Transcoding compositionally: using attention to find more generalizable solutions

Korrel, K.; Hupkes, D.; Dankers, V.; Bruni, E.

Transcoding compositionally: using attention to find more generalizable solutions

Kris Korrel
University of Amsterdam
`kris.korrel@gmail.com`

Dieuwke Hupkes
University of Amsterdam
`d.hupkes@uva.nl`

Verna Dankers
University of Amsterdam
`verna.dankers@gmail.com`

Elia Bruni
Universitat Pompeu Fabra
`elia.bruni@gmail.com`

Abstract

While sequence-to-sequence models have shown remarkable generalization power across several natural language tasks, their construct of solutions are argued to be less compositional than human-like generalization. In this paper, we present seq2attn, a new architecture that is specifically designed to exploit attention to find compositional patterns in the input. In seq2attn, the two standard components of an encoder-decoder model are connected via a *transcoder*, that modulates the information flow between them. We show that seq2attn can successfully generalize, without requiring any additional supervision, on two tasks which are specifically constructed to challenge the compositional skills of neural networks. The solutions found by the model are highly interpretable, allowing easy analysis of both the types of solutions that are found and potential causes for mistakes. We exploit this opportunity to introduce a new paradigm to test compositionality that studies the extent to which a model *overgeneralizes* when confronted with exceptions. We show that seq2attn exhibits such overgeneralization to a larger degree than a standard sequence-to-sequence model.

1 Introduction

In recent years, deep artificial neural networks have been at the root of many successes in a wide variety of AI tasks, including sequential tasks, for which encoder-decoder models are the de facto standard (Cho et al., 2014; Sutskever et al., 2014). These successes have also caused a renewed interest in the types of solutions that they learn (Linzen et al., 2018) and, in particular, have prompted the question: to what extent can their high accuracy be taken as evidence that they in fact understood the task they are modeling. A number of recent studies argues that it cannot, when 'understanding the task' is explained as understanding the im-

plicit rules by which it is governed (e.g., Johnson et al., 2017; Lake and Baroni, 2018; Liška et al., 2018; Feng et al., 2018; Ravfogel et al., 2018). More specifically, they argue that rather than understanding those implicit rules and being able to compositionally apply them, RNN models exploit biases in the data that are unrelated to the underlying system. While the latter strategy is remarkably effective when large amounts of training data are available, the lack of understanding of the actual task leads to sample inefficiency, inability to transfer knowledge between tasks and difficulty to generalize to sequences that are drawn from the same rule space, but differ distributionally from the training data. Furthermore, the use of such strategies, which deviate largely from human approaches, that are typically compositional (Lake et al., 2015), makes it difficult to understand what a model does and when it may make a mistake.

In this work, we propose a new component that aims to address this particular weakness of seq2seq models. This component, which is a recurrent attention module that can be integrated in any form of encoder-decoder model, modulates the information flow from encoder to decoder. We test our module, which we dub *seq2attn*, in a recurrent encoder-decoder model. Using two tasks that are designed such that their accuracy reflects directly whether the underlying rule-based system is learned – the lookup table task (Liška et al., 2018) and SCAN (Lake and Baroni, 2018; Loula et al., 2018) – we show that seq2attn strongly encourages rule-based behaviour, which is easily interpreted by studying the attention patterns generated by the module. Additionally, we propose a new testing paradigm based on *overgeneralization*, that can be used to gain more insights in the biases of a model which cannot be inferred from task success alone.

2 Related Work

2.1 Compositional datasets

The ability to learn and compositionally apply symbolic rules is considered to be an important prerequisite for understanding and modeling natural language. While (gated) recurrent neural networks are in principle capable of modeling compositional systems (e.g., Gers and Schmidhuber, 2001; Rodriguez, 2001), whether they in fact do so when trained on large amounts of data to perform natural language processing tasks remains an open question. Some positive results in this direction have been presented (e.g., Hupkes et al., 2018b), but a number of recent papers have argued that, rather than understanding the underlying compositional structure of a problem, RNNs rely on heuristics and exploit biases in the data. Particularly relevant to the current work are the studies of Lake and Baroni (2018) and Liška et al. (2018), who both present data sets specifically designed to reflect compositionality in their task accuracy. Using their compositional tests, they show that vanilla seq2seq models do not readily generalize to solutions that exhibit an understanding of the underlying rule system of the tasks.

2.2 Models

Some recent approaches attack the lack of compositional behaviour of RNNs by designing models that have compositionality explicitly built in, for instance by equiping architectures with a series of specialized modules and a controller that composes them (e.g., Andreas et al., 2016; Johnson et al., 2017). In this work, instead, we focus on inducing compositional solutions in RNN models, that are less rigid and generally require fewer supervision.

Our method draws inspiration from the work on compositional learning of Hupkes et al. (2018a). The authors introduce the concept of *Attentive Guidance*, a training signal given to the attention mechanism of a seq2seq model to induce more compositional solutions. While they convincingly show that seq2seq models with attention can in fact implement such solutions (see Baan et al. (2019) for an in-depth analysis), their model requires attention annotation of the training data, which may not always be available. In this work, we address this problem by designing a model that still aims to be compositional through the attention mechanism, but instead learns these patterns

fully automatically, obtaining similar or even improved performance without the need of extra supervision.

Another line of work which exploits attention as a regularization technique is proposed by Hudson and Manning (2018), who introduce the Memory, Attention and Composition (MAC) cell. The MAC cell consists of three components, whose communication within one cell is restricted to using attention. An important limitation of the MAC cell is that the number of reasoning steps needs to be specified in advance. Our model, as vanilla seq2seq models, doesn't suffer from this limitation.

3 Model

We propose *seq2attn*, a novel attention-centric module that connects the encoder and decoder of a seq2seq model.[1] The core component of seq2attn is the transcoder: a recurrent module that modulates the information flow between encoder and decoder by generating *sparse* attention vectors using separate *keys* and *values*. Below, we demonstrate and test how seq2attn can be used in combination with a vanilla encoder-decoder architecture.

3.1 Encoder

In our tests, we assume a standard recurrent encoder, that, given an input sequence $\{x_1, \ldots, x_N\}$ and an embedding layer $\mathcal{E}^{enc}$, generates a sequence of outputs and hidden states:

$$\mathbf{x}_t^{enc} = \mathcal{E}^{enc}(x_t) \tag{1}$$
$$\mathbf{y}_t^{enc}, \mathbf{h}_t^{enc} = \mathcal{S}^{enc}(\mathbf{x}_t^{enc}, \mathbf{h}_{t-1}^{enc}) \tag{2}$$

$\mathcal{S}$ is a recurrent state transition model, such as a vanilla RNN, LSTM or GRU.

3.2 Transcoder

The transcoder is initialized with $\mathbf{h}_0^{trans} = \mathbf{h}_N^{enc}$ and uses the hidden states of the encoder to compute context vectors $\mathbf{c}_t$ that will be passed to the decoder.

The input to the transcoder is the embedded output of te decoder (Eq. 11):

$$\mathbf{x}_t^{trans} = \mathcal{E}^{trans}(\hat{y}_{t-1}) \tag{3}$$
$$\mathbf{y}_t^{trans}, \mathbf{h}_t^{trans} = \mathcal{S}^{trans}(\mathbf{x}_t^{trans}, \mathbf{h}_{t-1}^{trans}) \tag{4}$$

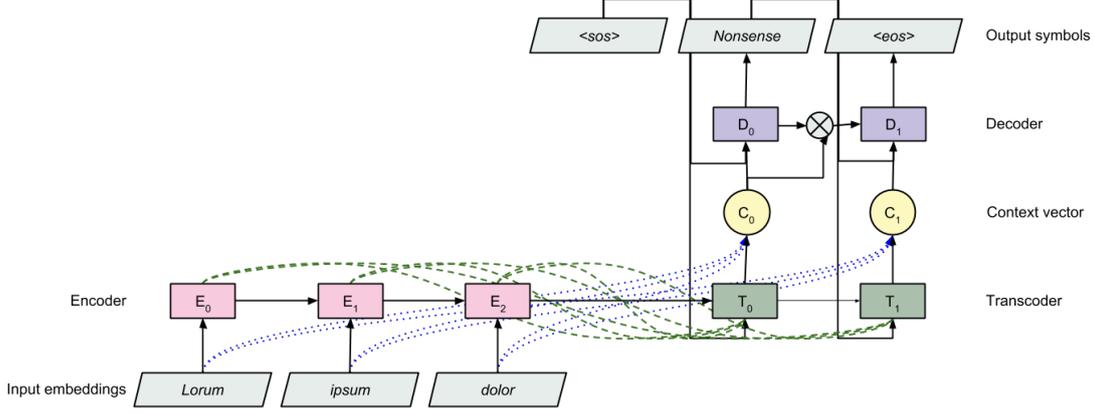[1]We will make our code available upon publication.

Figure 1: Schematic of the seq2attn architecture. The input sequence is processed by the encoder (E), after which the transcoder (T) generates context vectors, which are weighted means over the input embeddings.

The transcoder state is then used to query the hidden states of the encoder. The resulting scores are normalized using the Softmax function:

$$\alpha_t(s) = \boldsymbol{v}_a^\top \cdot ReLU(\boldsymbol{W}_a \cdot [\mathbf{h}_t^{enc}; \mathbf{h}_{t-1}^{trans}]) \quad (5)$$

$$\pi_t(s) = \frac{\exp \alpha_t(s)}{\sum_{i=1}^{N} \exp \alpha_t(i)} \quad (6)$$

Using the Softmax distribution often results in distributed vectors that attend to many input symbols at the same time, while an ideal compositional attention vector only focuses on the relevant parts of the input. To force the transcoder to be more selective in the information it selects, we use Gumbel-Softmax, which allows us to draw from the categorical distribution computed in Eq. 6, with continuous relaxation (Jang et al., 2017; Maddison et al., 2016). The Straight-Through estimator is then used as a biased gradient estimator of the $\arg\max$ operator:

$$\tilde{\mathbf{a}}_t(s) = \frac{\exp \frac{\log \pi_t(s) + g_s}{\tau}}{\sum_{i=1}^{N} \exp \frac{\log \pi_t(i) + g_i}{\tau}} \quad (7)$$

$$\mathbf{a}_t = \tilde{\mathbf{a}}_t - \hat{\mathbf{a}}_t + \text{one\_hot}(\arg\max(\tilde{\mathbf{a}}_t)) \quad (8)$$

The temperature τ can be interpreted as a measure of uncertainty. $\hat{\mathbf{a}}_t$ is a copy of $\tilde{\mathbf{a}}_t$ which we do not backpropagate through. At inference time the stochasticity of Gumbel-Softmax is not needed, and $\arg\max$ is used as activation function.

The resulting attention weights are used to compute the context vectors that will be passed to the decoder:

$$\mathbf{c}_t = \sum_{i=1}^{N} \mathbf{a}_t(i) \cdot \mathbf{x}_i^{enc} \quad (9)$$

Crucially, the context vector represents a weighted average of the *input embeddings* of the encoder, while the weights $\mathbf{a}_t(i)$ are depending on the *hidden states* of the encoder, thus introducing a separation between attention *keys* and *values* (similar to, e.g., Mino et al., 2017; Vaswani et al., 2017).

3.3 Decoder

The decoder of a seq2seq model is commonly initialized with the final hidden state of the encoder. However, as this state vector encodes the entire input sequence, this type of initialization does not urge compositional behavior of the decoder. When seq2attn is used, the decoder should be initialized with a fixed, *learned* initialization vector. In combination with using input embeddings as attention values (Eq. 9), this restricts the decoder to work only with disentangled representations of the input sequence, which encourages it to learn and process the individual meaning of all input symbols.

To model outputs, the decoder uses the context vector $\mathbf{c}_t$, its own embedded output (identical to Eq. 3) and a vector $\mathbf{h}_{t-1}^{dec}$ that integrates the current decoder hidden state with the context vector:

$$\mathbf{y}_t^{dec}, \tilde{\mathbf{h}}_t^{dec} = \mathcal{S}^{dec}([\mathbf{c}_t; \mathcal{E}^{trans}(\hat{y}_{t-1})], \mathbf{h}_{t-1}^{dec}) \quad (10)$$

$$\hat{y}_t = \arg\max(\text{Softmax}(\mathbf{y}_t^{dec})) \quad (11)$$

Where $\mathbf{h}_{t-1}^{dec}$ is computed using an element-wise multiplication of the context vector with the previous hidden state of the decoder:

$$\mathbf{h}_{t-1}^{dec} = \tilde{\mathbf{h}}_{t-1}^{dec} \odot \mathbf{c}_t \quad (12)$$

This way of integrating the context vector with the decoder, which we call *full focus*, makes the output of the decoder at decoding step t more directly dependent on the current context vector c_t.

3

	held-out inputs	held-out compositions	held-out tables
Baseline	38.25 ± 0.04	43.28 ± 0.09	7.86 ± 0.02
Seq2attn	$\mathbf{100 \pm 0.00}$	$\mathbf{100 \pm 0.00}$	$\mathbf{100 \pm 0.00}$

Table 1: Average sequence accuracies and standard deviations of the baseline and seq2attn models on all lookup tables test sets.

4 Test Case 1: Lookup tables

Our first test-case is the lookup table task introduced by Liška et al. (2018).

4.1 Task

The core of the *lookup table composition* domain consists in sequentially applying simple lookup table functions. The functions to be applied are bijective mappings from the set of all n-bit bitstrings onto itself. Following Liška et al. (2018), we focus on 3-bit strings, resulting in $2^3 = 8$ possible inputs and outputs. We create 8 random table lookup functions, to which we refer with the names t1, t2, ..., t8. Given the simplicity of the functions, the main challenge of the task resides in inferring that the input sequences should be treated *compositionally*, rather than considered as a whole.

We borrow the setup presented in Hupkes et al. (2018a), which differs slightly from the setup as it was originally presented. In this setup, a typical input output example could be 001 t1 t2 → 001 010 111. Computing the output for this example requires the sequential application of t1 to 001, and then t2 to the intermediate result. Since two tables are to be applied in succession, we refer to such an examples as a *binary composition*, as opposed to a *unary composition* in which only one function has to be applied on the input. The input bitstring and all intermediate outputs are included in the target output sequence.

Liška et al. (2018) train models on all 8 inputs for unary compositions and on 6 out of 8 input bitstrings of all binary compositions. The remaining 2 *held-out inputs* are used to test for generalization. Following Hupkes et al. (2018a), we do not include all 64 binary compositions in the training set, but leave out some for testing. In particular, we create one test set that contains all binary compositions containing t7 or t8, which are thus only seen in the training set as unary compositions. We call this condition *held-out tables*. Of the remaining binary compositions, that contain only functions in {t1, ..., t6}, 8 randomly

	held-out inputs	held-out compositions	held-out tables
Baseline+G	34.17 ± 8.25	38.54 ± 12.39	8.16 ± 3.57
Baseline+E	82.50 ± 12.42	85.42 ± 12.39	31.08 ± 7.85
Baseline+F	85.83 ± 16.50	91.67 ± 11.79	30.03 ± 16.12
Baseline+T	43.33 ± 12.30	47.40 ± 15.33	3.99 ± 2.70
Baseline+GE	82.50 ± 12.42	83.85 ± 7.48	30.21 ± 3.32
Baseline+GF	69.17 ± 21.25	76.04 ± 13.28	4.69 ± 1.47
Baseline+GT	32.50 ± 8.90	45.31 ± 10.13	1.56 ± 1.53
Baseline+EF	85.00 ± 9.35	82.29 ± 18.46	24.13 ± 2.99
Baseline+ET	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$	41.49 ± 3.30
Baseline+FT	68.33 ± 21.44	71.88 ± 23.00	19.44 ± 19.06
Baseline+GEF	74.17 ± 36.53	72.40 ± 37.94	37.33 ± 22.10
Baseline+GET	97.50 ± 3.54	98.44 ± 1.28	24.31 ± 17.87
Baseline+GFT	90.83 ± 3.12	91.15 ± 3.21	28.30 ± 7.23
Baseline+EFT	66.67 ± 47.14	66.67 ± 47.14	66.67 ± 47.14
Seq2attn	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$

Table 2: Mean sequence accuracies and standard deviation on the lookup tables task of a baseline seq2seq model with additional components of seq2attn. **G**=Gumbel-Softmax, **E**=embeddings as attention values, **F**=full focus, **T**=transcoder.

selected compositions are held out from the train set for all inputs, which form the *held-out compositions* test set. Lastly, we remove 2 of the 8 inputs for each binary composition independently to form the *held-out inputs* test set, which is similar to the generalization condition presented by Liška et al. (2018).

4.2 Results

We first compare the seq2attn architecture to a standard seq2seq model with an attention mechanism on generalization to new test examples. We establish the optimal parameters for both models using a grid search over a separate validation set. Our search includes the type of RNN cell ({GRU, LSTM}), the embedding and RNN sizes ({32, 64, 128, 256, 512, 1024}) and the dropout rate ({0, 0.2, 0.5}). The results are summarized in Table 3. The mini-batch size (1) and optimizer (Adam with default parameters (Kingma and Ba, 2014)) are fixed. We train 10 models with the optimal parameters and report mean sequence accuracy. For simplicity we will henceforth simply refer to this as the accuracy.

Our experiments confirm the findings previously presented by Hupkes et al. (2018a) and Liška et al. (2018): Vanilla seq2seq models do not find generalizing solutions for the lookup table task (Table 1, first row). Seq2attn, on the other hand, generalizes perfectly to data outside the training distribution. This first test confirms our hypothesized compositional bias of seq2attn.

4.3 Ablation Study

The difference between a traditional seq2seq and the seq2attn model can be summarized as the use of (i) a transcoder, (ii) the Gumbel-Softmax activation for the attention vector, (iii) using input embeddings as attention values and (iv) using full focus. To assess the contributions of these components, we take the seq2attn model with optimal hyper-parameters as a base model, and increasingly ablate components. The results of this study (Table 2) indicate that, while some of the components of seq2attn cause an increase in accuracy on their own, no subset of them can match the performance of the full seq2attn model.

4.4 Attention patterns

As the modeled output of the decoder is highly dictated by the context vectors that it receives, we can gain insights into the types of solutions the models are forming by studying their attention vectors. As illustrated in Figure 2, seq2attn learns to generate a "correct" attention trace, attending to the right input at the right time. Contrastingly, the baseline fails to capture a systematic pattern and produces a diffused attention instead or attends to irrelevant inputs, indicating that it does not utilize the attention mechanism to its full advantage.

4.5 Overgeneralization

The results for the lookup tables task indicate that seq2attn performs much better than the baseline on data containing held-out inputs, tables and compositions. The model is thus better able to infer the compositional rules underlying the data. To further explore seq2attn's bias towards compositionality, we test its behaviour when confronted with *uncompositional* examples, that do not adhere to the previously mentioned rules. Where a model unaware of the underlying task structure would have little problems learning such exceptions – or in fact, would not realise that they are exceptions – a model with a strong compositional bias may sometimes wrongly assume the exceptions also adhere to the underlying system, and *overgeneralize* an inferred rule. The extent to which a model overgeneralizes can thus be seen as a proxy for the strength of its compositional bias. Whether overgeneralization is actually preferentiable behavior is depend on the task to be solved.

In the proposed setup, a small number of training instances are assigned adapted output targets.

We call these instances *exceptions*. The target output sequences of the exceptions are changed such that they can only be learned through memorization. For the lookup table task, we adapt the training set such that one composition, `t1 t2`, is an exception to the general rules for three out of the eight existing input bitstrings. In the target output, the third bitstring is replaced with a randomly selected bitstring, thus changing the application of table `t2` in this context. Both the three adapted samples and the other five unadapted samples for `t1 t2` are included in the training set.

While training a model, we monitor the output sequences generated for these exceptions. The accuracy on the original targets is reported to identify whether the model is processing the exceptions compositionally despite being exposed to the adapted targets in the training set, i.e., whether the model is overgeneralizing.

Figure 3 displays the accuracy on the *original* targets of all eight inputs in composition `t1 t2` over the first 30 training epochs. While both the baseline and seq2attn learn to memorize the three exceptions, only seq2attn shows a strong bias to treat the inputs compositionally before memorizing the adapted targets. The performance goes as high up as $8/8$ between the fifth and fifteenth epoch for differently initialized models, before dropping to $5/8$. This indicates that the rules are learned before the adapted instances are memorized as exceptions to these rules.

5 Test Case 2: SCAN

While the lookup table task provides an excellent test case to evaluate the compositional abilities of a neural network model, its simplicity limits the conclusions that can be drawn about the usability of seq2attn in more challenging domains. In this section, we evaluate seq2attn on *SCAN* (Lake and Baroni, 2018), a task involving mapping navigational commands to sequences of output actions.

5.1 Task

The input commands of the SCAN task are composed of a small set of predefined atomic commands (*jump*, *walk*, *run* and *look*), modifiers (*twice*, *thrice*, *around*, *opposite*, *left* and *right*) and conjunctions (*after* and *and*) that are combined via a limited context free grammar, such that there are no ambiguities. An example input is *jump after walk left twice*, where the learn-
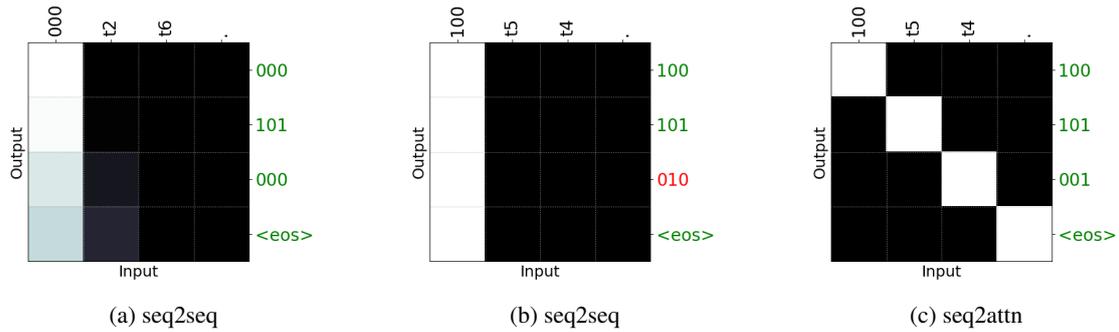
Figure 2: Examples of modeled attention patterns on held-out input examples of the lookup tables domain.

	Baseline	Seq2attn
Lookup tables	128, 512, 1, GRU, 0.5	256, 256, 1, GRU, 0.5
SCAN	200, 200, 2, LSTM, 0.5	512, 512, 1, GRU, 0.5

Table 3: Hyperparameters (embedding dimensions, RNN dimensions, RNN layers, RNN type, dropout rate) used for both the seq2seq baseline and seq2attn model for both tasks.
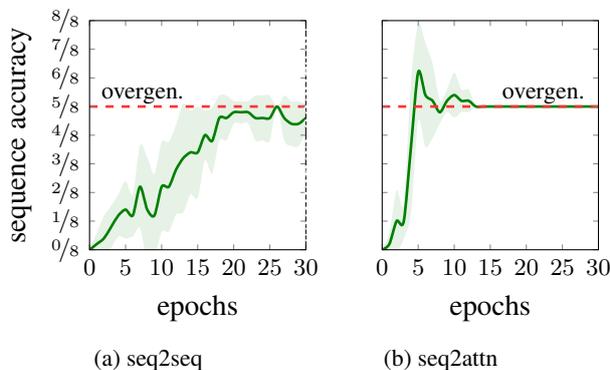

Figure 3: Average accuracies on *original targets* for the eight inputs in composition `t1 t2`. As three of these compositions are exceptions, we refer to accuracy higher than $5/8$ as overgeneralization. The 95% confidence interval is indicated.

ing agent has to mentally perform these actions in a 2-dimensional grid and output the sequence of actions it takes: "I_TURN_LEFT I_WALK I_TURN_LEFT I_WALK I_JUMP". For full details of the data set and experiments, we refer to Lake and Baroni (2018).

Lake and Baroni use three different train-test distributions of the total of 20.910 examples. They show that vanilla seq2seq models are able to almost perfectly generalize when the data is randomly split in a training and testing set, but that they are unfit for generalizing to longer test sequences and for one-shot learning to commands seen only in their atomic form. Later, Loula et al. (2018) proposed a new set of experiments based on the same task, which they argue are better

suited for assessing systematic compositionality. We focus on experiments 2 and 3 of their paper.

Experiment 2 contains four different train-test distributions as there are four primitive commands involved. For all four conditions, the test set is the same. This test set consists of all examples that contain "jump around right" in their input sequences. The first condition, which is called **0 fillers**, contains no subsequences of the form "*primitive* around right" in the training set, where *primitive* is either of the four primitives "jump", "look", "run" or "walk". This condition should thus test whether a model can induce a compositional understanding of "jump around right" by showing those symbols ("jump", "around" and "right") only in different contexts. The next three conditions, **1 filler**, **2 fillers** and **3 fillers**, are considered increasingly easier. They retain the same test set, but increasingly add more examples to the train set of the template "*primitive* around right". **1 filler** adds all examples of this template where *primitive* is "look". **2 fillers** and **3 fillers** add "walk" and "run" respectively.

As Loula et al. (2018) observed a great difference in performance between the **0 fillers** and **1 filler** conditions, they zoom in on these conditions in experiment 3. The **0 fillers** condition contains 0 examples with the subsequence "*primitive* around right" in the training set. The **1 filler** condition contains 1.100 of those, namely all examples which contain the subsequence "look around right". In experiment 3, they test a more smooth and dense transition from the **0 fillers** condition

to the **1 filler** conditions. They accomplish this by taking the training set of the **0 fillers** condition and adding respectively 1, 2, 4, 8, 16, 32, 64, 128, 256, 512 and 1024 extra examples containing the subsequence "look around right", resulting in 11 new training sets. The test set is again the same as in experiment 2.

5.2 Results

We compare a baseline seq2seq to the seq2attn architecture on these two tasks. First, we perform a grid search using a random split of the data to find the optimal parameters for seq2attn. The results of this are summarized in Table 3. As a baseline, we used the model which Lake and Baroni (2018) found to be overall best performing, which is a seq2seq model with 2-layer LSTMs, 200 hidden units per layer and a dropout rate of 0.5. For comparison to the seq2attn model, we also added an attention mechanism, which was missing in the original model. For all reported results we ran these models 10 times with random weight initialization. Since experiments 2 and 3 by Loula et al. (2018) do not have validation sets for early stopping, we ran all models for 50 epochs.

Firstly, we confirm the findings of Lake and Baroni (2018) and Loula et al. (2018) (see Fig. 4, left). A vanilla seq2seq with attention is able to perform analogical generalization (95.19% accuracy): it requires examples of **1 filler** only to generalize to other fillers of the same template. On the other hand, it is not able to apply "right" and "around" to a primitive verb in a productive way, when they were never seen together (0.26% accuracy, **0 fillers** condition). When we look at seq2attn, we notice how not only it is able to perform analogical generalization (94.32% accuracy, **1 filler**) but, to a certain extent, it is also able to generalize productively in the **0 fillers** condition (36.23% accuracy).

In Figure 4 (right) we report the results for experiment 3 of Loula et al. (2018) where we consider the **0 fillers** condition of Experiment 2 and progressively add extra training examples from **1 filler**. As Loula et al. (2018) observed, performance of a seq2seq model ramps up as more samples are injected in the training set. Yet, the fact that performance increases gradually and takes long to peak (at 512 examples) suggests that rather than systematically understanding the rule, the model is piling up evidence for a very spe-

cific pattern. The situation is quite different for the seq2attn model, whose performance spikes much earlier, reaching a plateau at 16 examples already. Interestingly, the performance peak is also at 512, but with an improvement of just over 5 percentage points over 16 examples vs. approximately 50 percentage points improvement in the case of baseline. Seq2attn seems then to show evidence for an opposite interpretation, namely for a network that, to a certain extent, is able to induce the compositional rules. A property that is often linked to sample efficiency (Lake and Baroni, 2018).

5.3 Analysis

In Figure 5, we now look at some attention patterns for the **0 fillers** condition. While baseline models emit sparser and more informative attentional patterns here than in the lookup table task, they still are locally diffused and, more importantly, do not maintain a systematic input-output alignment, which suggests that the models are not understanding the rules of the task, but use a pattern matching strategy instead. On the contrary, seq2attn shows always fully sparse, one-hot attention patterns. Figure 5c shows how the model usually aligns outputs to their respective primitive commands or directions in the input sequence, e.g., "I_JUMP" aligns to "jump", and "I_TURN_RIGHT" aligns to "right". A modifier like "opposite" is used as an indicator to repeat the last modeled directional action.

Seq2attn reaches an accuracy of 36.23% on the **0 fillers** condition, which still leaves room for improvement. However, the attentional patterns quickly show the main cause of error. Figure 5b shows how the model outputs "I_TURN_LEFT" instead of "I_TURN_RIGHT" whenever it attends to the input "around". Whenever the model does attend to "right", as is the expected, optimal behavior, the output is correct. This behavior can be easily explained by analyzing the data that the model was trained on. The input "around" has only been encountered within the context "*primitive* around left" during training. Thus, within this context, "around" and "left" could be used synonymically by the transcoder to communicate to the decoder to output "I_TURN_LEFT". The great majority of errors on this task by seq2attn have the same cause. Although seq2attn still does not perfectly solve the task, contrary to a standard seq2seq model, it provides an immediate under-
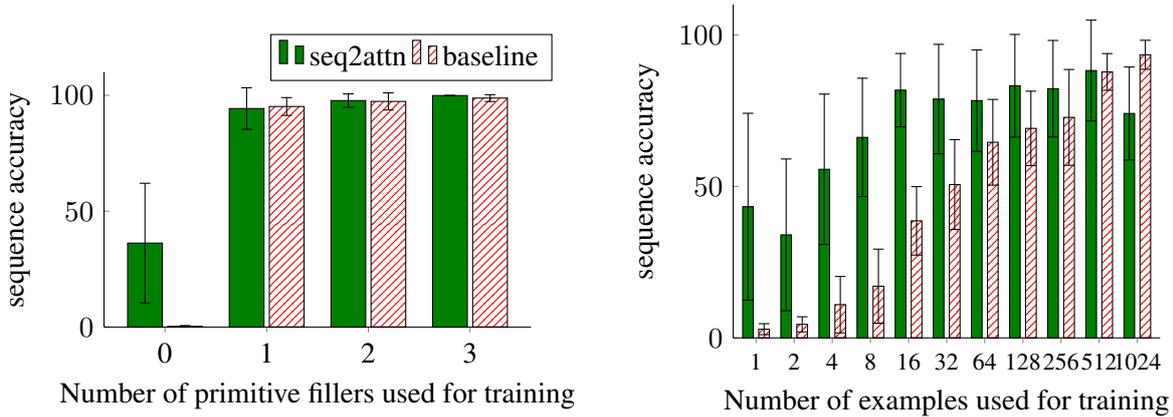
Figure 4: Mean sequence accuracies on experiment 2 (left) and 3 (right) of Loula et al. (2018). The bootstrapped 95% confidence intervals are indicated with error bars.

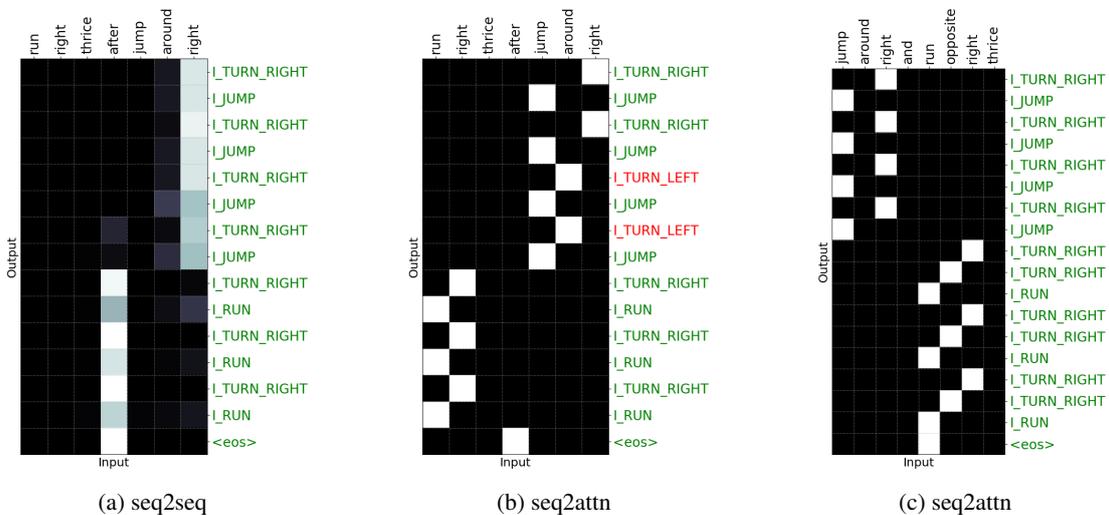
(a) seq2seq (b) seq2attn (c) seq2attn

Figure 5: Examples of attention patterns on the **0 fillers** condition. Seq2attn models the output incorrectly when the attention pattern is incorrect.

standing of the root of this.

5.4 Overgeneralization

To assess seq2attn's overgeneralization abilities for the SCAN task, we repeated experiment 3. In addition to gradually adding samples indicating the correct interpretation of "*primitive* around right", we also added a single exception for "jump around right" to the training set. The target for this sequence, originally consisting of four repetitions of "I_TURN_RIGHT I_JUMP", was modified to consist of only two repetitions.

For all conditions of experiment 3, we added the exception to the training set, trained multiple randomly initialized models, and monitored the output sequences generated for this exception over the course of training. In Figure 6, we visualize the distribution over the adapted and original tar-

gets for the conditions with 4 and 512 filler samples respectively. Note that the models have implicit and explicit evidence for the correct application of the rules for "*primitive* around right": explicit evidence through training examples containing "look around right" subsequences, and implicit evidence through training samples including "around" or "right" seen in different contexts.

Both models exhibit overgeneralization behavior for SCAN. Generally, overgeneralization occurs at the start of the training process and precedes memorization of the adapted target. However, the baseline model needs a substantially larger amount of explicit evidence to overgeneralize as much as the seq2attn model. The condition where 512 filler samples are included illustrates that the tendency to overgeneralize does not necessarily relate to the overall task performance.
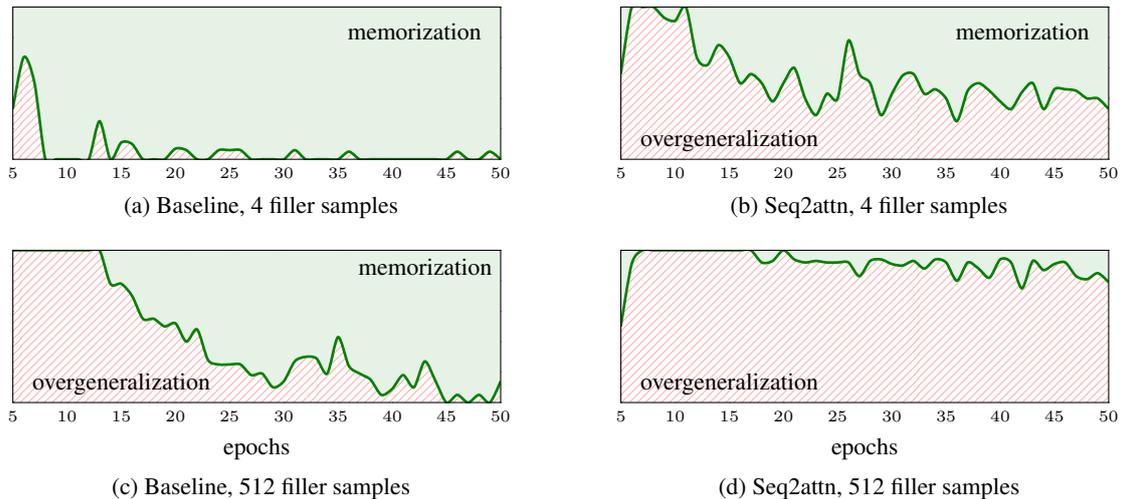
(a) Baseline, 4 filler samples

(b) Seq2attn, 4 filler samples

(c) Baseline, 512 filler samples

(d) Seq2attn, 512 filler samples

Figure 6: Mean sequence accuracy on the *original target* of "jump around right" of multiple models as training progresses. The distribution was normalized for cases in which the output emitted was neither the original or the adapted target.

For this condition, seq2attn and the baseline yield similar sequence accuracies in the original setup of experiment 3 (see Figure 6), but seq2attn overgeneralizes more frequently, indicating that seq2attn has a stronger compositional bias.

6 Discussion

In search for a neural network architecture that exhibits a bias towards systematic generalization, we introduced **seq2attn**, a recurrent attention-centric module that controls the information flow from encoder to decoder. We installed this module in a standard recurrent encoder-decoder architecture. To quantify its capabilities in terms of systematic compositionality, we tested the model on the lookup table and SCAN tasks.

On both tasks, we see significant improvements compared to a standard recurrent seq2seq model, providing evidence for a compositional bias in the system. Furthermore, because the architecture relies heavily on its attention mechanism, its solutions can more easily be interpreted by looking at the generated attention patterns. This provides opportunities for analyzing what the model has learned as well as for detecting potential biases in the training set.

Although on the considered tasks, which are specifically designed to evaluate compositionality, seq2attn leads to clear improvements, its contribution could not have been observed when considering a task for which the test accuracy is not directly linked to compositionality, such as nat-

ural language modeling and translation. We argue that, for those cases, additional assessment methods are needed to compare the compositional skills of different models. We propose one such method, which involves monitoring to what extent a model *overgeneralizes*. We show how a model with seq2attn, for both tasks, has a greater tendency to overgeneralize than the baseline.

A possible limitation of the design of seq2attn is that the flow of information from transcoder to decoder is very rigid. Possible solutions could be found in the use of less skewed activations than the Gumbel-Softmax such as Sparsemax (Martins and Astudillo, 2016), or allowing the transcoder to communicate multiple embeddings using adaptive computation time (Graves, 2016).

Importantly, seq2attn is not tied to a particular type of seq2seq architecture. In future work, we plan to install it into other popular seq2seq architectures such as convolutional seq2seq (Gehring et al., 2017) and Transformer models (Vaswani et al., 2017).

Acknowledgements

References

Jacob Andreas, Dan Klein, and Sergey Levine. 2016. Modular multitask reinforcement learning with policy sketches. *arXiv preprint arXiv:1611.01796*.

Joris Baan, Jana Leible, Nikolaus Mitja, Rau David, Ulmer Dennis, Tim Baumgrtner, Dieuwke Hupkes, and Elia Bruni. 2019. On the realization of compositionality in neural networks. *BlackboxNLP 2019, ACL*.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Fethi Bougares Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 17241734. Association for Computational Linguistics.

Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. 2018. Pathologies of neural models make interpretations difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3719–3728. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Felix A Gers and Jürgen Schmidhuber. 2001. LSTM recurrent networks learn simple context-free and context-sensitive languages. *Neural Networks, IEEE Transactions on*, 12(6):1333–1340.

Alex Graves. 2016. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*.

Drew A Hudson and Christopher D Manning. 2018. Compositional attention networks for machine reasoning. *arXiv preprint arXiv:1803.03067*.

Dieuwke Hupkes, Anand Singh, Kris Korrel, German Kruszewski, and Elia Bruni. 2018a. Learning compositionally through attentive guidance. *arXiv preprint arXiv:1805.09657*.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018b. Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *Proceedings of the International Conference on Learning Representations (ICLR2017)*.

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 *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 1988–1997. IEEE.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2879–2888.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Tal Linzen, Grzegorz Chrupała, and Afra Alishahi, editors. 2018. *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics.

Adam Liška, Germán Kruszewski, and Marco Baroni. 2018. Memorize or generalize? searching for a compositional rnn in a haystack. *arXiv preprint arXiv:1802.06467*.

João Loula, Marco Baroni, and Brenden M Lake. 2018. Rearranging the familiar: Testing compositional generalization in recurrent networks. *arXiv preprint arXiv:1807.07545*.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the International Conference on Learning Representations (ICLR2017)*.

Andre Martins and Ramon Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pages 1614–1623.

Hideya Mino, Masao Utiyama, Eiichiro Sumita, and Takenobu Tokunaga. 2017. Key-value attention mechanism for neural machine translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 290–295.

Shauli Ravfogel, Yoav Goldberg, and Francis Tyers. 2018. Can lstm learn to capture agreement? the case of basque. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 98–107.

Paul Rodriguez. 2001. Simple recurrent networks learn context-free and context-sensitive languages by counting. *Neural computation*, 13(9):2093–118.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.