Compositional generalization by factorizing alignment and translation

Jacob Russin
Psychology Department
UC Davis
jlrussin@ucdavis.edu

Jason Jo
MILA
Université de Montréal

Randall C. O'Reilly
Psychology Department
Computer Science Department
Center for Neuroscience
UC Davis

Yoshua Bengio
MILA
Université de Montréal
CIFAR Senior Fellow

Abstract

Standard methods in deep learning for natural language processing fail to capture the compositional structure of human language that allows for systematic generalization outside of the training distribution. However, human learners readily generalize in this way, e.g. by applying known grammatical rules to novel words. Inspired by work in cognitive science suggesting a functional distinction between systems for syntactic and semantic processing, we implement a modification to an existing approach in neural machine translation, imposing an analogous separation between alignment and translation. The resulting architecture substantially outperforms standard recurrent networks on the SCAN dataset, a compositional generalization task, without any additional supervision. Our work suggests that learning to align and to translate in separate modules may be a useful heuristic for capturing compositional structure.

1 Introduction

A crucial property underlying the expressive power of human language is its systematicity (Lake et al., 2017; Fodor and Pylyshyn, 1988): syntactic or grammatical rules allow arbitrary elements to be combined in novel ways, making the number of sentences possible in a language to be exponential in the number of its basic elements. Recent work has shown that standard deep learning methods in natural language processing fail to capture this important property: when tested on unseen combinations of known elements, standard models fail to generalize (Lake and Baroni, 2018; Loula et al., 2018; Bastings et al., 2018). It has been suggested that this failure represents a major deficiency of current deep learning models, especially when they are compared to human learners (Marcus, 2018; Lake et al., 2017, 2019).

From a statistical-learning perspective, this failure is quite natural. The neural networks trained on compositional generalization tasks fail to generalize because they have memorized biases that do indeed exist in the training set. These tasks require networks to make an out-of-domain (o.o.d.) *extrapolation* (Marcus, 2018), rather than merely *interpolate* according to the assumption that training and testing data are independent and identically distributed (i.i.d.). To the extent that humans can perform well on certain kinds of o.o.d. tests, they must be utilizing inductive biases that are lacking in current deep learning models (Battaglia et al., 2018).

It has long been suggested that the human capacity for systematic generalization is linked to mechanisms for processing syntax, and their functional separation from the meanings of individual words (Chomsky, 1957; Fodor and Pylyshyn, 1988). In this work, we take inspiration from this idea and explore operationalizing it as an inductive bias in an existing neural network architecture.

First, we notice a connection between syntactic structure and the correct alignment of words in the source sequence to meanings in the target. In our model, alignment is accomplished with an attention mechanism (Bahdanau et al., 2015) that determines the relevance of each word in the source to the translation of the next word in the target. This process must take into account the syntactic structure of both sequences (e.g. if a verb was just translated, it would be important to know whether there is in the source sequence an adverb that modifies it). We reasoned that if alignment was separated from direct translation (analogous to a separation of syntax and the meanings of individual words),

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, pages 313–327

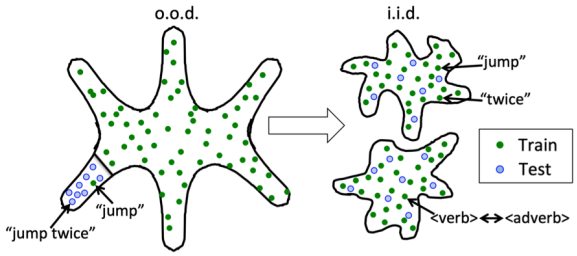

Figure 1: Illustration of the transformation of an out-of-domain (o.o.d.) generalization problem into two independent, identically distributed (i.i.d.) problems.

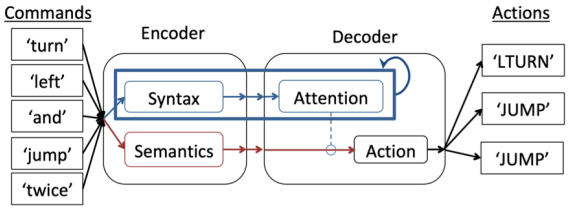

Figure 2: Syntactic Attention architecture. Information used for alignment ("syntax", shown in blue) is kept separate from information used for direct translation ("semantics", shown in red).

the difficult o.o.d. problem of composing known elements into a novel combination would be reduced to two easier i.i.d. problems, because the distributions of correct alignments and translations would be similar in training and testing data (see Figure 1).

We implemented this intuition by modifying an existing attention mechanism (Bahdanau et al., 2015), and call the resultant architecture Syntactic Attention to reflect the intuition that the attention mechanism used for alignment should operate primarily on syntactic information, which should be separated from the information relevant to translating individual words. We show that this modification achieves substantially improved compositional generalization performance over the original architecture on the SCAN dataset.

2 Syntactic Attention

The Syntactic Attention model improves the compositional generalization capability of an existing attention mechanism (Bahdanau et al., 2015) by separating two streams of information processing for alignment and translation (see Figure 2). We describe the mechanisms of this separation and the other details of the model below.

2.1 Factorizing alignment and translation

In the seq2seq problem, models must learn a mapping from arbitrary-length sequences of inputs $\mathbf{x} = \{x_1, x_2, ..., x_{T_x}\}$ to arbitrary-length sequences of outputs $\mathbf{y} = \{y_1, y_2, ..., y_{T_y}\}$: $p(\mathbf{y}|\mathbf{x})$. The underlying assumption made by the Syntactic Attention architecture is that the dependence of target words on the input sequence can be separated into two independent factors. One factor, $p(y_i|x_j)$, models the conditional distribution from individual words in the input to individual words in the target. Note that, unlike in the model of Bahdanau et al. (2015), these x_j do not contain any information about the other words in the input sequence because they are not processed with an RNN. The other factor, $p(j \rightarrow i|\mathbf{x}, y_{1:i-1})$, models the conditional probability that word j in the input is relevant to word i in the target sequence, given the entire input sequence. This alignment is accomplished from encodings of the inputs produced by an RNN. The crucial architectural assumption, then, is that any temporal dependency between individual words in the input that can be captured by an RNN should only be relevant to their alignment to words in the target sequence, and not to the translation of individual words. This assumption will be made clearer in the model description below.

2.2 Encoder

The encoder produces two separate vector representations for each word in the input sequence. Unlike the previous attention model (Bahdanau et al., 2015)), we separately extract the information that will be used for direct translation with a linear transformation: $m_j = W_m x_j$, where W_m is a learned weight matrix that multiplies the one-hot encodings $\{x_1, ..., x_{T_x}\}$. Note that these representations do not contain any information about the other words in the sentence. As in the previous attention mechanism (Bahdanau et al., 2015), we use a bidirectional RNN (biRNN) to extract the information that will be used for alignment. The biRNN produces a vector for each word on the forward pass, $(\overrightarrow{h_1}, ..., \overrightarrow{h_{T_x}})$, and a vector for each word on the backward pass, $(\overleftarrow{h_1}, ..., \overleftarrow{h_{T_x}})$. The representation of each word x_j is determined by the two vectors $\overrightarrow{h_{j-1}}, \overleftarrow{h_{j+1}}$ corresponding to the words surrounding it: $h_j = [\overrightarrow{h_{j-1}}; \overleftarrow{h_{j+1}}]$.

In all experiments, we used a bidirectional LSTM for this purpose. Note that h_j is encoding the context of the surrounding words in the sen-

tence. Our motivation for doing this was to force the RNN in the encoder to rely on the "role" the word is playing in the sentence. Note also that because there is no sequence information in the m_j, all of the information required to align the input sequence correctly (e.g. phrase structure, modifying relationships, etc.) must be encoded by the biRNN.

2.3 Decoder

The decoder models the conditional probability of each target word given the input and the previous targets: $p(y_i|y_1, y_2, ..., y_{i-1}, \mathbf{x})$, where y_i is a target and $\mathbf{x}$ is the whole input sequence. As in the previous model, we use an RNN to determine an attention distribution over the inputs at each time step (i.e. to align words in the input to the current target). However, our decoder diverges from this model in that the mapping from inputs to outputs is performed from a weighted average of the m_j:

$$p(y_i|y_{1:i-1}, \mathbf{x}) = f(d_i) \quad d_i = \sum_{j=1}^{T_x} \alpha_{ij} m_j \quad (1)$$

where f is parameterized by a linear function with a softmax, and the α_{ij} are the weights determined by the attention model. The attention weights are computed by a function measuring how well the input representations h_j align with the current hidden state of the decoder RNN, s_i:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \qquad e_{ij} = a(s_i, h_j) \quad (2)$$

where e_{ij} can be thought of as measuring the importance of a given input word x_j to the current target word y_i, and s_i is the current hidden state of the decoder RNN. Bahdanau et al. (2015) model the function a with a feedforward network, but we choose to use a simple dot product: $a(s_i, h_j) = s_i \cdot h_j$. Finally, the hidden state of the RNN is updated with the same weighted combination of the h_j:

$$s_i = g(s_{i-1}, c_i) \quad c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \qquad (3)$$

where g is the decoder RNN, s_i is the current hidden state, and c_i can be thought of as the information in the attended words that can be used to determine what to attend to on the next time step. Again, in all experiments an LSTM was used.

3 Experiments

3.1 SCAN dataset

The SCAN[1] dataset was specifically designed to test compositional generalization (details can be found in the appendix, or in Lake and Baroni, 2018). It is composed of 20,910 sequences of commands that must be mapped to sequences of actions, and is generated from a simple finite phrase-structure grammar that includes things like adverbs and conjunctions. The splits of the dataset include: 1) Simple split, where training and testing data are split randomly, 2) Length split, where training includes only shorter sequences, and 3) Add primitive split, where a primitive command (e.g. "turn left" or "jump") is held out of the training set, except in its most basic form (e.g. " jump" $\rightarrow$ JUMP)

Here we focus on the most difficult problem in the SCAN dataset, the add-jump split, where "jump" is held out of the training set.

3.2 Implementation details

Experimental procedure is described in detail in the appendix. Training and testing sets were kept as they were in the original dataset, but following (Bastings et al., 2018), we used early stopping by validating on a 20% held out sample of the training set. All reported results are from runs of 200,000 iterations with a batch size of 1. Unless stated otherwise, each architecture was trained 5 times with different random seeds for initialization, to measure variability in results. All experiments were implemented in PyTorch. Details of the hyperparameter search are given in the appendix. Our best model used LSTMs, with 2 layers and 200 hidden units in the encoder, and 1 layer and 400 hidden units in the decoder, and 120-dimensional vectors for the m_j. The model included a dropout rate of 0.5, and was optimized using an Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001.

3.3 Compositional generalization results

The Syntactic Attention model achieves high compositional generalization performance on the standard seq2seq SCAN dataset (see table 1). The table shows results (mean test accuracy (%) $\pm$ standard deviation) on the test splits of the dataset. Syntactic Attention is compared to the previous models, which were a CNN (Dessì and Baroni, 2019), GRUs augmented with an attention mechanism ("+

[1]The SCAN dataset can be downloaded at https://github.com/brendenlake/SCAN

attn"), which either included or did not include a dependency ("- dep") in the decoder on the previous action (Bastings et al., 2018), and the recent model of Li et al. (2019).

Lake (2019) showed that a meta-learning architecture using an external memory achieves 99.95% accuracy on a meta-seq2seq version of the SCAN task. In this version, models are trained to learn how to generalize compositionally across a number of variants of a compositional seq2seq problem. Here, we focus on the standard seq2seq version, which limits the model to one training episode.

The best model from the hyperparameter search showed strong compositional generalization performance, attaining a mean accuracy of 91.1% (median = 98.5%) on the test set of the add-jump split. However, as in Dessì and Baroni (2019), we found that our model showed variance across initialization seeds (see appendix for details). For this reason, we ran the best model 25 times on the add-jump split to get a more accurate assessment of performance. These results were highly skewed, with a mean accuracy of 78.4% but a median of 91.0% (see appendix for detailed results). Overall, this represents an improvement in the compositional generalization performance compared to the original attention mechanism (Bahdanau et al., 2015; Bastings et al., 2018), and rivals the recent results from Li et al. (2019).

3.4 Additional SCAN experiments

We hypothesized that a key feature of our architecture was that an RNN was used to encode the information in the input sequence relevant to alignment, while one was *not* used to encode the information relevant to translation. To test this hypothesis, we conducted two more experiments:

1. *RNN for translation-encoding*. An additional biLSTM was used to process the input sequence: $m_j = [\overrightarrow{m_j}; \overleftarrow{m_j}]$, where $\overrightarrow{m_j}$ and $\overleftarrow{m_j}$ are the vectors produced for the source word x_j by a biLSTM on the forward and backward passes, respectively. These m_j replace those generated by the simple linear layer in the Syntactic Attention model.

2. c_i *used for translation*. Sequential information from the encoder RNN (i.e. the c_i) was allowed to directly influence the output at each time step in the decoder: $p(y_i|y_1, y_2, ..., y_{i-1}, \mathbf{x}) = f([d_i; c_i])$, where again f is parameterized with a linear function and a softmax output nonlinearity.

The results of the additional experiments (mean test accuracy (%) $\pm$ standard deviations) are shown in table 2. These results partially confirmed our hypothesis: performance on the jump-split test set was worse when encodings from an RNN were directly used for translation. However, when sequential information from the biLSTM encoder was used an additional input in the final production of actions, the model maintained good compositional generalization performance. We hypothesize that this was because in this setup, it was easier for the model to learn to use the m_j to directly translate actions, so it largely ignored the sequential information. This experiment suggests that the factorization between alignment and translation does not have to be perfectly strict, as long as non-sequential representations are available for direct translation.

Additional results, including on other SCAN splits and analyses of the attention distributions, can be found in the appendix.

3.5 Machine translation experiments

Although the purpose of this work was to study the inductive biases that might encourage compositional generalization, we also validated our architecture on a small machine translation dataset to obtain a basic measure of its efficacy in a more naturalistic setting. The dataset (Lake and Baroni, 2018; Bastings et al., 2018) is composed of 10,000 English/French sentence pairs in the training set and 1,190 pairs in the test set. We trained and tested our existing model without making any changes, except for adjusting the learning rate. We also ran the same experiment with the architecture described above that used c_i for translation, as this architecture also showed strong compositional generalization performance on SCAN. BLEU scores on the test set for the best learning rate (0.00015 for both models) are shown in the table below, with comparison to previously reported results using basic recurrent architectures. Our model performs comparably in neural MT, validating it in a more naturalistic setting.

4 Related work

The principle of compositionality has recently regained the attention of deep learning researchers (Bahdanau et al., 2019b,a; Lake et al., 2017; Lake

Model	**Simple**	**Length**	**Add turn left**	**Add jump**
GRU + attn (Bastings et al., 2018)	100.0 ± 0.0	18.1 ± 1.1	59.1 ± 16.8	12.5 ± 6.6
GRU + attn - dep (Bastings et al., 2018)	100.0 ± 0.0	17.8 ± 1.7	90.8 ± 3.6	0.7 ± 0.4
CNN (Dessì and Baroni, 2019)	100.0 ± 0.0	-	-	69.2 ± 8.2
Li et al. (2019)	99.9 ± 0.0	20.3 ± 1.1	99.7 ± 0.4	98.8 ± 1.4
Syntactic Attention (ours)	100.0 ± 0.0	15.2 ± 0.7	99.9 ± 0.16	$91.0^{*} \pm 27.4$

Table 1: Compositional generalization results. The Syntactic Attention model achieves an improvement on the compositional generalization tasks of the SCAN dataset in the standard seq2seq setting, compared to the standard recurrent models (Bastings et al., 2018; Dessì and Baroni, 2019). Star* indicates median of 25 runs.

Model	**Simple**	**Length**	**Add turn left**	**Add jump**
RNN for translation-encoding	99.3 ± 0.7	13.1 ± 2.5	99.4 ± 1.1	42.3 ± 32.7
c_i *used for translation*	99.3 ± 0.85	15.2 ± 1.9	98.2 ± 2.2	88.7 ± 14.2
Syntactic Attention	100.0 ± 0.0	15.2 ± 0.7	99.9 ± 0.16	$91.0^{*} \pm 27.4$

Table 2: Results of additional experiments. Star* indicates median of 25 runs.

Model	En-Fr	Fr-En
LSTM + attn	28.6	-
GRU + attn	32.1	37.5
Syntactic Attention	36.8	35.2
c_i *used for translation*	35.1	33.8

Table 3: Results on small MT dataset (Lake and Baroni, 2018; Bastings et al., 2018).

and Baroni, 2018; Battaglia et al., 2018; Johnson et al., 2017; Keysers et al., 2020) . In particular, the issue has been explored in the visual-question answering (VQA) setting (Andreas et al., 2016; Hudson and Manning, 2018; Johnson et al., 2017; Perez et al., 2018; Hu et al., 2017). Many of the successful models in this setting learn hand-coded operations (Andreas et al., 2016; Hu et al., 2017), use highly specialized components (Hudson and Manning, 2018), or use additional supervision (Hu et al., 2017). In contrast, our model uses standard recurrent networks and simply imposes the additional constraint that mechanisms for alignment and translation are separated. In the Compositional Attention Network, built for VQA, the representations used to encode images and questions are restricted to interact only through attention distributions (Hudson and Manning, 2018). Our model utilizes a similar restriction, reinforcing the idea that compositionality is enhanced when information from different modules are only allowed to interact through discrete probability distributions.

Li et al. (2019) recently showed good performance on the SCAN tasks using a very similar approach. Our results lend additional support to the idea that separating alignment and translation can facilitate compositional generalization. The results from the meta-seq2seq version of the SCAN task (Lake, 2019) suggest that meta-learning may also be a viable approach to inducing compositionality in neural networks.

We were inspired by work in cognitive science emphasizing the relationship between systematicity and syntax (Chomsky, 1957; Fodor and Pylyshyn, 1988). Others have explored similar ideas in different natural language tasks (Bastings et al., 2017, 2019; Chen et al., 2018; Havrylov et al., 2019; Strubell et al., 2018). This work supports the suggestion that intuitions from cognitive science can aid architecture design in deep learning.

5 Conclusion

In this work we attempt to operationalize an intuition from cognitive science, implementing it as inductive bias in the form of a factorization between alignment and translation in the seq2seq setting. We showed that this can improve compositional generalization performance on the SCAN task, and that it doesn't degrade performance on a small MT task. We believe this factorization prevents the model from memorizing spurious correlations in the data, and note that similar ideas may be useful in other natural language tasks.

Acknowledgments

We would like to thank reviewers for their thorough comments and useful suggestions and references.

We would also like to thank all of the members of the Computational Cognitive Neuroscience Lab at UC Davis for helpful ongoing discussion on these topics. This work was supported by ONR N00014-19- 1-2684 / N00014-18-1-2116, ONR N00014-14-1-0670 / N00014-16-1-2128, and ONR N00014-18-C-2067.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural Module Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 39–48, Las Vegas, NV, USA. IEEE.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Dzmitry Bahdanau, Harm de Vries, Timothy J. O'Donnell, Shikhar Murty, Philippe Beaudoin, Yoshua Bengio, and Aaron Courville. 2019a. CLOSURE: Assessing Systematic Generalization of CLEVR Models. *arXiv:1912.05783 [cs]*.

Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron C. Courville. 2019b. Systematic Generalization: What Is Required and Can It Be Learned? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Joost Bastings, Wilker Aziz, Ivan Titov, and Khalil Sima'an. 2019. Modeling Latent Sentence Structure in Neural Machine Translation. *arXiv:1901.06436 [cs]*.

Joost Bastings, Marco Baroni, Jason Weston, Kyunghyun Cho, and Douwe Kiela. 2018. Jump to better conclusions: SCAN both left and right. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 47–55, Brussels, Belgium. Association for Computational Linguistics.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph Convolutional Encoders for Syntax-aware Neural Machine Translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, Copenhagen, Denmark. Association for Computational Linguistics.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261 [cs, stat]*.

Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2018. Syntax-Directed Attention for Neural Machine Translation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 4792–4799. AAAI Press.

N. Chomsky, editor. 1957. *Syntactic Structures*. Mouton & Co., The Hague.

Roberto Dessì and Marco Baroni. 2019. CNNs found to jump around more skillfully than RNNs: Compositional Generalization in Seq2seq Convolutional Networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3919–3923, Florence, Italy. Association for Computational Linguistics.

Jerry A. Fodor and Zenon W. Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

Serhii Havrylov, Germán Kruszewski, and Armand Joulin. 2019. Cooperative Learning of Disjoint Syntax and Semantics. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1118–1128, Minneapolis, Minnesota. Association for Computational Linguistics.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 2017. Learning to Reason: End-to-End Module Networks for Visual Question Answering. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 804–813.

Drew A. Hudson and Christopher D. Manning. 2018. Compositional Attention Networks for Machine Reasoning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. 2017. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1988–1997.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. page 38.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Brenden M Lake. 2019. Compositional generalization through meta sequence-to-sequence learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 9788–9798. Curran Associates, Inc.

Brenden M. Lake and Marco Baroni. 2018. Generalization without Systematicity: On the Compositional Skills of Sequence-to-Sequence Recurrent Networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2879–2888. PMLR.

Brenden M. Lake, Tal Linzen, and Marco Baroni. 2019. Human few-shot learning of compositional instructions. In *Proceedings of the 41th Annual Meeting of the Cognitive Science Society, CogSci 2019: Creativity + Cognition + Computation, Montreal, Canada, July 24-27, 2019*, pages 611–617. cognitivesciencesociety.org.

Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. 2017. Building machines that learn and think like people. *The Behavioral and Brain Sciences*, 40:e253.

Yuanpeng Li, Liang Zhao, Jianyu Wang, and Joel Hestness. 2019. Compositional Generalization for Primitive Substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4293–4302, Hong Kong, China. Association for Computational Linguistics.

João Loula, Marco Baroni, and Brenden Lake. 2018. Rearranging the Familiar: Testing Compositional Generalization in Recurrent Networks. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 108–114, Brussels, Belgium. Association for Computational Linguistics.

Gary Marcus. 2018. Deep learning: A critical appraisal.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. 2018. FiLM: Visual Reasoning with a General Conditioning Layer. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3942–3951. AAAI Press.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

A SCAN dataset details

The SCAN dataset (Lake and Baroni, 2018) is composed of sequences of instructions that must be mapped to sequences of actions (see Figure 3).

The instruction sequences are generated using the pharase-structure grammar described in Figure 4. This simple grammar is not recursive, and so can generate a finite number of command sequences (20,910 total).

These commands are interpreted according to the rules shown in Figure 5. Although the grammar used to generate and interpret the commands is simple compared to any natural language, it captures the basic properties that are important for testing compositionality (e.g. modifying relationships, discrete grammatical roles, etc.). The add-primitive splits (described in main text) are meant to be analogous to the capacity of humans to generalize the usage of a novel verb (e.g. "dax") to many constructions (Lake and Baroni, 2018).

B Experimental procedure details

The cluster used for all experiments consists of 3 nodes, with 68 cores in total (48 times Intel(R) Xeon(R) CPU E5-2650 v4 at 2.20GHz, 20 times Intel(R) Xeon(R) CPU E5-2650 v3 at 2.30GHz), with 128GB of ram each, connected through a 56Gbit infiniband network. It has 8 pascal Titan X GPUs and runs Ubuntu 16.04.

All experiments were conducted with the SCAN dataset as it was originally published (Lake and Baroni, 2018). No data were excluded, and no preprocessing was done except to encode words in the input and action sequences into one-hot vectors, and to add special tokens for start-of-sequence and end-of-sequence tokens. Train and test sets were

jump	⇒	JUMP
jump left	⇒	LTURN JUMP
jump around right	⇒	RTURN JUMP RTURN JUMP RTURN JUMP RTURN JUMP
turn left twice	⇒	LTURN LTURN
jump thrice	⇒	JUMP JUMP JUMP
jump opposite left and walk thrice	⇒	LTURN LTURN JUMP WALK WALK WALK
jump opposite left after walk around left	⇒	LTURN WALK LTURN WALK LTURN WALK LTURN WALK LTURN LTURN JUMP

Figure 3: Examples from the SCAN dataset. Figure reproduced from (Lake and Baroni, 2018).

C → S and S	V → D[1] opposite D[2]	D → turn left
C → S after S	V → D[1] around D[2]	D → turn right
C → S	V → D	U → walk
S → V twice	V → U	U → look
S → V thrice	D → U left	U → run
S → V	D → U right	U → jump

Figure 4: Phrase-structure grammar used to generate SCAN dataset. Figure reproduced from (Lake and Baroni, 2018).

kept as they were in the original dataset, but following (Bastings et al., 2018), we used early stopping by validating on a 20% held out sample of the training set. All reported results are from runs of 200,000 iterations with a batch size of 1. Except for the additional batch of 25 runs for the add-jump split, each architecture was trained 5 times with different random seeds for initialization, to measure variability in results. All experiments were implemented in PyTorch.

Initial experimentation included different implementations of the assumption that syntactic information be separated from semantic information. After the architecture described in the main text showed promising results, a hyperparameter search was conducted to determine optimization (stochastic gradient descent vs. Adam), RNN-type (GRU vs. LSTM), regularizers (dropout, weight decay), and number of layers (1 vs. 2 layers for encoder and decoder RNNs). We found that the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001, two layers in the encoder RNN and 1 layer in the decoder RNN, and dropout worked the best, so all further experiments used these specifications. Then, a grid-search was conducted to find the number of hidden units and dropout rate. We tried hidden dimensions ranging from 50 to 400, and dropout rates ranging from 0.0 to 0.5.

The best model used an LSTM with 2 layers and 200 hidden units in the encoder, and an LSTM with 1 layer and 400 hidden units in the decoder, and used 120-dimensional m_j vectors, and a dropout rate of 0.5. The results for this model are reported in the main text. All additional experiments were done with models derived from this one, with the same hyperparameter settings.

All evaluation runs are reported in the main text: for each evaluation except for the add-jump split, models were trained 5 times with different random seeds, and performance was measured with means and standard deviations of accuracy. For the add-jump split, we included 25 runs to get a more accurate assessment of performance. This revealed a strong skew in the distribution of results, so we included the median as the main measure of performance. Occasionally, the model did not train at all due to an unknown error (possibly very poor random initialization, high learning rate or numerical error). For this reason, we excluded runs in which training accuracy did not get above 10%. No other runs were excluded.

C Skew of add-jump results

As mentioned in the results section of the main text, we found that test accuracy on the add-jump split was variable and highly skewed. Figure 6 shows a histogram of these results (proportion correct). The model performs near-perfectly most of the time, but is also prone to catastrophic failures. This may be because, at least for our model, the add-jump

⟦walk ⟧ = WALK
⟦look⟧ = LOOK
⟦run⟧ = RUN
⟦jump⟧ = JUMP
⟦turn left⟧ = LTURN
⟦turn right⟧ = RTURN
⟦u left⟧ = LTURN ⟦u⟧
⟦u right⟧ = RTURN ⟦u⟧
⟦turn opposite left⟧ = LTURN LTURN
⟦turn opposite right⟧ = RTURN RTURN

⟦u opposite left⟧ = ⟦turn opposite left⟧ ⟦u⟧
⟦u opposite right⟧ = ⟦turn opposite right⟧ ⟦u⟧
⟦turn around left⟧ = LTURN LTURN LTURN LTURN
⟦turn around right⟧ = RTURN RTURN RTURN RTURN
⟦u around left⟧ = LTURN ⟦u⟧ LTURN ⟦u⟧ LTURN ⟦u⟧ LTURN ⟦u⟧
⟦u around right⟧ = RTURN ⟦u⟧ RTURN ⟦u⟧ RTURN ⟦u⟧ RTURN ⟦u⟧
⟦x twice⟧ = ⟦x⟧ ⟦x⟧
⟦x thrice⟧ = ⟦x⟧ ⟦x⟧ ⟦x⟧
⟦x_1 and x_2⟧ = ⟦x_1⟧ ⟦x_2⟧
⟦x_1 after x_2⟧ = ⟦x_2⟧ ⟦x_1⟧

Figure 5: Rules for interpreting command sequences to generate actions in SCAN dataset. Figure reproduced from (Lake and Baroni, 2018).

split represents a highly nonlinear problem in the sense that slight differences in the way the primitive verb "jump" is encoded during training can have huge differences for how the model performs on more complicated constructions. We recommend that future experiments with this kind of compositional generalization problem take note of this phenomenon, and conduct especially comprehensive analyses of variability in results. Future research will also be needed to better understand the factors that determine this variability, and whether it can be overcome with other priors or regularization techniques.

D Supplementary experiments

D.1 Testing nonlinear translation

Our main hypothesis is that the separation between sequential information used for alignment and information about the meanings of individual words encourages systematicity. The results reported in the main text are largely consistent with this hypothesis, as shown by the performance of the Syntactic Attention model on the compositional generalization tests of the SCAN dataset. However, it is also possible that the simplicity of the translation stream in the model is also important for improving compositional generalization. To test this, we replaced the linear layer in this stream with a nonlinear neural network. From the model description in the main text:

$$p(y_i|y_1, y_2, ..., y_{i-1}, \mathbf{x}) = f(d_i), \qquad (4)$$

In the original model, f was parameterized with a simple linear layer, but here we use a two-layer feedforward network with a ReLU nonlinearity, before a softmax is applied to generate a distribution over the possible actions. We tested this model on the add-primitive splits of the SCAN dataset. The results (mean (%) with standard deviations) are shown in Table 4, with comparison to the baseline Syntactic Attention model.

The results show that this modification did not substantially degrade compositional generalization performance, suggesting that the success of the Syntactic Attention model does not depend on the parameterization of the translation stream with a simple linear function.

D.2 Add-jump split with additional examples

The original SCAN dataset was published with compositional generalization splits that have more than one example of the held-out primitive verb (Lake and Baroni, 2018). The training sets in these splits of the dataset include 1, 2, 4, 8, 16, or 32 random samples of command sequences with the "jump" command, allowing for a more fine-grained measurement of the ability to generalize the usage of a primitive verb from few examples. For each number of "jump" commands included in the training set, five different random samples were taken to capture any variance in results due to the selection of particular commands to train on.

Lake and Baroni (2018) found that their best model (an LSTM without an attention mechansim) did not generalize well (below 39%), even when it was trained on 8 random examples that included the "jump" command, but that the addition of further examples to the training set improved performance. Subsequent work showed better performance at lower numbers of "jump" examples, with GRU's augmented with an attention mechanism ("+ attn"), and either with or without a dependence in the decoder on the previous target ("- dep") (Bastings et al., 2018). Here, we compare the Syntactic Attention model to these results.

The Syntactic Attention model shows a substantial improvement over these previous approaches

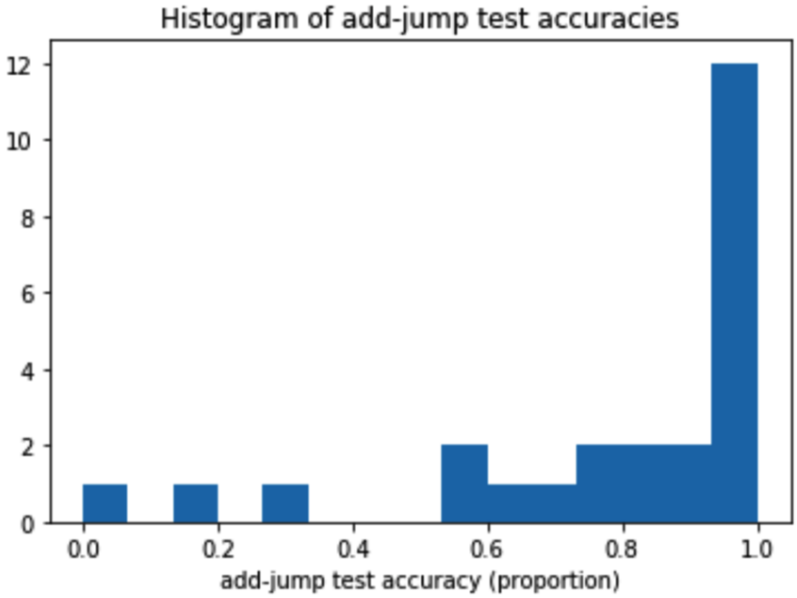

Figure 6: Histogram of test accuracies across all 25 runs of add-jump split.

Table 4: Results of nonlinear translation experiment. Star* indicates median of 25 runs.

Model	**Add turn left**	**Add jump**
Nonlinear translation	99.0 ± 1.7	84.4 ± 14.1
Syntactic Attention	99.9 ± 0.16	$91.0^{*} \pm 27.4$

at the lowest numbers of "jump" examples used for training (see Figure 7 and Table 5). Compositional generalization performance is already quite high at 1 example, and at 2 examples is almost perfect (99.997% correct).

D.3 Template splits

The compositional generalization splits of the SCAN dataset were originally designed to test for the ability to generalize known primitive verbs to valid unseen constructions (Lake and Baroni, 2018). Further work with SCAN augmented this set of tests to include compositional generalization based not on known verbs but on known *templates* (Loula et al., 2018). These template splits included the following (see Figure 8 for examples):

- *Jump around right*: All command sequences with the phrase "jump around right" are held out of the training set and subsequently tested.
- Primitive *right*: All command sequences containing primitive verbs modified by "right" are held out of the training set and subsequently tested.
- Primitive *opposite right*: All command sequences containing primitive verbs modified by "opposite right" are held out of the training set and subsequently tested.
- Primitive *around right*: All command sequences containing primitive verbs modified by "around right" are held out of the training set and subsequently tested.

Results of the Syntactic Attention model on these template splits are compared to those originally published (Loula et al., 2018) in Table 6. The model, like the one reported in (Loula et al., 2018), performs well on the *jump around right* split, consistent with the idea that this task does not present a problem for neural networks. The rest of the results are mixed: Syntactic Attention shows good compositional generalization performance on the Primitive *right* split, but fails on the Primitive *opposite right* and Primitive *around right* splits. All of the template tasks require models to generalize based on the symmetry between "left" and "right" in the dataset. However, in the *opposite right* and *around right* splits, this symmetry is substantially violated, as one of the two prepositional phrases in which they can occur is never seen with "right."

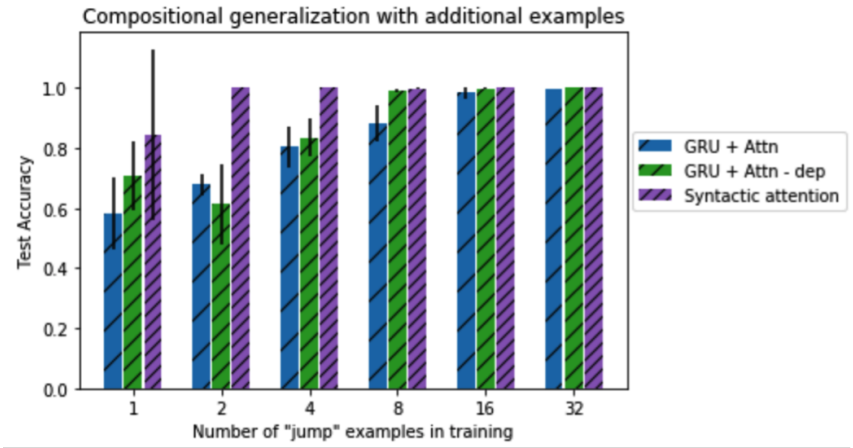

Figure 7: Compositional generalization performance on add-jump split with additional examples. Syntactic Attention model is compared to previously reported models (Bastings et al., 2018) on test accuracy as command sequences with "jump" are added to the training set. Mean accuracy (proportion correct) was computed with 5 different random samples of "jump" commands. Error bars represent standard deviations.

Table 5: Results of Syntactic Attention compared to models of Bastings et al. (2018) on jump-split with additional examples. Mean accuracy (% - rounded to tenths) is shown with standard deviations. Same data as depicted in Figure 7.

	Number of jump commands in training set					
Model	1	2	4	8	16	32
GRU + attn	$58.2_{\pm 12.0}$	$67.8_{\pm 3.4}$	$80.3_{\pm 7.0}$	$88.0_{\pm 6.0}$	$98.3_{\pm 1.8}$	$99.6_{\pm 0.2}$
GRU + attn - dep	$70.9_{\pm 11.5}$	$61.3_{\pm 13.5}$	$83.5_{\pm 6.1}$	$99.0_{\pm 0.4}$	$99.7_{\pm 0.2}$	$100.0_{\pm 0.0}$
Syntactic Attention	$84.4_{\pm 28.5}$	$100.0_{\pm 0.01}$	$100.0_{\pm 0.02}$	$99.9_{\pm 0.2}$	$100.0_{\pm 0.01}$	$99.9_{\pm 0.2}$

E Visualizing attention

Here, we visualize the attention distributions over the words in the command sequence at each step during the decoding process. In the following figures (Figures 9 to 14), the attention weights on each command (in the columns of the image) is shown for each of the model's outputs (in the rows of the image) for some illustrative examples. Darker blue indicates a higher weight. The examples are shown in pairs for a model trained and tested on the add-jump split, with one example drawn from the training set and a corresponding example drawn from the test set. Examples are shown in increasing complexity, with a failure mode depicted in Figure 14.

In general, it can be seen that although the attention distributions on the test examples are not exactly the same as those from the corresponding training examples, they are usually good enough for the model to produce the correct action sequence. This shows the model's ability to apply the same syntactic rules it learned on the other verbs to the novel verb "jump." In the example shown in Figure 14, the model fails to attend to the correct sequence of commands, resulting in an error.

Condition	Example train commands	Example test commands
jump around right	"*jump left*", "*jump around left*", "*walk around right*"	"*jump around right*", "*jump around right and walk*"
Primitive *right*	"*jump left*", "*walk around right*"	"*jump right*", "*walk right*"
Primitive *opposite right*	"*jump left*", "*jump opposite left*", "*walk right*"	"*jump opposite right*", "*walk opposite right*"
Primitive *around right*	"*jump left*", "*jump around left*", "*walk right*"	"*jump around right*", "*walk around right*"

Figure 8: Table of example command sequences for each template split. Reproduced from (Loula et al., 2018)

.

Table 6: Results of Syntactic Attention compared to models of Loula et al. (2018) on template splits of SCAN dataset. Mean accuracy (%) is shown with standard deviations. **P** = Primitive

Model	Template split			
	jump around right	**P** *right*	**P** *opposite right*	**P** *around right*
LSTM (Loula et al. (2018))	98.43±0.54	23.49±8.09	47.62±17.72	2.46±2.68
Syntactic Attention	98.9±2.3	99.1±1.8	10.5±8.8	28.9±34.8

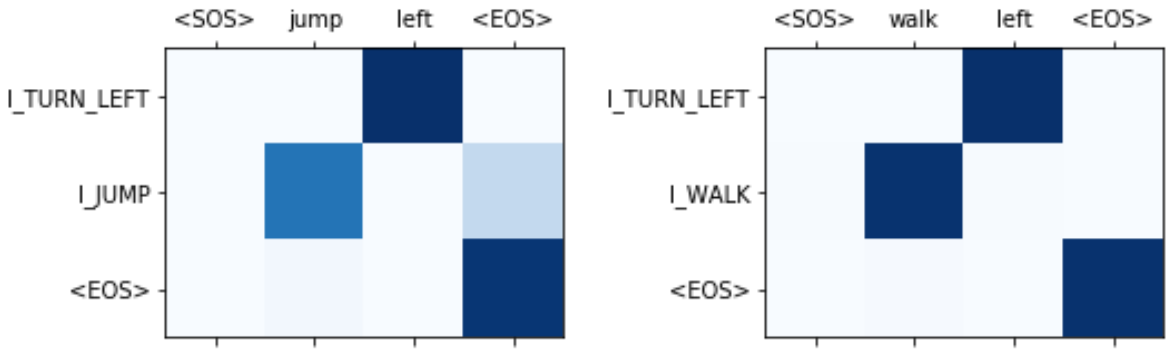

Figure 9: Attention distributions: correct example

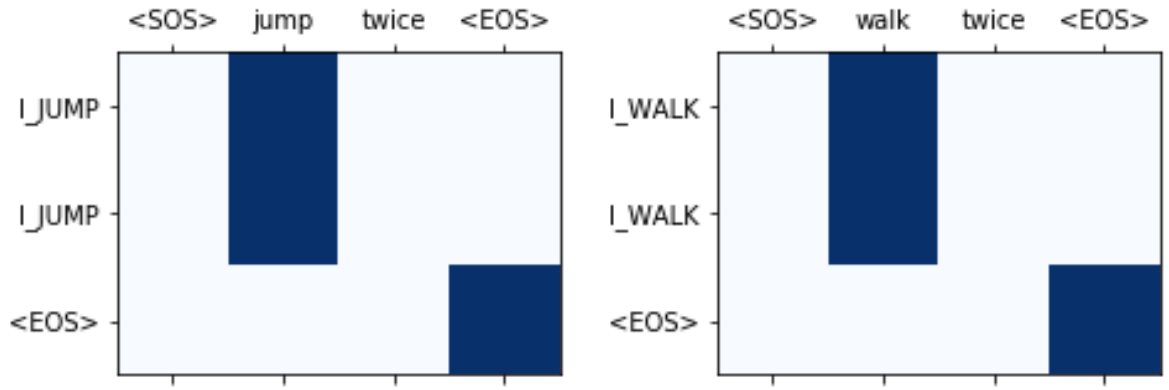

Figure 10: Attention distributions: correct example

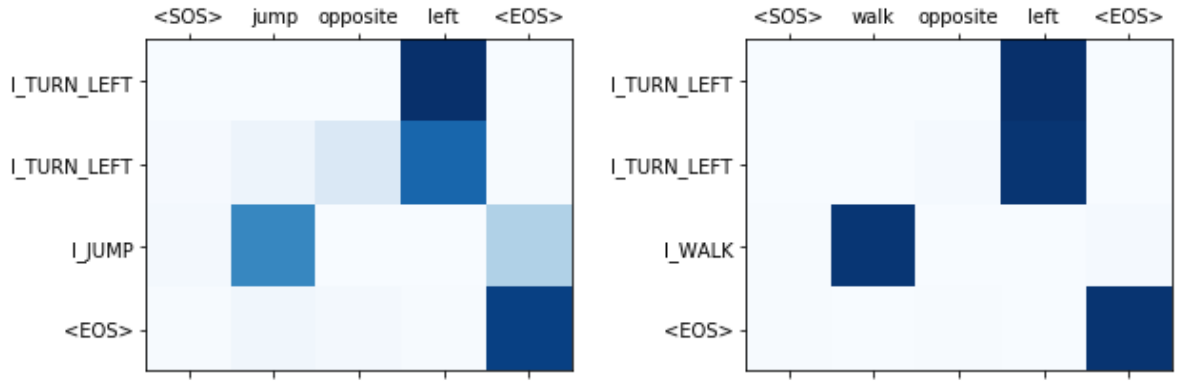

Figure 11: Attention distributions: correct example

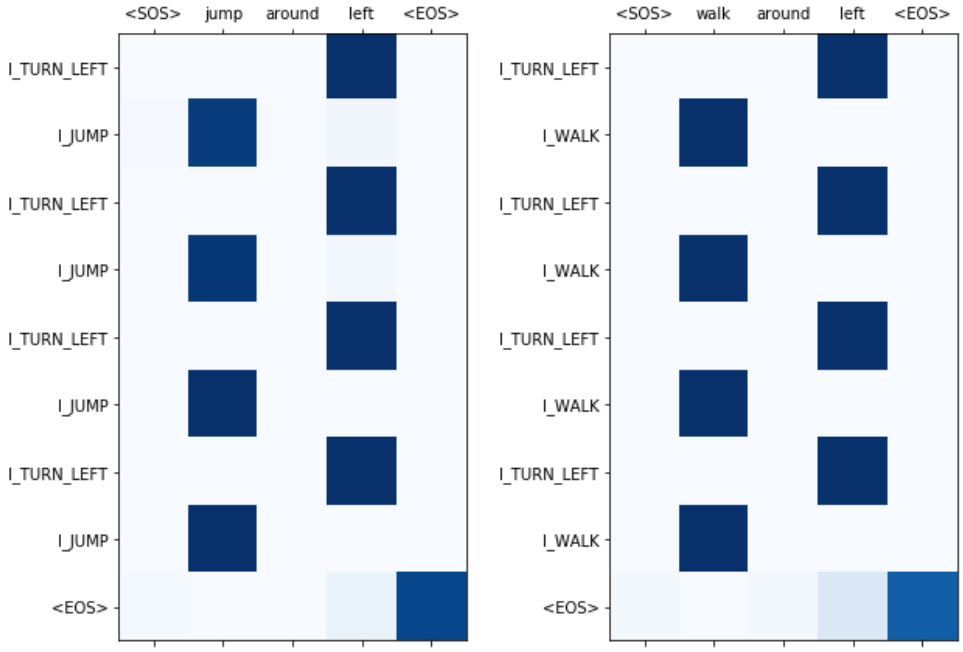

Figure 12: Attention distributions: correct example

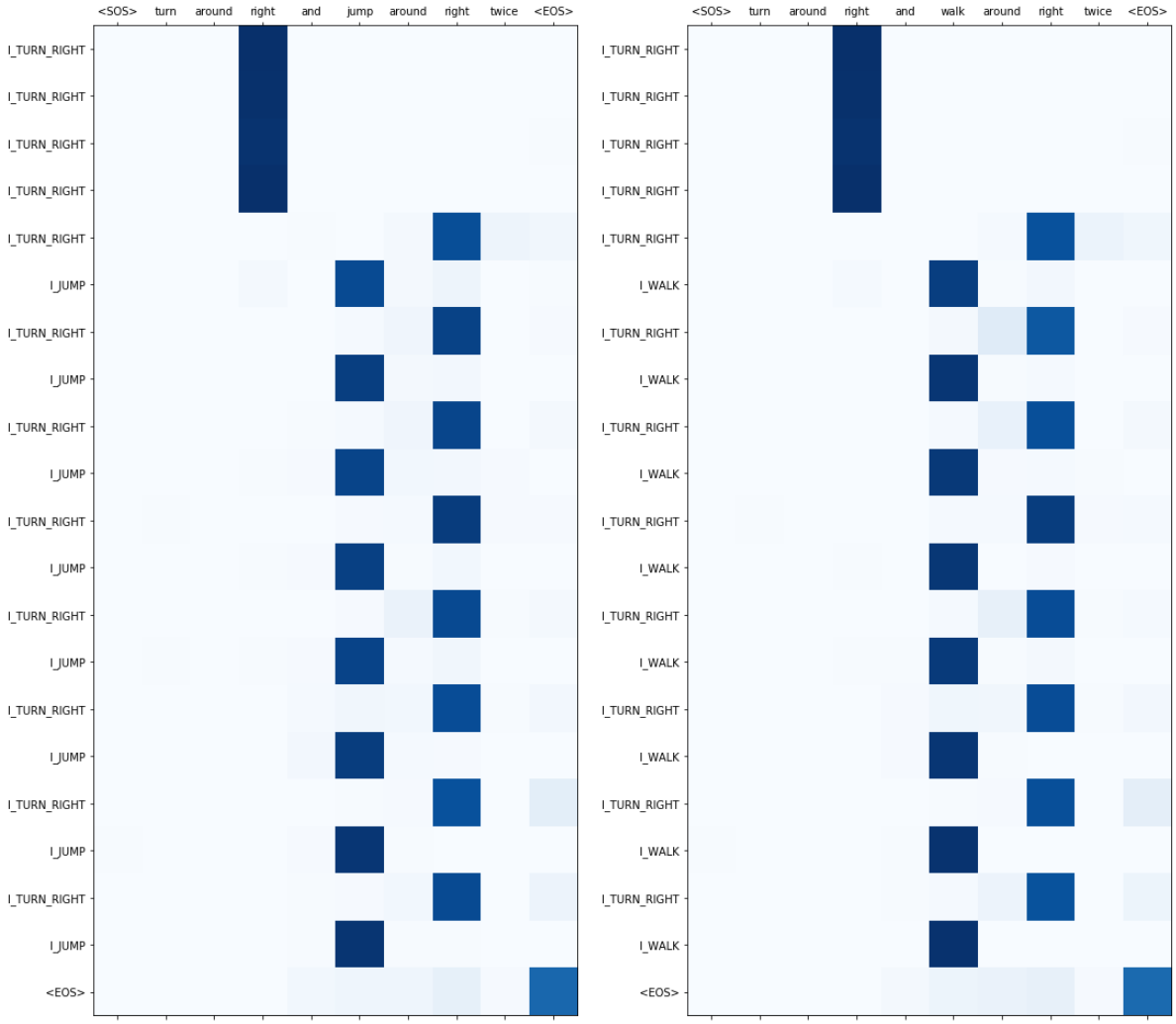

Figure 13: Attention distributions: correct example

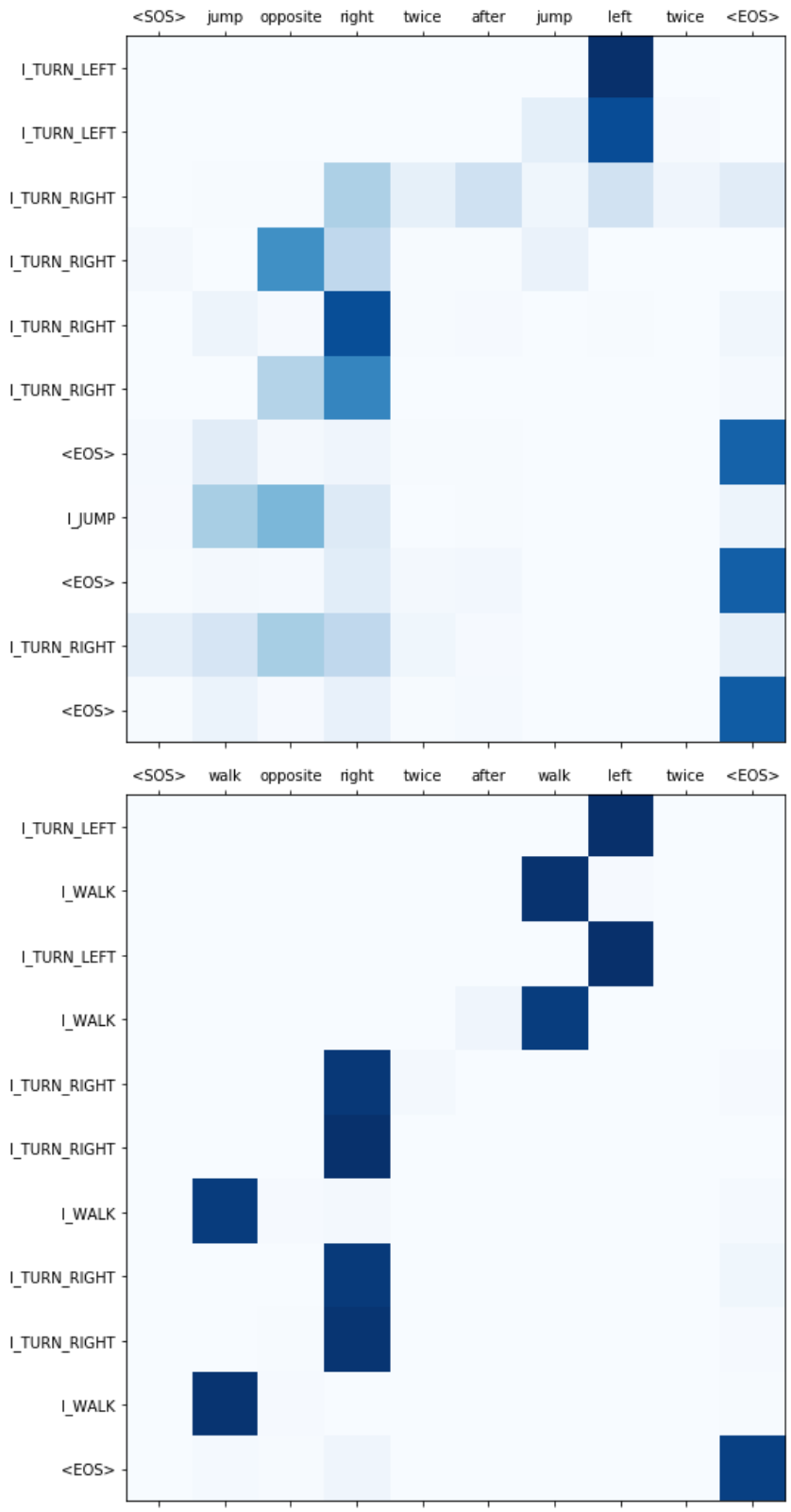

Figure 14: Attention distributions: incorrect example