Generalization without Systematicity: Supplementary materials

SCAN grammar and interpretation function

The phrase-structure grammar generating all SCAN commands is presented in Figure 1. The corresponding interpretation functions is in Figure 2.

Standard Encoder-Decoder RNN

We describe the encoder-decoder framework, borrowing from the description in Bahdanau et al. (2015). The encoder receives a natural language command as a sequence of T words. The words are transformed into a sequence of vectors, $\{w_1, \ldots, w_T\}$, which are learned embeddings with the same number of dimensions as the hidden layer. A recurrent neural network (RNN) processes each word

$$h_t = f_E(h_{t-1}, w_t),$$

where h_t is the encoder hidden state. The final hidden state h_T (which may include multiple layers for multi-layer RNNs) is passed to the RNN decoder as hidden state g_0 (see seq2seq diagram in the main article). Then, the RNN decoder must generate a sequence of output actions $a_1, \ldots, a_R$. To do so, it computes

$$g_t = f_D(g_{t-1}, a_{t-1}),$$

where g_t is the decoder hidden state and a_{t-1} is the (embedded) output action from the previous time step. Last, the hidden state g_t is mapped to a softmax to select the next action a_t from all possible actions.

Attention Encoder-Decoder RNN

For the encoder-decoder with attention, the encoder is identical to the one described above. Unlike the standard decoder that can only see h_T, the attention decoder can access all of the encoder hidden states, $h_1, \ldots, h_T$ (in this case, only the last layer if multi-layer). At each step i, a context vector c_i is computed as a weighted sum of the encoder hidden states

$$c_i = \sum_{t=1}^{T} \alpha_{it} h_t.$$

The weights α_{it} are computed using a softmax function

$$\alpha_{it} = \exp(e_{it}) / \sum_{j=1}^{T} \exp(e_{ij}),$$

where $e_{it} = v_a^\top \tanh(W_a g_{i-1} + U_a h_t)$ is an alignment model that computes the similarity between the previous decoder hidden state g_{i-1} and an encoder hidden state h_t (for the other variables, v_a, W_a, and U_a are learnable parameters) (Bahdanau et al., 2015). This context vector c_i is then passed as input to the decoder RNN at each step with the function

$$g_i = f_D(g_{i-1}, a_{i-1}, c_i),$$

which also starts with hidden state $g_0 = h_T$, as in the standard decoder. Last, the hidden state g_i is concatenated with c_i and mapped to a softmax to select new action a_i.

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

Figure 1: Phrase-structure grammar generating SCAN commands. We use indexing notation to allow infixing: D[i] is to be read as the i-th element directly dominated by category D.

$[\![\text{walk}]\!] = \text{WALK}$
$[\![\text{look}]\!] = \text{LOOK}$
$[\![\text{run}]\!] = \text{RUN}$
$[\![\text{jump}]\!] = \text{JUMP}$
$[\![\text{turn left}]\!] = \text{LTURN}$
$[\![\text{turn right}]\!] = \text{RTURN}$
$[\![u \text{ left}]\!] = \text{LTURN } [\![u]\!]$
$[\![u \text{ right}]\!] = \text{RTURN } [\![u]\!]$
$[\![\text{turn opposite left}]\!] = \text{LTURN LTURN}$
$[\![\text{turn opposite right}]\!] = \text{RTURN RTURN}$
$[\![u \text{ opposite left}]\!] = [\![\text{turn opposite left}]\!]\ [\![u]\!]$
$[\![u \text{ opposite right}]\!] = [\![\text{turn opposite right}]\!]\ [\![u]\!]$
$[\![\text{turn around left}]\!] = \text{LTURN LTURN LTURN LTURN}$
$[\![\text{turn around right}]\!] = \text{RTURN RTURN RTURN RTURN}$
$[\![u \text{ around left}]\!] = \text{LTURN } [\![u]\!] \text{ LTURN } [\![u]\!] \text{ LTURN } [\![u]\!] \text{ LTURN } [\![u]\!]$
$[\![u \text{ around right}]\!] = \text{RTURN } [\![u]\!] \text{ RTURN } [\![u]\!] \text{ RTURN } [\![u]\!] \text{ RTURN } [\![u]\!]$
$[\![x \text{ twice}]\!] = [\![x]\!]\ [\![x]\!]$
$[\![x \text{ thrice}]\!] = [\![x]\!]\ [\![x]\!]\ [\![x]\!]$
$[\![x_1 \text{ and } x_2]\!] = [\![x_1]\!]\ [\![x_2]\!]$
$[\![x_1 \text{ after } x_2]\!] = [\![x_2]\!]\ [\![x_1]\!]$

Figure 2: Double brackets ($[\![\,]\!]$) denote the interpretation function translating SCAN's linguistic commands into sequences of actions (denoted by uppercase strings). Symbols x and u denote variables, the latter limited to words in the set $\{$walk, look, run, jump$\}$. The linear order of actions denotes their temporal sequence.

References

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR Conference Track*, San Diego, CA. Published online: `http://www.iclr.cc/doku.php?id=iclr2015:main`.