Compositional generalization through meta sequence-to-sequence learning

Brenden M. Lake

Department of Psychology and Center for Data Science, New York University

Facebook AI Research

Abstract

People can learn a new concept and use it compositionally, understanding how to "blicket twice" after learning how to "blicket." In contrast, powerful sequence-to-sequence (seq2seq) neural networks fail such tests of compositionality, especially when composing new concepts together with existing concepts. In this paper, I show that neural networks can be trained to generalize compositionally through meta seq2seq learning. In this approach, models train on a series of seq2seq problems to acquire the compositional skills needed to solve new seq2seq problems. Meta se2seq learning solves several of the SCAN tests for compositional learning and can learn to apply rules to variables.

1 Introduction

People can learn new words and use them immediately in a rich variety of ways, thanks to their skills in compositional learning. Once a person learns the meaning of the verb "to Facebook", she or he can understand how to "Facebook slowly," "Facebook eagerly," or "Facebook while walking." These abilities are due to systematic compositionality, or the algebraic capacity to understand and produce novel utterances by combining familiar primitives [5, 26]. The "Facebook slowly" example depends on knowledge of English, but people generalize compositionally in other domains too, such as learning novel commands and meanings in artificial languages [17]. A key challenge for cognitive science and artificial intelligence is to understand the computational underpinnings of human compositional learning and to build machines with similar capabilities.

Neural networks have long been criticized for lacking compositionality, leading critics to argue they are inappropriate for modeling language and thought [8, 23, 24]. Nonetheless neural architectures have continued to advance and make important contributions in natural language processing (NLP) [19]. Recent work has revisited these classic critiques through studies of modern neural architectures [10, 15, 3, 20, 22, 2, 6], with a focus on the sequence-to-sequence (seq2seq) models used successfully in machine translation and other NLP tasks [32, 4, 36]. These studies show that powerful seq2seq approaches still have substantial difficulties with compositional generalization, especially when combining a new concept ("to Facebook") with previous concepts ("slowly" or "eagerly") [15, 3, 20].

New benchmarks have been proposed to encourage progress [10, 15, 2], including the SCAN dataset for compositional learning [15]. SCAN involves learning to follow instructions such as "walk twice and look right" by performing a sequence of appropriate output actions; in this case, the correct response is to "WALK WALK RTURN LOOK." A range of SCAN examples are shown in Table 1. Seq2seq models are trained on thousands of instructions built compositionally from primitives ("look", "walk", "run", "jump", etc.), modifiers ("twice", "around right," etc.) and conjunctions ("and" and "after"). After training, the aim is to execute, zero-shot, novel instructions such as "walk around right after look twice." Previous studies show that seq2seq recurrent neural networks (RNN) generalize well when the training and test sets are similar, but fail catastrophically when generalization requires systematic compositionality [15, 3, 20]. For instance, models often fail to understand how to "jump twice" after learning how to "run twice," "walk twice," and how to "jump." Developing neural architectures with these compositional abilities remains an open problem.

In this paper, I show that neural networks can be trained to generalize compositionally through "meta sequence-to-sequence learning" (meta seq2seq learning). As is standard with meta learning, training is distributed across a series of small datasets called "episodes" instead of a single static dataset [34, 30, 7], in a process called "meta-training." Specific to meta seq2seq learning, each episode is a novel seq2seq problem that provides "support" sequence pairs (input and output) and "query" sequences (input only), as shown in Figures 1 and 2. The network loads the support sequences into an external memory [31, 11, 29], which is used as context when responding to each query sequence by producing an output sequence. During each training episode, the network's output sequences are compared to the targets, showing the network how to generalize compositionally from the support items to the query items.

Meta seq2seq networks train on multiple seq2seq problems that require compositional generalization, with the aim of generalizing to new problems with similar characteristics. New seq2seq problems are solved entirely using the activation dynamics and external memory of the networks; no weight updates are used after meta-training stops. Through its reasoning capabilities, the network can learn implicit rules that operate on "variables," a long-standing challenge for neural network architectures [23, 24, 11, 22]. As demonstrated below, meta seq2seq learning can also solve several of the SCAN tasks for compositional learning, although generalizing to longer sequences remains a challenge.

2 Related work

Meta sequence-to-sequence learning builds on several areas of active research. Meta learning has been successfully applied to few-shot image classification [34, 30, 7, 18], including tasks that require storing information in an external memory [29]. Few-shot visual tasks are qualitatively different from the compositional, reasoning-based seq2seq tasks studied here, which demand different architectures and learning principles. Closer to the present work, meta learning has been recently applied to low resource machine translation [12], demonstrating one application of meta learning [7] to seq2seq translation problems. Crucially, these networks tackle a new task through weight updates rather than through memory and reasoning, and it is unclear whether this approach would work for compositional reasoning.

External memories have also expanded the capabilities of modern neural network architectures. Memory networks have been applied to reasoning and question answering tasks [31], in cases where only a single output is needed instead of a series of outputs. The Differentiable Neural Computer (DNC) [11] is also related to my proposal, in that a single architecture can reason through a wide range of scenarios, including seq2seq-like graph traversal tasks. The DNC addresses problems by using a complex architecture with multiple heads for reading and writing to memory, temporal links between memory cells, and trackers to monitor memory usage. In contrast, the meta seq2seq learner uses a simple memory mechanism akin to memory networks [31] and does not call the memory module with every new input symbol. Meta seq2seq uses higher-level abstractions to store and reason with entire sequences.

Recent work has demonstrated improvements on SCAN through other means, including clever forms of data augmentation [1] and syntax-based attention [28]. Although effective on some SCAN tasks, data augmentation does not aim to be a general approach to tackling compositional learning and reasoning tasks. In concurrent work, a syntactic attention model [28] has shown substantial improvements on some SCAN tasks. The approach can quickly learn new primitives, but it relies on SCAN-specific mechanisms that may not generalize well to other domains. Syntactic attention and meta seq2seq learning are compared in the SCAN experiments that follow.

3 Model

The meta sequence-to-sequence approach *learns how to learn* sequence-to-sequence (seq2seq) problems – it uses a series of training seq2seq problems to develop the needed compositional skills for solving new seq2seq problems. An overview of the meta seq2seq learner is illustrated in Figure 1. In this figure, the network is processing a query instruction "jump twice" in the context of a support set that shows how to "run twice," "walk twice", "look twice," and "jump." In broad strokes, the architecture is a standard seq2seq model [21] translating a query input into a query output (Figure 1). A recurrent neural network (RNN) encoder (f_{ie}; red RNN in bottom right of Figure 1) and a RNN decoder (f_{od}; green RNN in top right of Figure 1) work together to interpret the query sequence as an output sequence, with the encoder passing an embedding at each timestep (Q) to a Luong attention decoder [21]. The architecture differs
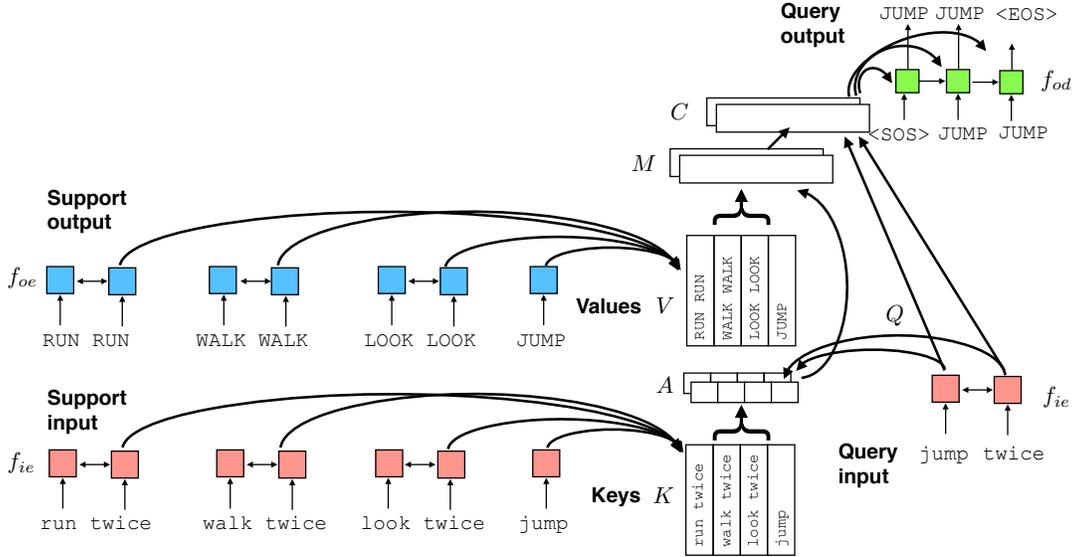
Figure 1: The meta sequence-to-sequence learner. The backbone is a sequence-to-sequence (seq2seq) network augmented with a context C produced by an external memory. The seq2seq model uses an RNN encoder (f_{ie}; bottom right) to read a query and then pass stepwise messages Q to an attention-based RNN decoder (f_{od}; top right). Distinctive to meta seq2seq learning, the messages Q are transformed into C based on context from the support set (left). The transformation operates through a key-value memory. Support item inputs are encoded and used a keys K while outputs are encoded and used as value V. The query is stepwise compared to the keys, retrieving weighted sums M of the most similar values. This is mapped to C which is decoded as the final output sequence. Color coding indicates shared RNN modules.

from standard seq2seq modeling through its use of the support set, external memory, and training procedure. As the messages pass from the query encoder to the query decoder, they are infused with stepwise context C provided by an external memory that stores the support items. The inner-working of the architecture are described in detail below.

Input encoder. The input encoder $f_{ie}(\cdot, \cdot)$ is shown as red in Figure 1 and is used for the query input instruction (e.g., "jump twice") and each of the of n_s support items and their input instructions ("run twice", "walk twice", "jump", etc.). The encoder first embeds the sequence of symbols (e.g., words) using an embedding layer to get a sequence of input embeddings $w_t \in \mathbb{R}^m$. The RNN processes each w_t to produce the RNN embedding $h_t \in \mathbb{R}^m$ such that

$$h_t = f_{ie}(h_{t-1}, w_t). \tag{1}$$

For the query sequence, the embedding h_t at each step $t = 1, \ldots, T$ passes through both the external memory and the output decoder. For each support sequence, only the last step embedding is needed, and thus each support instruction is expressed as a single vector $K_i \in \mathbb{R}^m$ for $i = 1, \ldots, n_s$. These RNN embeddings K_i become the keys in the external key-value memory (Figure 1). All of the experiments in this paper use bidirectional long short-term memory (biLSTM) encoders [13] although other choices are possible.

Output encoder. The output encoder $f_{oe}(\cdot, \cdot)$ is shown in blue in Figure 1 and used for each of the of n_s support items and their output sequences (e.g., "RUN RUN", "WALK WALK", "JUMP JUMP", etc.). First, the encoder embeds the sequence of output symbols (e.g., actions) using an embedding layer. Second, a single embedding for the entire sequence is computed using the same process as $f_{ie}(\cdot, \cdot)$ (Equation 1). Only the final RNN state is captured for each support item i and stored as the value vector $V_i \in \mathbb{R}^m$ for $i = 1, \ldots, n_s$ in the key-value memory. A biLSTM encoder is also used.

External memory. The architecture uses a soft key-value memory that operates similarly to memory networks [31]. The precise formulation used is described in [33]. The key-value memory uses the attention function

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{m}})V = AV, \tag{2}$$

3

with matrices Q, K, and V for the queries, keys, and values respectively, and the matrix A as the attention weights, $A = \text{softmax}(QK^T/\sqrt{m})$. Each query instruction spawns T embeddings from the RNN encoder, one for each query symbol, which populate the rows of the query matrix $Q \in \mathbb{R}^{T,m}$. The encoded support items form the rows of $K \in \mathbb{R}^{n_s,m}$ and the rows of $V \in \mathbb{R}^{n_s,m}$ for their input and output sequences, respectively. Attention weights $A \in \mathbb{R}^{T,n_s}$ indicates which memory cells are active for each query step. The output of the memory is a matrix $M = AV$ where each row is a weighted combination of the value vectors, indicating the memory output for each of the T query input steps, $M \in \mathbb{R}^{T,m}$. Finally, a stepwise context is computed by combining the query input embeddings h_t and the stepwise memory outputs $M_t \in \mathbb{R}^m$ with a concatenation layer $C_t = \tanh(W_{c_1}[h_t; M_t])$ producing a stepwise context matrix $C \in \mathbb{R}^{T,m}$.

For additional representational power, the key-value memory could replace the simple attention module with a multi-head attention module, or even a transformer-style multi-layer multi-head attention module [33]. This additional power was not needed for the tasks tackled in this paper, but it is compatible with the meta seq2seq approach.

Output decoder. The output decoder is shown in green in Figure 1 and translates the stepwise context C into a sequence of output symbols. The decoder embeds the previous output symbol to get a vector $o_{j-1} \in \mathbb{R}^m$ which is fed to the RNN (LSTM) along with the previous hidden state $g_{j-1} \in \mathbb{R}^m$ to get the next hidden state

$$g_j = f_{od}(g_{j-1}, o_{j-1}). \tag{3}$$

The initial hidden state g_0 is seeded with the context from the last step $C_T \in \mathbb{R}^m$. Luong-style attention [21] is used to compute a decoder context $u_j \in \mathbb{R}^m$ such that $u_j = \text{Attention}(g_j, C, C)$. This context is passed through another concatenation layer $\tilde{g}_j = \tanh(W_{c_2}[g_j; u_j])$ which is then mapped to a softmax output layer to produce an output symbol. This process repeats until all of the output symbols are produced and the RNN terminates the response by producing an end-of-sequence symbol.

Meta-training. Meta-training optimizes the network across a set of related training episodes. Each episode is a novel seq2seq problem that consists of a set of n_s support items pairs and a set of n_q query items (see Figure 2 for an example). Each seq2seq item has a sequence of input symbols and a sequence of output symbols. The support items are embedded and read into the key-value memory as described above. The vocabulary of the entire model is the union of the vocabulary from each of the training episodes. The loss function is the negative log-likelihood of the predicted output sequences for the queries.

If reasonable initial training progress can be made without the external memory, it is important to encourage the network to use its memory. One method passes the support items through the network as additional "query items" when computing the training loss, such that the overall loss is based on both the query and support items, i.e. using an auxiliary "support loss." The support output sequences have been observed, embedded, and stored in the key-value memory, thus it is not noteworthy that the network can learn to reconstruct these output sequences. Nevertheless, the support loss can help train the memory if the queries alone would lead the network to under-utilize the memory or get stuck in a local optimum. Other than the support loss, the network is never trained directly on the support items; it is only trained to make generalizations to query items.

My implementation processes each episode as a batch and takes just one gradient step per episode. For improved sample efficiency and training efficiency, the optimizer can take multiple gradient steps per episode or repeatedly cycle through the training episodes, although this was not explored here. The meta seq2seq learner is implemented in PyTorch.

4 Experiments

4.1 Architecture and training parameters

I use a common architecture and training procedure for all experiments in this paper. The meta seq2seq architecture builds upon the seq2seq architecture from [15] that performed best across a range of SCAN evaluations. The input and output sequence encoders are two-layer biLSTMs with $m = 200$ hidden units per layer and produce m dimensional embeddings. The output decoder is a two-layer LSTM also with $m = 200$ hidden units per layer. Dropout is applied with probability 0.5 to each symbol embedding and to each LSTM. A greedy decoder is used since it is effective on SCAN's deterministic outputs [15].

In each experiment, the network is meta-trained for 10,000 episodes with the ADAM optimizer [14]. Halfway through training, the learning rate is reduced from 0.001 to 0.0001.
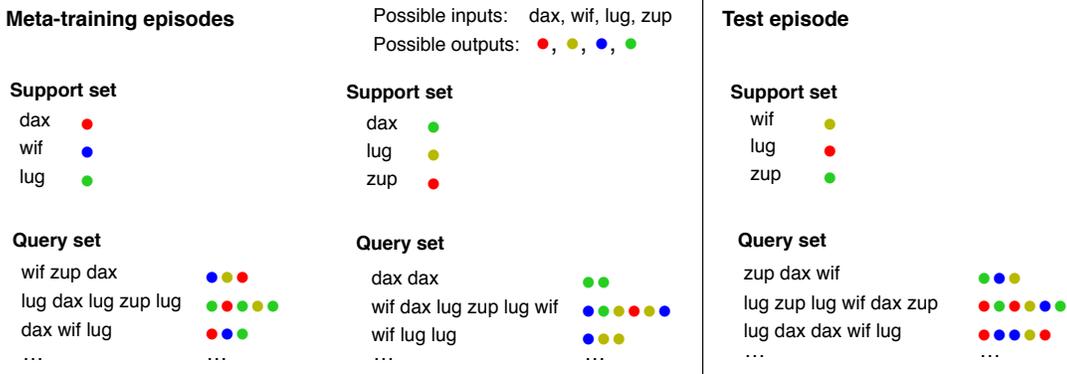
Figure 2: The mutual exclusivity task showing two meta-training episodes (left) and one test episode (right). Each episode requires executing instructions in a novel language of pseudowords (4 possible words include "dax", "wif", etc.) by producing sequences of colored circles (4 possible meanings including "red", "yellow", etc.). Each episode has a random mapping from pseudowords to meanings, with three isolated words and their outputs provided as the support set. Answering the queries requires concatenation as well as reasoning by mutual exclusivity to infer the mapping of the fourth word ("dax" means "blue" in the test episode). Best viewed in color.

Gradients with a l_2-norm larger than 50 are clipped. On the SCAN tasks, my meta seq2seq implementation takes less than an hour to train on a single NVIDIA Titan X GPU. For comparison, my seq2seq implementation takes less than 30 minutes. All models were trained five times with different random initializations and random meta-training episodes.

4.2 Experiment: Mutual exclusivity

This experiment examines the compositional skills of meta seq2seq learning through a synthetic task. As shown in Figure 2, each episode introduces a new mapping from non-sense words ("dax", "wif", etc.) to non-sense meanings ("red circle", "green circle", etc.), as demonstrated by the support set. To answer the queries, a model must demonstrate two abilities inspired by human generalization patterns [17]: 1) it must learn to use isolated symbol mappings to translate concatenated symbol sequences, and 2) it must learn to reason by mutual exclusivity (ME) to resolve unseen mappings. Children use ME to help learn the meaning of new words [25], making ME an important area of study in cognitive development. Using ME, children assume that if an object already has one label, then it does not need another. When provided with a familiar object (e.g., a cup) and an unfamiliar object (e.g., a cherry pitter) and asked to "Show me the dax," children tend to pick the unfamiliar object rather than the familiar one.

Adults also use ME to help resolve ambiguity. When presented with episodes like Figure 2 in a laboratory setting, participants use ME to resolve unseen mappings and translate queries of concatenated sequences in a symbol-by-symbol manner. Most people generalize in this way *spontaneously*, without any instructions or feedback about how to respond to compositional queries [17]. An untrained meta seq2seq learner would not be expected to generalize spontaneously – human participants come to the task with a starting point that is richer in every way – but computational models should nonetheless be capable of these inferences if trained to make them. This is a challenge for neural networks because the mappings change every episode, and standard architectures do not reason using ME – in fact, they tend to map novel inputs to the most familiar outputs [9], which is the opposite of reasoning by ME.

Experimental setup. The domain consists of four possible pseudowords (input symbols) and four possible meanings (output symbols). During meta-training, each episode is generated by sampling a random mapping from input symbols to output symbols (19 possibilities used for training). Three mappings are presented as support items and one is withheld. The queries consist of arbitrary concatenations of the pseudowords ranging in length from 2 to 6, which can be translated symbol-by-symbol to produce the proper output responses (20 queries per episode). The fourth input symbol – not shown in the support set – is also be queried. The model must learn how to use ME to map this unseen symbol to an unseen meaning rather than a seen meaning (Figure 2). During testing, the model is evaluated on five word-to-meaning mappings that were not seen during meta-training.

5

Table 1: SCAN task for compositional learning [15]. Examples show input instructions (left) and their corresponding output actions (right).

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

Results. Meta seq2seq successfully learns to concatenate and reason about novel mappings using ME, achieving 100% accuracy on the task (SD = 0%). Based on the isolated mappings stored in memory, the network learns to translate sequences of those items. Moreover, it can acquire and use new mappings at test time, utilizing only its external memory and the activation dynamics. By learning to use ME, the network shows it can reason about the *absence* of symbols in the memory rather than simply their presence. The attention weights and use of memory is visualized and presented in the appendix (Figure A.1).

4.3 Experiment: Adding a new primitive through permutation meta-training

This experiment applies meta seq2seq learning to the SCAN task of adding a new primitive [15]. Models are trained to generalize compositionally by decomposing the original SCAN seq2seq task into a series of related seq2seq sub-tasks. The goal is to learn a new primitive instruction and use it compositionally, operationalized in SCAN as the "add jump" split [15]. Models learn a new primitive "jump" and aim to use it compositionally in other instructions, resembling the "to Facebook" example introduced earlier in this paper. First, the original seq2seq problem from [15] is described. Second, the adapted problem for training meta seq2seq learners is described.

Seq2seq learning. Standard seq2seq models applied to SCAN have both a training and a test phase. During training, seq2seq models are exposed to the "jump" instruction in a single context demonstrating how to jump in isolation. Also during training, the models are exposed to all primitive and composed instructions for the other actions (e.g., "walk", "walk twice", "look around right and walk twice", etc.) along with the correct output sequences, which is about 13,000 unique instructions. Following [15], the critical "jump" demonstration is overrepresented in training to ensure it is learned.

During test, models are evaluated on all of the composed instructions that use the "jump" primitive, examining the ability to integrate new primitives and use them productively. For instance, models are evaluated on instructions such as "jump twice", "jump around right and walk twice", "walk left thrice and jump right thrice," along with about 7,000 other instructions using jump.

Meta seq2seq learning. Meta seq2seq models applied to SCAN have both a meta-training and a test phase. During meta-training, the models observe episodes that are variants of the original seq2seq problem, each of which requires rapid learning of new meanings for the primitives. Specifically, each meta-training episode provides a different random assignment of the primitive instructions ('jump','run', 'walk', 'look') to their meanings ('JUMP','RUN','WALK','LOOK'), with the restriction that the proper (original) permutation not be observed during meta-training. Withholding the original permutation, there are 23 possible permutations for meta-training. Each episode presents 20 support and 20 query instructions, with instructions sampled from the full SCAN set. The models predict the response to the query instructions, using the support instructions and their outputs as context. Through meta-training, the models are familiarized with all of the possible SCAN training and test instructions, but no episode maps all of its instructions to their original (target) outputs sequences. In fact, models have no signal to learn which primitives in general correspond to which actions, since the assignments are sampled anew for each episode.

During test, models are evaluated on rapid learning of new meanings. Just four support items are observed and loaded into memory, consisting of the isolated primitives ('jump','run', 'walk', 'look') paired with their original meanings ('JUMP','RUN','WALK','LOOK'). Notably, memory use at test time (with only four primitive items loaded in memory) diverges substantially from memory use during meta-training (with 20 complex instructions loaded in memory). To evaluate test accuracy, models make predictions on each of the original SCAN test instructions

Table 2: Test accuracy on the SCAN "add jump" task across different training paradigms.

Model	standard training	permutation meta-training	augmentation meta-training
meta seq2seq learning	—	**99.95%**	**98.71%**
-without support loss	—	5.43%	**99.48%**
-without decoder attention	—	10.32%	9.29%
standard seq2seq	0.03%	—	12.26%
syntactic attention [28]	78.4%	—	—

consisting of all composed instructions using "jump." An output sequence is considered correct only if it perfectly matches the whole target sequence.

Alternative models. The meta seq2seq learner is compared with an analogous "standard seq2seq" learner [21], which uses the same architecture with the external memory removed. The standard seq2seq learner is trained on the original SCAN problem with a fixed meaning for each primitive. Each meta seq2seq "episode" can be interpreted as a standard seq2seq "batch," and a batch size of 40 is chosen to equate the total number of presentations between approaches. All other architectural and training parameters are shared between meta seq2seq learning and seq2seq learning.

The meta seq2seq learner is also compared with two additional lesioned variants that examine the importance of different architectural components. First, the meta seq2seq learner is trained "without support loss" (Section 3 meta-training), which guides the architecture about how to best use its memory. Second, the meta seq2seq learner is trained "without decoder attention" (Section 3 output decoder). This leads to substantial differences in the architecture operation; rather than producing a sequence of context embeddings $C_1, \ldots, C_T$ for each step of the T steps of a query sequence, only the last step context C_T is computed and passed as a message to the decoder.

Results. The results are summarized in Table 2. On the "add jump" test set [15], standard seq2seq modeling completely fails to generalize compositionally, reaching an average performance of only 0.03% correct (SD = 0.02). It fails even while achieving near perfect performance on the training set (>99% on average). This replicates the results from [15] which trained many seq2seq models, finding the best network performed at only 1.2% accuracy. Again, standard seq2seq models do not show the necessary systematic compositionality.

The meta seq2seq learner succeeds at learning compositional skills, achieving an average performance of 99.95% correct (SD = 0.08). At test, the support set contains only the four primitives and their mappings, demonstrating that meta seq2seq learning can handle test episodes that are qualitatively different from those seen during training. Moreover, the network learns how to store and retrieve variables from memory with arbitrary assignments, as long as the network is familiarized with the possible input and output symbols during meta-training (but not necessarily how they correspond). A visualization of how meta seq2seq uses attention on SCAN is shown in the appendix (Figure A.2). The meta seq2seq learner also outperforms syntactic attention which achieves 78.4% and varies widely in performance across runs (SD = 27.4) [28].

The lesion analyses demonstrate the importance of various components. The meta seq2seq learner fails to solve the task without the guidance of the support loss, achieving only 5.43% correct (SD = 7.6). These runs typically learn the consistent, static meanings such as "twice", "thrice", "around right" and "after", but fail to use its memory properly to learn the dynamic primitives. The meta seq2seq learner also fails when the decoder attention is removed (10.32% correct; SD = 6.4), suggesting that a single m dimensional embedding is not sufficient to relate a query to the support items.

4.4 Experiment: Adding a new primitive through augmentation meta-training

Experiment 4.3 demonstrates that the meta seq2seq approach can learn how to learn the meaning of a primitive and use it compositionally. However, only a small set of four input primitives and four meanings was considered; it is unclear whether meta seq2seq learning works in more complex compositional domains. In this experiment, meta seq2seq is evaluated on a much larger domain produced by augmenting the meta-training with 20 additional input and action primitives. This more challenging task requires that the networks handle a much larger

set of possible meanings. The architecture and training procedures are identical to those used in Experiment 4.3 except where noted.

Seq2seq learning. To equate learning environment across approaches, standard seq2seq models use a training phase that is substantially expanded from that in Experiment 4.3. During training, the input primitives include the original four ('jump','run', 'walk', 'look') as well as 20 new symbols ('Primitive1,' ..., 'Primitive20'). The output meanings include the original four ('JUMP','RUN','WALK','LOOK') as well as 20 new actions ('Action1,' ..., 'Action20'). In the seq2seq training (but notably, not in meta seq2seq training), 'Primitive1' always corresponds to 'Action1,' 'Primitive2' corresponds to 'Action2,' and so on. A training batch uses the original SCAN templates with primitives sampled from the augmented set rather than the original set; for instance, a training instruction may be "look around right and Primitive20 twice." During training the "jump" primitive is only presented in isolation, and it is included in every batch to ensure the network learns it properly. Compared to Experiment 4.3, the augmented SCAN domain provides substantially more evidence for compositionality and productivity.

Meta seq2seq learning. Meta seq2seq models are trained similarly to Experiment 4.3 with an augmented primitive set. During meta-training, episodes are generated by randomly sampling a set of four primitive instructions (from the set of 24) and their corresponding meanings (from the set of 24). For instance, an example training episode could use the four instruction primitives 'Primitive16', 'run', 'Primitive2', and 'Primitive12' mapped respectively to actions 'Action3', 'Action20', 'JUMP', and 'Action11'. Although Experiment 4.3 has only 23 possible assignments, this experiment has orders-of-magnitude more possible assignments than training episodes, ensuring meta-training only provides a very small subset. Moreover, the models are evaluated using a stricter criterion for generalization: the primitive "jump" is never assigned to the proper action "JUMP" during meta-training.

The test phase is analogous to the previous experiment. Models are evaluated by loading all of the isolated primitives ('jump','run', 'walk', 'look') paired with their original meanings ('JUMP','RUN','WALK','LOOK') into memory as support items. No other items are included in memory. To evaluate test accuracy, models make predictions on the original SCAN test instructions consisting of all composed instructions using "jump."

Results. The results are summarized in Table 2. The meta seq2seq learner succeeds as picking up the meaning of "jump" and using it correctly, achieving 98.71% correct (SD = 1.49) on the test instructions. The slight decline in performance compared to Experiment 4.3 is not statistically significant with five runs. The standard seq2seq learner takes advantage of the augmented training to generalize better than in standard SCAN training (Experiment 4.3 and [15]), achieving 12.26% accuracy (SD = 8.33) on the test instructions (with >99% accuracy during training). The augmented task provides 23 fully compositional primitives during training, compared to the three in the original task. Despite this salient compositionality, the basic seq2seq model is still largely unable to make use of it.

The lesion analyses show that the support loss is not critical in this setting, and the meta seq2seq learner achieves 99.48% correct without it (SD = 0.37). In contrast to Experiment 4.3, using many primitives more strongly guides the network to use the memory, since the network cannot substantially reduce the training loss without it. The decoder attention remains critical in this setting, and the network attains merely 9.29% correct without it (SD = 13.07). These results demonstrate that only the full meta seq2seq learner is a satisfactory solution to both the learning problems in this experiment and the previous experiment (Table 2).

4.5 Experiment: Combining familiar concepts through meta-training

The previous experiments show that the meta seq2seq approach can learn how to learn a new primitive. The next experiment examines whether the approach can learn how to combine familiar concepts in new ways, based on the SCAN primitive "around right" split [20].

Seq2seq learning. Seq2seq training holds out all instances of "around right" while training on all of the other SCAN instructions. Using the symmetry between "left" and "right," the network must extrapolate to "jump around right" from training examples like "jump around left," "jump left," and "jump right." During test, the models are evaluated on all uses of "around right."

Meta seq2seq learning. Meta-training proceeds similarly to Experiment 4.4 with the goal of learning to infer the meaning of "around right" from "around" and "right" through augmentation. Instead of just "left" and "right", the possibilities also include "Direction1" and "Direction2" (or since the labels are arbitrary, "forward" and "backward"). Meta-training episodes are generated by randomly sampling two directions to be used in the instructions

(from "left", "right", "forward", "backward") and their meanings (from "LTURN," "RTURN," "FORWARD","BACKWARD"), permuted to have no systematic correspondence. The primitive "right" is never assigned to the proper meaning during meta-training, and thus "around right" is never mapped to its correct meaning either. As in the previous SCAN experiments, there are 20 support instructions and 20 query instructions. During test, models must infer the proper meaning of "around right" and use it compositionally to interpret all of it uses in the original SCAN instructions. The test support set is simply just "turn left" and "turn right" mapped to their proper meanings.

Results. Meta seq2seq learning is nearly perfect at inferring the meaning of "around right" from its components (99.96% correct; SD = 0.08; Table 3), while standard seq2seq fails catastrophically (0.0% correct). Syntactic attention also struggles (28.9%; SD = 34.8) [28].

In additional informal experiments, I experienced difficulty training the meta seq2seq learner with 20 additional directions instead of just two in the augmentation set. In this setting, it has trouble learning to use variables and achieves only 36.33% correct (SD = 7.30). Compared to the "add jump" split which was successful with 20 additional primitives, the "around right" split provides fewer meaning combinations per episode (two rather than four) and learning the directions is more nuanced than learning the actions.

4.6 Experiment: Generalizing to longer instructions through meta-training

The final experiment examines whether the meta seq2seq approach can learn to generalize to longer sequences, even when the test sequences are longer than any experienced during meta-training. This experiment uses the SCAN "length" split [15].

Seq2seq learning. The SCAN instructions are divided into training and test sets based on the number of required output actions. Standard seq2seq models are trained on all instructions that require 22 or fewer actions (about 17,000 instructions) and evaluated on all instructions that require longer action sequences (about 4,000 instructions ranging in length from 24-28). During test, the network must execute instructions such as "jump around right twice and look opposite right thrice" that require 25 actions. Both sub-instructions "jump around right twice" and "look opposite right thrice" are presented during training, but the model was never before asked to produce the conjunction or any output sequence of that length.

Meta seq2seq learning. Meta-training optimizes the network to extrapolate from shorter items in the support set to longer items in the query set. During test, the model is examined on *even longer* queries than seeing during meta-training (drawn from the SCAN "length" test set). To produce this training specification, the original "length" training set is sub-divided into the support pool (all instructions with less than 12 output actions) and a query pool (all instructions with 12 to 22 output actions). During meta-training, the network learns to respond to 20 longer instructions in the query pool given the shorter instructions in the support pool. To encourage the network to use its external memory (rather than learned weights) when answering queries, each episode applies primitive augmentation as in Experiment 4.4. To further amplify the memory, each episode also provides 100 support items. During test, the models load 100 support items from the original "length" split training set (lengths 1 to 22 output actions) and responds to queries from the original test set (lengths 24-28).

Results. None of the models perform well on longer sequences (Table 3). The meta seq2seq learner achieves 16.64% accuracy (SD = 2.10) while the baseline seq2seq learner achieves 7.71% (SD = 1.90). Syntactic attention [28] performs similarly to meta seq2seq at 15.2% (SD = 0.7). Although the meta seq2seq learner has compositional capabilities, it lacks the truly systematic compositionality needed to properly produce longer output sequences.

Table 3: Test accuracy on the SCAN "around right" and "length" tasks.

Model	around right	length
meta seq2seq learning	**99.96%**	16.64%
standard seq2seq	0.0%	7.71%
syntactic attention [28]	28.9%	15.2%

5 Discussion

People are skilled compositional learners while standard neural networks are not. After learning how to "dax," people understand how to "dax twice," "dax slowly," or even "dax like there is no tomorrow." These abilities are central to language and thought yet they are conspicuously lacking in modern neural networks [15, 3, 20, 22, 2].

In this paper, I introduced a meta sequence-to-sequence (meta seq2seq) approach for learning to generalize compositionally, exploiting the algebraic structure of a domain to help

understand novel utterances. In contrast to standard seq2seq, meta seq2seq learners can abstract away the surface patterns and operate closer to rule space. Rather than attempting to solve "jump around right twice and walk thrice" by comparing surface level patterns with training items, meta seq2seq learns to treat the instruction as a template "x around right twice and y thrice", where x and y are variables that can be filled arbitrarily. This approach is able to solve SCAN tasks for compositional learning that have eluded standard NLP approaches, with the exception of generalizing to longer sequences [15]. In this way, meta seq2seq learners are several steps closer to capturing the compositional abilities studied in synthetic learning tasks [17] and motivated in the "to dax" or "to Facebook" thought experiments.

Meta seq2seq learning has implications for understanding how people generalize compositionally. Similarly to meta-training, people learn in dynamic rather than static environments, tackling a series of changing learning problems rather than iterating repeatedly through a static dataset. There is natural pressure to generalize systematically after a single experience with a new verb like "to Facebook," and thus people are incentivized to generalize compositionally in ways that may resemble the meta-training loss introduced here. Meta learning is a powerful new toolbox for studying learning-to-learn and other elusive cognitive abilities [16, 35], although more work is needed to understand its implications for cognitive science.

The models studied here can learn variables that assign novel meanings to words at test time, using only the network dynamics and the external memory. Although powerful, this is a limited concept of "variable" since it requires familiarity with all of the possible input and output assignments during meta-training. This limitation is shared by nearly all existing neural architectures [31, 11, 29] and shows that the meta seq2seq framework falls short of addressing Marcus's challenge of extrapolating outside the training space [23, 24, 22]. In future work, I intend to explore adding more symbolic machinery to the architecture [27] with the goal of handling genuinely new symbols. Hybrid models could also address the challenge of generalizing to longer output sequences, a problem that continues to vex neural networks [15, 3, 28] including meta seq2seq learning.

The meta seq2seq approach could be applied to a wide range of tasks including low resource machine translation [12] or to graph traversal problems [11]. For traditional seq2seq tasks like machine translation, standard seq2seq training could be augmented with hybrid training that alternates between standard training and meta-training to encourage compositional generalization. I am excited about the potential of the meta seq2seq approach both for solving practical problems and for illuminating the foundations of human compositional learning.

Acknowledgments

References

[1] Jacob Andreas. Good-Enough Compositional Data Augmentation. *arXiv preprint*, 2019.

[2] Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. Systematic generalization: What is required and can it be learned? pages 1–16, 2018.

[3] Joost Bastings, Marco Baroni, Jason Weston, Kyunghyun Cho, and Douwe Kiela. Jump to better conclusions: SCAN both left and right. In *Proceedings of the EMNLP BlackboxNLP Workshop*, pages 47–55, Brussels, Belgium, 2018.

[4] Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. Findings of the 2016 Conference on Machine Translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, 2016.

[5] Noam Chomsky. *Syntactic Structures*. Mouton, Berlin, Germany, 1957.

[6] Ishita Dasgupta, Demi Guo, Andreas Stuhlmuller, Samuel J Gershman, and Noah D Goodman. Evaluating Compositionality in Sentence Embeddings. *arXiv preprint*, 2018.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *International Conference on Machine Learning (ICML)*, 2017.

[8] Jerry Fodor and Zenon Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28:3–71, 1988.

[9] Kanishk Gandhi and Brenden M Lake. Mutual exclusivity as a challenge for neural networks. *arXiv preprint*, 2019.

[10] Samuel J Gershman and Joshua B Tenenbaum. Phrase similarity in humans and machines. In *Proceedings of the 37th Annual Conference of the Cognitive Science Society*, 2015.

[11] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 2016.

[12] Jiatao Gu, Yong Wang, Yun Chen, Kyunghyun Cho, and Victor OK Li. Meta-Learning for Low-Resource Neural Machine Translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[13] S Hochreiter and J Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–1780, 1997.

[14] Diederik P Kingma and Max Welling. Efficient Gradient-Based Inference through Transformations between Bayes Nets and Neural Nets. In *International Conference on Machine Learning (ICML 2014)*, 2014.

[15] Brenden M Lake and Marco Baroni. Generalization without Systematicity: On the Compositional Skills of Sequence-to-Sequence Recurrent Networks. In *International Conference on Machine Learning (ICML)*, 2018.

[16] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40:E253, 2017.

[17] Brenden M Lake, Tal Linzen, and Marco Baroni. Human few-shot learning of compositional instructions. In *Proceedings of the 41st Annual Conference of the Cognitive Science Society*, 2019.

[18] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. The Omniglot Challenge: A 3-Year Progress Report. *Current Opinion in Behavioral Sciences*, 29:97–104, 2019.

[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[20] João Loula, Marco Baroni, and Brenden M Lake. Rearranging the Familiar: Testing Compositional Generalization in Recurrent Networks. *arXiv preprint*, 2018. URL `http://arxiv.org/abs/1807.07545`.

[21] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

[22] Gary Marcus. Deep Learning: A Critical Appraisal. *arXiv preprint*, 2018.

[23] Gary F Marcus. Rethinking Eliminative Connectionism. *Cognitive Psychology*, 282(37):243–282, 1998.

[24] Gary F Marcus. *The Algebraic Mind: Integrating Connectionism and Cognitive Science*. MIT Press, Cambridge, MA, 2003.

[25] Ellen M Markman and Gwyn F Wachtel. Children's Use of Mutual Exclusivity to Constrain the Meanings of Words. *Cognitive Psychology*, 20:121–157, 1988.

[26] Richard Montague. Universal Grammar. *Theoria*, 36:373–398, 1970.

[27] Scott Reed and Nando de Freitas. Neural Programmer-Interpreters. In *International Conference on Learning Representations (ICLR)*, 2016.

[28] Jake Russin, Jason Jo, Randall C. O'Reilly, and Yoshua Bengio. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *arXiv preprint*, 2019. URL `http://arxiv.org/abs/1904.09708`.

[29] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-Learning with Memory-Augmented Neural Networks. In *International Conference on Machine Learning (ICML)*, 2016.

[30] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[31] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-To-End Memory Networks. In *Advances in Neural Information Processing Systems 29*, 2015.

[32] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Advances in Neural Information Processing Systems.*, 2017.

[34] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching Networks for One Shot Learning. In *Advances in Neural Information Processing Systems 29 (NIPS)*, 2016.

[35] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv*, 2016. URL `http://arxiv.org/abs/1611.05763`.

[36] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. `http://arxiv.org/abs/1609.08144`, 2016.

A   Appendix

Attention maps for mutual exclusivity (ME) task. Attention is visualized in Figure A.1 for a test episode in the ME task (Experiment 4.2) with two queries "lug zup lug wif dax zup" and "lug dax dax wif lug." When passing a query symbol-by-symbol through the key-value memory (Figure A.1 left), the network allocates attention to all of the cells that *do not* contain the current query symbol, a counterintuitive but valid encoding strategy. This pattern is reversed in the last step before the end-of-sequence symbol (<EOS>), where more intuitively the input symbol activates the memory cell that contains its corresponding support item. The withheld ME symbol "dax" leads to a broad, uniform pattern of attention spread across the support items, indicating its novelty.

The RNN decoder attention is more straightforward. The diagonal pattern indicates strong alignment between each output symbol from the decoder (color; row) and its corresponding input symbol in the encoder (pseudoword; column). The first decoder step is an exception because the decoder hidden state is initialized with the last context step C_T (Section 3). The attention vectors do not sum to 1 because of padded elements from the batched decoder.
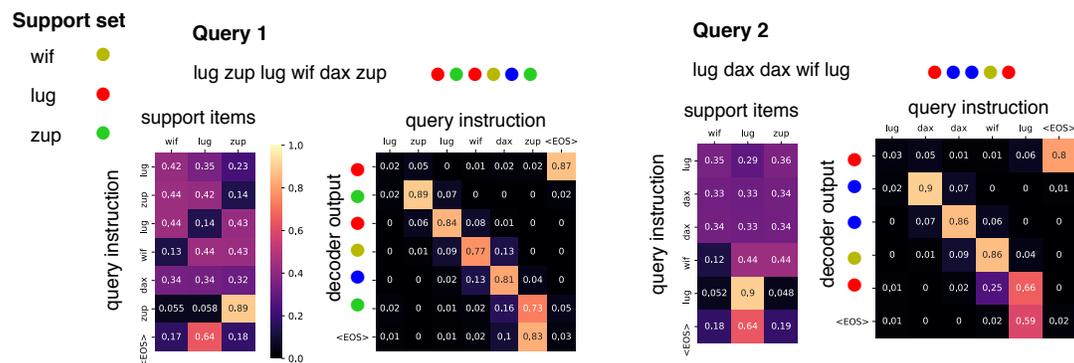

Figure A.1: During a test episode of the ME task, the support set (top left) and two queries are shown. The ME inference is that "dax" maps to "blue." The key-value memory attention A for each query is shown in the left matrix, with rows as encoder steps and columns as support items. The decoder attention for each query is shown in the right matrix, with rows as the decoder steps and columns as encoder steps. The end-of-sequence symbol is <EOS>.

Attention maps for SCAN. Attention is visualized in Figure A.2 for a test episode in the "add jump" task (Experiment 4.4). The key-value memory attention provides a lookup mechanism for retrieving the response for each input primitive, including "walk" and "run" (Figure A.2 left). The decoder attention also provides an intuitive alignment, attending to "run", "right," and "thrice" in alternation while executing "run right thrice" (Figure A.2 right).
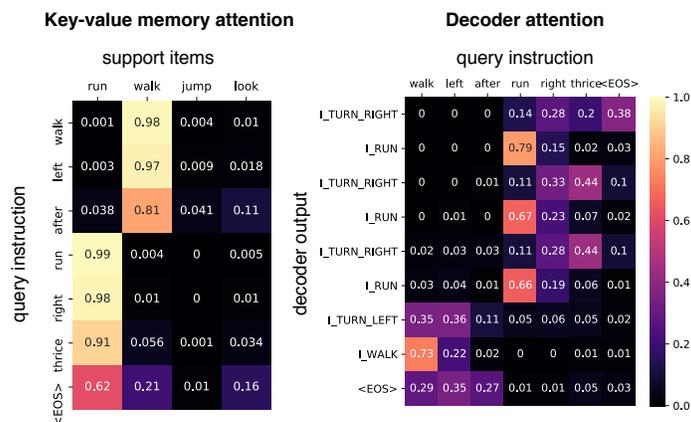

Figure A.2: Attention in meta seq2seq learning on the SCAN task. During test, the network is evaluated on the query "walk left after run right thrice." The end-of-sequence symbol is <EOS>.