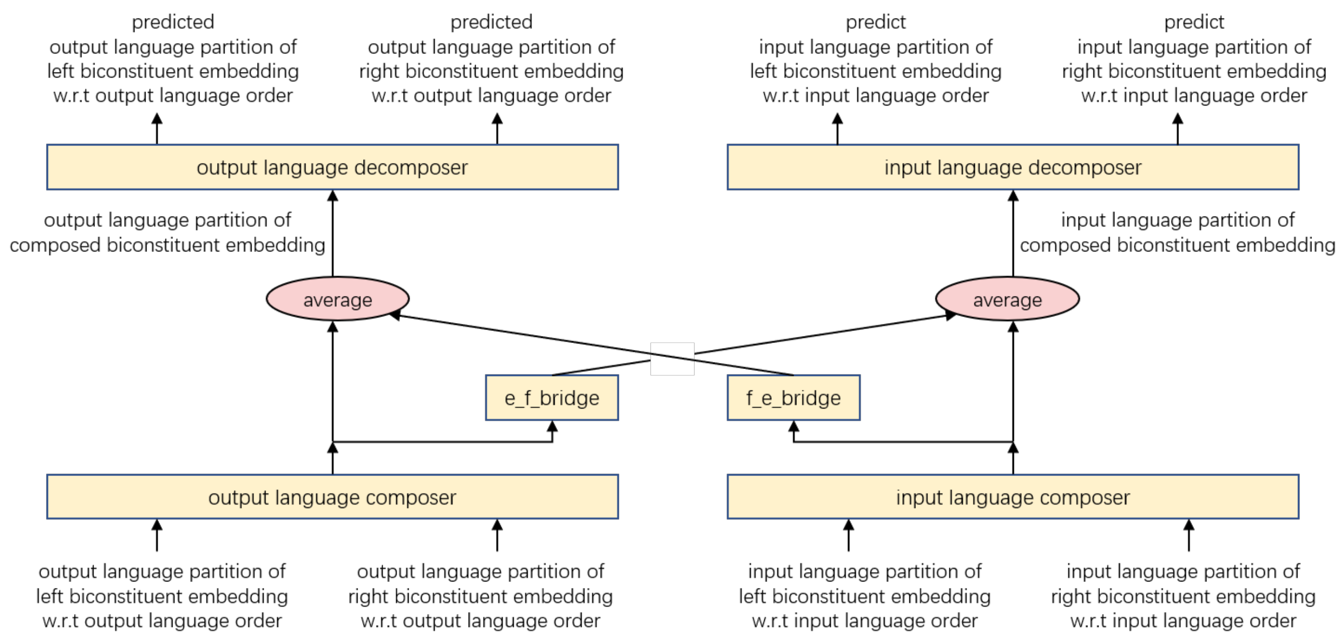
predicted
output language partition of
left biconstituent embedding
w.r.t output language order
predicted
output language partition of
right biconstituent embedding
w.r.t output language order
predict
input language partition of
left biconstituent embedding
w.r.t input language order
predict
input language partition of
right biconstituent embedding
w.r.t input language order
output language decomposer
input language decomposer
output language partition of
composed biconstituent embedding
input language partition of
composed biconstituent embedding
average
average
e_f_bridge
f_e_bridge
output language composer
input language composer
output language partition of
left biconstituent embedding
w.r.t output language order
output language partition of
right biconstituent embedding
w.r.t output language order
input language partition of
left biconstituent embedding
w.r.t input language order
input language partition of
right biconstituent embedding
w.r.t input language order

Figure 3: parallel composer/decomposer with parallel embedding

an output language model and an input language model, with each monolingual language model capable of composing a monolingual phrase into a monolingual embedding. To compose a monolingual phrase, perform viterbi parsing on the phrase and then use the monolingual GRAAM to recursively compose constituents.

bi_lexicon_readout: the output language phrase is generated by the output language model from the output language embedding. Similarly, the input language phrase is generated by the input language model from the input language embedding. To generate a monolingual phrase from a monolingual embedding, we recursively decompose it using corresponding GRAAM.

bi_lexicon_evaluate the bilexicon degree of goodness is computed from the reconstruction error. The reconstruction error is a sum of (1) **E→E reconstruction error** computed by reconstructing the output language phrase from output language embedding, (2) **E→F reconstruction error** computed by reconstructing the input language phrase from output language embedding (with the help from embedding bridge), (3) **F→E reconstruction error** and (4) **F→E reconstruction error**, where (3) and (4) are computed in ways similar to those of (1) and (2), respectively.

bi_compose the output language embedding is computed by composing output language child embeddings with output language GRAAM; the input language embedding is computed by composing input language child embeddings with input language GRAAM.

bi_decompose: the output language child embeddings are computed by decomposing the output language parent embedding; the input language child embeddings are computed by decomposing the input language parent child embedding. After the child embeddings have been computed, we have two hypothesis: either (1) the left/right output language child aligns to the left/right input language child or (2) the left/right output language child aligns to the right/left input language child. The embedding bridges are applied to test which permutation incurs less bridging error.

bi_compose_evaluate: the degree of goodness of a biconstituent composition is computed from the reconstruction error. The reconstruction error is a sum of (1) **E→E reconstruction error** computed by reconstructing back the child output language embeddings from the composed output language embedding, (2) **E→E reconstruction error** computed by reconstructing back the child input language embeddings from the composed output language embedding, (3) **F→E reconstruction error** and (4) **F→F reconstruction error**, where (3) and (4) are computed in ways similar to those of (1) and (2), respectively.

6.1. Training

The training pipeline is divided into three steps. Firstly, we break the bilingual corpus into output language and input language corpus and perform monolingual **parse backpropagation** training loop, in which each sentence is Viterbi parsed, followed by backpropagation on the reconstruction loss along the parse tree. After this step, both the output language GRAAM and the input language GRAAM should have already learned how to compose monolingual sentences. Secondly, we bootstrap the two embedding bridges by training them individually on the parallel embedding dataset (which is a byproduct from the previous step). Finally, we perform a **biparse backpropagation** training loop on bilingual corpus, in which each sentence pair is Viterbi biparsed, followed by backpropagation on reconstruction loss along the biparse tree.

7. Experiments

As proof of concept, we have experimental results on the Chinese-English blocks world dataset, that has 90 parallel training sentences and 25 parallel development sentences, comprised of commands manipulating different colored objects over different shapes. The dataset was kept simple so as to provide an easy and transparent view of how the model was discovering biparse tree patterns through epochs.

During the stage of bilingual training, we monitored how the Viterbi biparse trees evolved over time to have a transparent visualization of how the model was learning. We first analyzed a short sentence, as shown in Figure 4.

- Epoch#5: The biparse tree didn't make sense.
- Epoch#10: The model correctly learned the noun bilexicons. The verb ”put” was mistranslated as ”上” (which means ”on”), but considering the fact that the training data only contained ”on” not ”under”, the model had found a good explanation.
- Epoch#15: The model improved its understanding of how constituents compose. ”On the red circle” is a better constituent than ”the block on”.
- Epoch#20: Nothing was improved. This biparse tree was finalized.

We then analyzed a longer sentence, as shown in Figure 5.

- Epoch#5: The biparse tree didn't make sense.
- Epoch#10: The model correctly learned some noun bilexicons. Consistent with what was happening with the shorter sentence, the model tried to use its special understanding of ”put” and ”on” to interpret this sentence pair and it worked to some degree.
- Epoch#15: The model got most part of the biparse tree correct, but was a little bit confused of how to handle the determinant ”the”.
- Epoch#20: The model correctly figured out how to handle determinant ”the”.
- Epoch#25: Nothing was improved. This biparse tree was finalized.

As can be seen our model has done an excellent job capturing bilingual generalization in our experiments. We also compared the performance of our model with the Transformer (Vaswani *et al.*, 2017) for Chinese to English MT on this dataset. We understand the fact that such comparison is to some extend unfair since the Transformer (as well as many other NMT systems) is

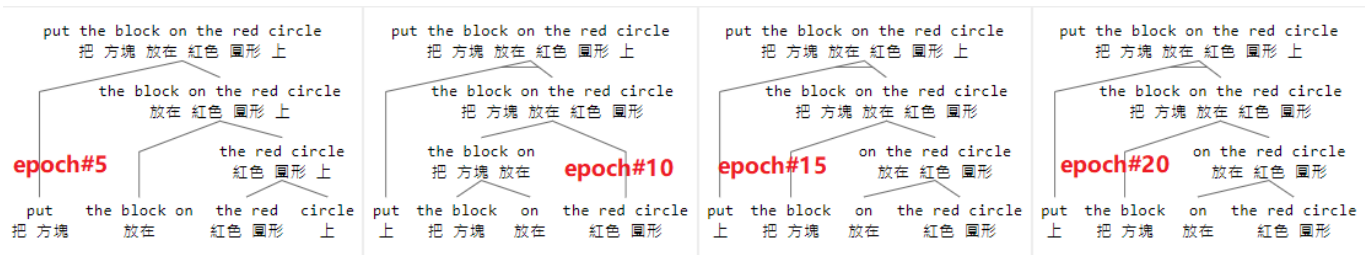

Figure 4: The Viterbi biparse tree of a short dev sentence at different epochs, demonstrating the process of the modeling learning bilingual generalizations. Noun biphrases were learned at epoch#10 and longer biconstituents were learned at epoch#15.

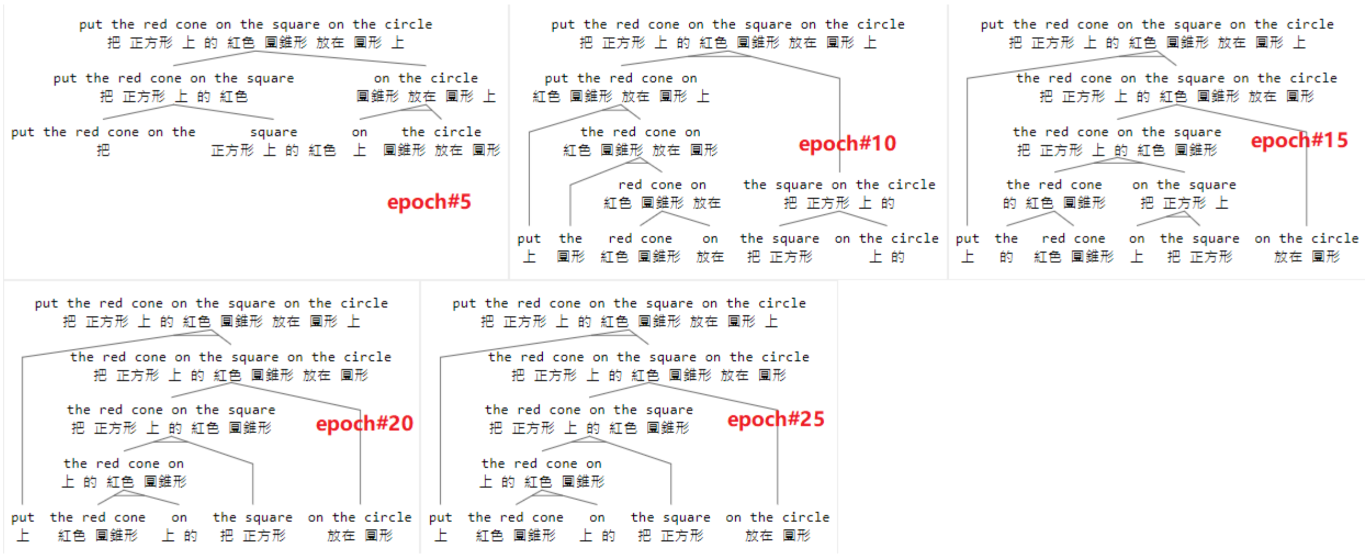
put the red cone on the square on the circle
把 正方形 上 的 紅色 圓錐形 放在 圓形 上
epoch#5
epoch#10
epoch#15
epoch#20
epoch#25
the red cone on the square on the circle
把 正方形 上 的 紅色 圓錐形 放在 圓形
the red cone on the square
把 正方形 上 的 紅色 圓錐形
the red cone on
上 的 紅色 圓錐形
put
上
the red cone
紅色 圓錐形
on
上 的
the square
把 正方形
on the circle
放在 圓形

Figure 5: The Viterbi biparse tree of a longer dev sentence at different epochs, demonstrating the process of the modeling learning bilingual generalizations. Noun biphrases were learned at epoch#10, followed by determinant ”the” being learned at epoch #20.

not designed for low-resource training. Nevertheless, our model had a BLEU score 0.78 and the transformer scored 0.71.

8. Conclusion

The twin concepts of soft transduction grammar and soft biparsing have been proposed in this paper, bringing a new approach to the field of bilingual deep learning. We have introduced new neural network implementation of the soft transduction grammar, and demonstrated its excellent ability in capturing generalization of bilingual relationships with a small amount of data. Its desirable properties include its compositional structure enforcement, constrained search space and explicit representation of tree structure. Besides MT and biparsing, various tasks are naturally incorporated as subproblems, such as parallel sentence evaluation, parallel sentence embedding, and parallel sentence generation, and a high degree of compatibility with monolingual language models is retained.

We are currently pursuing several directions for future development. We are deploying performance optimizations so that experiments can be completed on large corpora in a reasonable amount of time. We are also trying alternative composer, decomposer and bridge designs to see the impact of different architectures. Furthermore, in contrast to learning from bilingual corpora, we are working on incorporating external tree constraints so that the model can be forced to learn certain biparse tree patterns.

9. Acknowledgements

This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under LORELEI contract HR0011-15-C-0114, BOLT contracts HR0011-12-C-0014 and HR0011-12-C-0016, and GALE contracts HR0011-06-C-0022 and HR0011-06-C-0023; by the European Union under the Horizon 2020 grant agreement 645452 (QT21) and FP7 grant agreement 287658; and by the Hong Kong Research Grants Council (RGC) research grants 16215718, 16214315, 16210714, 620811 and 621008. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, the EU, or RGC.

10. References

Karteek Addanki and Dekai Wu. Transduction Recursive Auto-Associative Memory: Learning Bilingual Compositional Distributed Vector Representations of Inversion Transduction Grammars. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 112–121, Stroudsburg, PA, USA, 2014. Association for Computational Linguistics.

Roee Aharoni and Yoav Goldberg. Towards String-to-Tree Neural Machine Translation. In *Proceedings ofthe 55th Annual Meeting ofthe Association for Computational Linguistics (Short Papers)*, number 2015, pages 132–140, 2017.

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

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

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, nov 1997.

Chunpeng Ma, Akihiro Tamura, Masao Utiyama, Tiejun Zhao, and Eiichiro Sumita. Forest-Based Neural Machine Translation. In *Proceedings ofthe 56th Annual Meeting ofthe Association for Computational Linguistics (Long Papers)*, pages 1253–1263. Association for Computational Linguistics, 2018.

Richard Socher, Eric H Huang, Jeffrey Pennington, Andrew Y Ng, and Christopher D Manning. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection Distributional Seman... *Advances in neural information processing systems*, pages 801—-809, 2011.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. *arXiv preprint arXiv:1503.00075*, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, (Nips):6000–6010, 2017.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a Foreign Language. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, volume 2, pages 2773–2781, 2015.

Xinyi Wang, Hieu Pham, Pengcheng Yin, and Graham Neubig. A Tree-based Decoder for Neural Machine Translation. In *Proceedings ofthe 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4772–4777, 2018.

Dekai Wu. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23(3):377–403, 1997.