Jump to better conclusions: SCAN both left and right

Jasmijn Bastings¹ Marco Baroni² Jason Weston² Kyunghyun Cho²,³,⁴ Douwe Kiela²
¹ILLC, University of Amsterdam
²Facebook AI Research
³New York University
⁴CIFAR Global Scholar
bastings@uva.nl {mbaroni,jase,kyunghyuncho,dkiela}@fb.com

Abstract

Lake and Baroni (2018) recently introduced the SCAN data set, which consists of simple commands paired with action sequences and is intended to test the strong generalization abilities of recurrent sequence-to-sequence models. Their initial experiments suggested that such models may fail because they lack the ability to extract systematic rules. Here, we take a closer look at SCAN and show that it does not always capture the kind of generalization that it was designed for. To mitigate this we propose a complementary dataset, called NACS. We show that models that do well on SCAN do not necessarily do well on NACS, and that NACS exhibits properties more closely aligned with realistic use-cases for sequence-to-sequence models.

1 Introduction

In a recent paper, Lake and Baroni (2018) (L&B) investigate if recurrent sequence-to-sequence models can exhibit the same strong generalization that humans are capable of, by virtue of our capacity to infer the meaning of a phrase from its constituent parts (i.e., compositionality), providing empirical tests for this long-standing goal (Fodor and Pylyshyn, 1988). Compositional generalization might be a fundamental component in making models drastically less sample-thirsty than they currently are. L&B introduce the SCAN data set (§2), meant to study such generalization to novel examples. It consists of simple command-action pairs, in which more complex commands are composed of simpler ones (see Figure 1 for examples).

SCAN comprises several tests of generalization, namely with respect to (1) a random subset of the data (‘simple’), (2) commands with action sequences longer than those seen during training (‘length’), and (3) commands that compose a primitive in novel ways that was only seen in isolation during training (‘primitive’). In the latter case, the training set would for example only include the command ‘jump’, after which the test set includes all other commands containing ‘jump’, e.g. ‘jump opposite left after walk twice’.

In this paper we take a closer look at SCAN. We start with the observation (§3) that there are few target-side dependencies in the data, a consequence of SCAN being generated from a phrase-structure grammar. We show (§6) that this allows simple sequence-to-sequence models (§5) to obtain good accuracies e.g. on tasks involving a new primitive, even without access to previous outputs. However, these simple models do not use composition in any interesting way, and their performance is therefore not a realistic indicator of their generalization capability. We hence propose NACS (§4) as a more realistic alternative: SCAN with commands and actions flipped, i.e., mapping actions back to their original commands. This is harder, because different commands may map to the same action sequence, and it introduces target-side dependencies, so that previous outputs need to be remembered.

We show in particular that well-tuned attention-based models do achieve a certain degree of gen-

jump
JUMP

turn around left
LTURN LTURN LTURN LTURN

jump thrice and turn left twice
JUMP JUMP JUMP LTURN LTURN

jump opposite left after walk twice
WALK WALK LTURN LTURN JUMP

Figure 1: SCAN maps commands to actions
eralization on SCAN, and, as predicted, simpler models do better there. However, the models still struggle in the more demanding NACS setup, which we offer as a challenge for future work.

Our contributions can be summarized as follows:

1. we provide an analysis of SCAN and make the important observation that it does not test for target-side dependencies, allowing too simple models to do well;
2. we propose NACS to introduce target-side dependencies and remedy the problem;
3. we repeat all experiments in Lake and Baroni (2018) using early-stopping on validation sets created from the training data.

2 SCAN

SCAN stands for Simplified version of the Comma AI Navigation tasks (Mikolov et al., 2016). Each example in SCAN is constructed by first sampling a command \( X = (x_1, \ldots, x_T) \) from a finite phrase-structure grammar with start symbol \( C \):

\[
C \to S \text{ and } S \mid S \text{ after } S \mid S \\
S \to V \text{ twice } \mid V \text{ thrice } \mid V \\
V \to D_{[1]} \text{ opposite } D_{[2]} \mid D_{[1]} \text{ around } D_{[2]} \mid D \mid U \\
D \to U \text{ left } \mid U \text{ right } \mid \text{ turn left } \mid \text{ turn right } \\
U \to \text{ walk } \mid \text{ look } \mid \text{ run } \mid \text{ jump }
\]

For each command, the corresponding target action sequence \( Y = (y_1, \ldots, y_T') \) then follows by applying a set of interpretation functions, such as:

\[
\begin{align*}
\text{jump} & \mapsto \text{JUMP} \\
[u \text{ around left}] & \mapsto \text{LTURN} [u] \text{ LTURN} [u] \\
\text{LTURN} [u] \text{ LTURN} [u] \\
[x_1 \text{ after } x_2] & \mapsto [x_2] [x_1]
\end{align*}
\]

of which only the last function requires global reordering, which occurs at most once per command. See the supplementary materials for the full set. Figure 1 shows examples of commands and their action sequences as obtained by the interpretation functions. The commands can be decoded compositionally by a learner by discovering the interpretation functions, enabling generalization to unseen commands. The total data set is finite but large (20910 unambiguous commands).

3 SCAN prefers simple models

We observe an important property of the data set generation process for SCAN: temporal dependencies of the action sequence are limited to the phrasal boundaries of each sub-phrase, which span at most 24 actions (e.g. jump around left thrice). Crucially, even rules that require repetition (such as ‘thrice’) as well as global reordering, can be resolved by simple counting and without remembering previously generated outputs, due to the limited depth of the phrase-structure grammar (see e.g. Rodriguez and Wiles (1998)).

This observation has two important implications. First, because SCAN is largely a phrase-to-phrase conversion problem, any machine learning method that aims at solving SCAN needs to have an alignment mechanism between the source and target sequences. Such an alignment mechanism could work fairly accurately by simply advancing a pointer. Somewhat contrary to the observation by Lake and Baroni (L&B), we therefore hypothesize that an attention mechanism (Bahdanau et al., 2015) always helps when a neural conditional sequence model (Sutskever et al., 2014; Cho et al., 2014) is used to tackle any variant of SCAN. Second, we speculate that any algorithm with strong long-term dependency modeling capabilities can be detrimental in terms of generalization, because such an approach might inappropriately capture spurious target-side regularities in the training data. We thus hypothesize that less powerful decoders generalize better on to unseen action combinations on SCAN when equipped with an attention mechanism.

To summarize: good performance on SCAN does not necessarily indicate the capability of a model to strongly generalize.

SCAN favors simpler models that need not capture target-side dependencies, which might not work well on more realistic sequence-to-sequence problems, such as machine translation, where strong auto-regressive models are needed for good results (Bahdanau et al., 2015; Kaiser and Bengio, 2016).

4 NACS: actions to commands

By simply flipping the source \( X \) and target \( Y \) of each example, we obtain a data-set that suddenly features strong target-side dependencies. Even when the mapping \( p(Y|X) \) from the source to target is simple, the opposite \( p(X|Y) \propto
get sequence given a source sequence \( p(Y|X) \). Our encoder-decoder is modeled after Cho et al. (2014) and our attention-based encoder-decoder after Bahdanau et al. (2015). The attention-based decoder is a function that takes as input the previous target word embedding \( e_{y_{i-1}} \), the context vector \( c_{t_i} \), and the previous hidden state \( s_{i-1} \) (see also Figure 2): \( s_i = f(e_{y_{i-1}}, c_{t_i}, s_{i-1}) \).

The prediction for the current time step is then made from a pre-output layer \( t_i \): \( t_i = W_e e_{y_{i-1}} + W_o c_{t_i} + W_s s_i \). We do not apply a max-out layer and directly obtain the output by \( o_i = W_t t_i \). For the encoder-decoder without attention, the prediction is made directly from decoder state \( s_i \). We vary the recurrent cell, experimenting with simple RNN (Elman, 1990), GRU (Cho et al., 2014), and LSTM cells (Hochreiter and Schmidhuber, 1997). For conciseness we only report results with RNN and GRU cells in the main text, and LSTM results in the appendix.

In this paper, we investigate the properties of both SCAN and NACS using RNN-based sequence-to-sequence models for evaluation. We leave further investigation of alternative architectures (see, e.g., Vaswani et al., 2017; Gehring et al., 2017; Chen et al., 2018) for the future.

6 Experiments

6.1 Settings

Our models are implemented in PyTorch and trained using mini-batch SGD with an initial learning rate of 0.2, decayed by 0.96 each epoch. We use a batch size of 32, 256 hidden units (64 for embeddings), and a dropout rate of 0.2. We test on all SCAN/NACS tasks\(^1\), as well as on the Fr-En Machine Translation (MT) task that L&B used. The reported results are averaged over three runs for each experiment. Models with attention are marked as such with +Attn, e.g., ‘GRU +Attn’.

Validation Set. L&B split each SCAN subtask into a training set (80%) and a test set (20%). They train for a fixed number of updates (100k) and evaluate on the test set. Because any training run without early stopping may have missed the optimal solution (Caruana et al., 2001), we believe their results may not reflect the reality as closely as they could. We thus augment each of the SCAN variants with a validation set that follows the training distribution but contains examples that are not

---

\(^1\)github.com/facebookresearch/NACS
have a perfect accuracy on SCAN, while at the simple task for NACS show that it is possible to SCAN simple task. While this is impressive, re-
model variants perform (near) perfectly on the
tasks. If our analysis in § 
put in the supplementary materials. We will
evaluate future progress. For each experiment we remove 10% of the training examples to be
experiments so that they are better benchmarks for evaluating future progress. For each experiment
preferential (composed) ‘jump’ commands were added
Since there is no evidence in the training data that
occurring on the target-side in other contexts dur-
while this is impressive, re-
sults for the severed models (+Attn -Dep) obtain the high-
est scores on SCAN, but are the worst on NACS.

Ablations. To validate our analysis, we remove
the connections from the previous target word embedding \( e_{y_{t-1}} \) to the decoder state and the pre-
output layer (\( es \) and \( et \) in Figure 2), so that the current prediction is not informed by previous out-
puts. If our analysis in §3 is correct, then these simpler models should still be able to make the correct predictions on SCAN, but not on NACS.

6.2 Results and Analysis

Results on the three SCAN and NACS tasks are listed in Table 1. The full results including mod-
els with LSTM cells and MT experiments may be found in the supplementary materials. We will
now discuss our observations.

SCAN is not enough. Table 1 shows that all model variants perform (near) perfectly on the
SCAN simple task. While this is impressive, re-
sults for the severed models (+Attn -Dep) on the
simple task for NACS show that it is possible to have a perfect accuracy on SCAN, while at the
same time failing to do well on NACS.\(^2\) Crucially,
a (near) perfect score on SCAN does not imply
strong generalization. A model can exploit the deter-
minism and lack of target-side dependencies of
SCAN by developing a simple translation strategy
such as advancing a pointer and translating word
by word, and the use of such a simple strategy is not revealed by SCAN.

NACS is harder. NACS is a harder problem to solve compared to SCAN, as evidenced by con-
sistently lower accuracies in Table 1 for all tasks. The discrepancy between SCAN and NACS per-
ance is the most extreme when we look at the primitive tasks (turn left and jump). For turn left, the severed models (+Attn -Dep) obtain the highest
scores on SCAN, but are the worst on NACS.

The ‘turn left’ task benefits from TURNL occurring on the target-side in other contexts dur-
ing training, which is not the case for ‘jump’.\(^3\) Since there is no evidence in the training data that ‘jump’ is a verb, Table 2 shows results where addi-
tional (composed) ‘jump’ commands were added
for training. We see that performance quickly goes up when adding more commands.\(^4\) Again here the simpler models (+Attn -Dep) perform better.

Machine Translation. We repeat L&Bs
English-French MT experiment for both direc-
tions. Table 3 shows that models that perform
well on NACS also perform well here, with the
GRU outperforming the other cells (see appendix

\(^2\)We made similar observations using LSTM cells, as we show in the appendix.
\(^3\)See Lake and Baroni (2018) for a discussion.
\(^4\)L&B performed this experiment without attention, which we show has a large positive impact.
and Wiles (1998) found that simple RNNs can account for trivial solutions, which do not work on target side. We argue that a good benchmark needs at least those properties, in order not to fall prey to trivial solutions, which do not work on novel 2-tuples on an auto-association task, while Wong and Wang (2007) and Brakel and Frank (2009) found that RNNs can show systematic behavior in a language modeling task.

Frank (2009) found that RNNs can show systematic behavior in a language modeling task. While Phillips (1998) found that neural networks are unable to show strong generalization, many attempts were made to show that the opposite is true, leading to inconclusive evidence. For example, Weiss et al. (2018) show that LSTMs and simple RNNs with ReLU-activation can learn to count unboundedly, in contrast to GRUs.

Linzen et al. (2016) probed the sensitivity of LSTMs to hierarchical structure (not necessarily in novel constructions). Instead of a binary choice, with SCAN a sequence-to-sequence model productively generates an output string.

Liska et al. (2018) found that a small number of identical RNNs trained with different initializations show compositional behavior on a function composition task, suggesting that more specific architectures may not be necessary.

Finally, Lake and Baroni (2018) introduced the SCAN data set to study systematic compositionality in recurrent sequence-to-sequence models, including gating mechanisms and attention. This work is a direct response to that and aims to facilitate future progress by showing that SCAN does not necessarily test for strong generalization.

### 7 Related Work

Ever since Fodor and Pylyshyn (1988) conjectured that neural networks are unable to show strong generalization, many attempts were made to show that the opposite is true, leading to inconclusive evidence. For example, Phillips (1998) found that feed-forward nets and RNNs do not always generalize to novel 2-tuples on an auto-association task, while Wong and Wang (2007) and Brakel and Frank (2009) found that RNNs can show systematic behavior in a language modeling task.

In the context of analyzing RNNs, Rodriguez and Wiles (1998) found that simple RNNs can develop a symbol-sensitive counting strategy for accepting a simple (palindrome) context-free language. Weiss et al. (2018) show that LSTMs and simple RNNs with ReLU-activation can learn to count unboundedly, in contrast to GRUs.

Table 2: Test scores on the ‘jump’ task with additional commands. +Attn marks attention, -Dep has the es and et connections removed (Figure 2). The test set contains all jump commands except the 32 used for training. Columns indicate how many commands with ‘jump’ were added to the training set, such as ‘jump around left thrice.’

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RNN +Attn</strong></td>
<td>35.0 ± 2.8</td>
<td>48.6 ± 8.1</td>
<td>77.6 ± 2.6</td>
<td>89.2 ± 3.8</td>
<td>98.7 ± 1.3</td>
<td>99.8 ± 0.1</td>
</tr>
<tr>
<td><strong>RNN -Attn</strong></td>
<td>29.5 ± 10.5</td>
<td>53.3 ± 10.2</td>
<td>82.4 ± 4.7</td>
<td>98.8 ± 0.8</td>
<td>99.8 ± 0.1</td>
<td>100.0 ± 0.0</td>
</tr>
<tr>
<td><strong>GRU +Attn</strong></td>
<td>58.2 ± 12.0</td>
<td>67.8 ± 3.4</td>
<td>80.3 ± 7.0</td>
<td>88.0 ± 6.0</td>
<td>98.3 ± 1.8</td>
<td>99.6 ± 0.2</td>
</tr>
<tr>
<td><strong>GRU -Attn</strong></td>
<td>70.9 ± 11.5</td>
<td>61.3 ± 13.5</td>
<td>83.5 ± 6.1</td>
<td>99.0 ± 0.4</td>
<td>99.7 ± 0.2</td>
<td>100.0 ± 0.0</td>
</tr>
<tr>
<td><strong>NACS</strong></td>
<td>2.8 ± 0.8</td>
<td>9.3 ± 7.3</td>
<td>24.7 ± 4.2</td>
<td>43.7 ± 4.4</td>
<td>57.1 ± 5.2</td>
<td>69.1 ± 2.1</td>
</tr>
<tr>
<td><strong>NACS +Attn</strong></td>
<td>0.4 ± 0.1</td>
<td>0.9 ± 0.2</td>
<td>2.4 ± 0.3</td>
<td>3.9 ± 0.3</td>
<td>9.3 ± 0.3</td>
<td>15.9 ± 1.4</td>
</tr>
<tr>
<td><strong>NACS -Attn</strong></td>
<td>5.5 ± 1.8</td>
<td>9.2 ± 2.8</td>
<td>11.0 ± 1.5</td>
<td>21.9 ± 2.4</td>
<td>23.5 ± 0.6</td>
<td>42.0 ± 1.5</td>
</tr>
<tr>
<td><strong>NACS +Attn -Dep</strong></td>
<td>0.1 ± 0.1</td>
<td>0.6 ± 0.2</td>
<td>2.0 ± 0.2</td>
<td>3.2 ± 0.2</td>
<td>5.8 ± 1.1</td>
<td>10.9 ± 0.8</td>
</tr>
<tr>
<td><strong>L&amp;B SCAN</strong></td>
<td>0.1</td>
<td>0.1</td>
<td>4.1</td>
<td>15.3</td>
<td>70.2</td>
<td>89.9</td>
</tr>
</tbody>
</table>

Table 3: Results (BLEU) on the Machine Translation experiment for both directions using a GRU. See appendix for results using SRN and LSTM cells.

<table>
<thead>
<tr>
<th></th>
<th>En-Fr</th>
<th>Fr-En</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GRU +Attn</strong></td>
<td>32.1 ± 0.3</td>
<td>37.5 ± 0.6</td>
</tr>
<tr>
<td><strong>GRU -Attn -Dep</strong></td>
<td>30.2 ± 0.3</td>
<td>35.9 ± 0.3</td>
</tr>
</tbody>
</table>

Other observations. As expected, Table 1 shows that attention always helps. Generalizing to longer sequences is generally hard, and this remains an open problem.

### 8 Conclusion

In the quest for strong generalization, benchmarks measuring progress are an important component. The existing SCAN benchmark allows too simple models to shine, without the need for compositional generalization. We proposed NACS to remedy this. NACS still requires systematicity, while introducing stochasticity and strong dependencies on the target side. We argue that a good benchmark needs at least those properties, in order not to fall prey to trivial solutions, which do not work on more realistic use-cases for sequence-to-sequence models such as machine translation.
Acknowledgments

We would like to thank Brenden Lake and Marc’Aurelio Ranzato for useful discussions and feedback.

References


Supplementary Materials

\[
\begin{align*}
[\text{walk}] &= \text{WALK} \\
[\text{look}] &= \text{LOOK} \\
[\text{run}] &= \text{RUN} \\
[\text{jump}] &= \text{JUMP} \\
[\text{turn left}] &= \text{LTURN} \\
[\text{turn right}] &= \text{RTURN} \\
[\text{u left}] &= \text{LTURN} [u] \\
[\text{u right}] &= \text{RTURN} [u] \\
[\text{x twice}] &= [x] [x] \\
[\text{x thrice}] &= [x] [x] [x] \\
[\text{turn around left}] &= \text{LTURN} \text{LTURN} \text{LTURN} \text{LTURN} \\
[\text{turn around right}] &= \text{RTURN} \text{RTURN} \text{RTURN} \text{RTURN} \\
[\text{u around left}] &= \text{LTURN} [u] \text{LTURN} [u] \text{LTURN} [u] \text{LTURN} [u] \\
[\text{u around right}] &= \text{RTURN} [u] \text{RTURN} [u] \text{RTURN} [u] \text{RTURN} [u] \\
[\text{x}_1 \text{ and } x_2] &= [x_1] [x_2] \\
[\text{x}_1 \text{ after } x_2] &= [x_2] [x_1] \\
\end{align*}
\]

Figure 3: The interpretation functions for translating SCAN commands to actions.

<table>
<thead>
<tr>
<th></th>
<th>Simple</th>
<th>Length</th>
<th>Turn left</th>
<th>Jump</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN</td>
<td>75.6±5.4</td>
<td>0.2±0.0</td>
<td>26.7±12.8</td>
<td>0.0±0.0</td>
</tr>
<tr>
<td>GRU</td>
<td>100.0±0.0</td>
<td>14.4±0.8</td>
<td>53.4±11.7</td>
<td>0.0±0.0</td>
</tr>
<tr>
<td>LSTM</td>
<td>99.8±0.1</td>
<td>10.1±2.0</td>
<td>56.5±0.8</td>
<td>0.1±0.0</td>
</tr>
<tr>
<td>RNN +Attn</td>
<td>100.0±0.0</td>
<td>9.6±0.9</td>
<td>81.1±14.7</td>
<td>1.9±1.2</td>
</tr>
<tr>
<td>RNN +Attn-Dep</td>
<td>100.0±0.0</td>
<td>11.7±3.2</td>
<td>92.0±5.8</td>
<td>2.7±1.7</td>
</tr>
<tr>
<td>GRU +Attn</td>
<td>100.0±0.0</td>
<td>18.1±1.1</td>
<td>59.1±16.8</td>
<td>12.5±6.6</td>
</tr>
<tr>
<td>GRU +Attn-Dep</td>
<td>100.0±0.0</td>
<td>17.8±1.7</td>
<td>90.8±3.6</td>
<td>0.7±0.4</td>
</tr>
<tr>
<td>LSTM +Attn</td>
<td>100.0±0.0</td>
<td>15.6±1.6</td>
<td>83.8±16.8</td>
<td>9.7±2.9</td>
</tr>
<tr>
<td>LSTM +Attn-Dep</td>
<td>100.0±0.0</td>
<td>12.5±1.3</td>
<td>57.6±3.8</td>
<td>0.8±0.5</td>
</tr>
<tr>
<td>L&amp;B  best</td>
<td>99.8</td>
<td>20.8</td>
<td>90.3</td>
<td>1.2</td>
</tr>
<tr>
<td>L&amp;B  best overall</td>
<td>99.7</td>
<td>13.8</td>
<td>90.0</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 4: SCAN test scores on the simple, length, and primitive (turn left and jump) tasks. For ‘+Attn-Dep’ models we removed the connections from the previous target word embedding to the decoder state and the pre-output layer.
we removed the connections from the previous target word embedding to the decoder state and the pre-output layer.

Table 5: NACS test scores on the simple, length, and primitive (turn left and jump) tasks. For ‘+Attn-Dep’ models we removed the connections from the previous target word embedding to the decoder state and the pre-output layer.

<table>
<thead>
<tr>
<th></th>
<th>Simple</th>
<th>Length</th>
<th>Turn left</th>
<th>Jump</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN</td>
<td>26.9 ± 0.2</td>
<td>0.2 ± 0.1</td>
<td>26.4 ± 12.0</td>
<td>0.0 ± 0.0</td>
</tr>
<tr>
<td>GRU</td>
<td>99.0 ± 0.1</td>
<td>12.9 ± 1.2</td>
<td>47.5 ± 4.7</td>
<td>0.0 ± 0.0</td>
</tr>
<tr>
<td>LSTM</td>
<td>99.1 ± 0.1</td>
<td>10.9 ± 1.3</td>
<td>42.9 ± 2.9</td>
<td>0.0 ± 0.0</td>
</tr>
<tr>
<td>RNN + Attn-Dep</td>
<td>99.8 ± 0.1</td>
<td>19.4 ± 0.7</td>
<td>44.1 ± 0.9</td>
<td>0.3 ± 0.3</td>
</tr>
<tr>
<td>RNN + Attn-Dep</td>
<td>61.1 ± 0.3</td>
<td>0.5 ± 0.2</td>
<td>18.6 ± 1.0</td>
<td>0.0 ± 0.0</td>
</tr>
<tr>
<td>GRU + Attn-Dep</td>
<td>99.8 ± 0.1</td>
<td>17.2 ± 1.9</td>
<td>55.9 ± 3.5</td>
<td>0.0 ± 0.0</td>
</tr>
<tr>
<td>GRU + Attn-Dep</td>
<td>51.2 ± 1.2</td>
<td>2.0 ± 1.4</td>
<td>16.9 ± 1.2</td>
<td>0.0 ± 0.0</td>
</tr>
<tr>
<td>LSTM + Attn-Dep</td>
<td>99.1 ± 0.2</td>
<td>17.1 ± 2.0</td>
<td>48.3 ± 1.7</td>
<td>0.0 ± 0.0</td>
</tr>
<tr>
<td>LSTM + Attn-Dep</td>
<td>38.9 ± 0.9</td>
<td>1.0 ± 0.5</td>
<td>17.2 ± 1.2</td>
<td>0.0 ± 0.0</td>
</tr>
</tbody>
</table>

Table 6: SCAN test scores for jump with additional composed commands.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN</td>
<td>0.0 ± 0.0</td>
<td>0.0 ± 0.0</td>
<td>0.0 ± 0.0</td>
<td>0.1 ± 0.0</td>
<td>0.1 ± 0.1</td>
<td>0.5 ± 0.3</td>
<td>1.4 ± 0.3</td>
</tr>
<tr>
<td>GRU</td>
<td>0.1 ± 0.0</td>
<td>0.2 ± 0.1</td>
<td>0.6 ± 0.2</td>
<td>2.5 ± 1.1</td>
<td>3.3 ± 0.9</td>
<td>13.1 ± 2.4</td>
<td>42.4 ± 2.5</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.1 ± 0.0</td>
<td>0.3 ± 0.2</td>
<td>1.3 ± 0.2</td>
<td>3.8 ± 1.8</td>
<td>2.5 ± 1.1</td>
<td>6.5 ± 2.7</td>
<td>21.3 ± 1.4</td>
</tr>
<tr>
<td>RNN + Attn</td>
<td>3.5 ± 3.0</td>
<td>35.0 ± 2.8</td>
<td>48.6 ± 8.1</td>
<td>77.6 ± 2.6</td>
<td>89.2 ± 3.8</td>
<td>98.7 ± 1.3</td>
<td>99.8 ± 0.1</td>
</tr>
<tr>
<td>RNN + Attn-Dep</td>
<td>2.7 ± 1.7</td>
<td>29.5 ± 10.5</td>
<td>53.3 ± 10.2</td>
<td>82.4 ± 4.7</td>
<td>98.8 ± 0.8</td>
<td>99.8 ± 0.1</td>
<td>100.0 ± 0.0</td>
</tr>
<tr>
<td>GRU + Attn-Dep</td>
<td>12.5 ± 6.6</td>
<td>58.2 ± 12.0</td>
<td>67.8 ± 3.4</td>
<td>80.3 ± 7.0</td>
<td>88.0 ± 6.0</td>
<td>98.3 ± 1.8</td>
<td>99.6 ± 0.2</td>
</tr>
<tr>
<td>GRU + Attn-Dep</td>
<td>0.7 ± 0.4</td>
<td>70.9 ± 11.5</td>
<td>61.3 ± 13.5</td>
<td>83.5 ± 6.1</td>
<td>99.0 ± 0.4</td>
<td>99.7 ± 0.2</td>
<td>100.0 ± 0.0</td>
</tr>
<tr>
<td>LSTM + Attn</td>
<td>7.8 ± 0.9</td>
<td>40.2 ± 9.3</td>
<td>37.7 ± 10.7</td>
<td>50.3 ± 13.9</td>
<td>62.2 ± 7.7</td>
<td>94.0 ± 2.7</td>
<td>98.6 ± 1.0</td>
</tr>
<tr>
<td>LSTM + Attn-Dep</td>
<td>0.8 ± 0.6</td>
<td>39.0 ± 6.5</td>
<td>43.6 ± 17.6</td>
<td>66.0 ± 1.6</td>
<td>86.1 ± 2.3</td>
<td>98.7 ± 1.6</td>
<td>99.8 ± 0.2</td>
</tr>
<tr>
<td>L&amp;B</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>4.1</td>
<td>15.3</td>
<td>70.2</td>
<td>89.9</td>
</tr>
</tbody>
</table>

Table 7: NACS test scores for jump with additional composed commands.
Table 8: Results (BLEU) on the Machine Translation experiment for both directions.

<table>
<thead>
<tr>
<th>Model</th>
<th>En-Fr</th>
<th>Fr-En</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN +Attn</td>
<td>29.1 ± 0.4</td>
<td>34.9 ± 0.8</td>
</tr>
<tr>
<td>RNN +Attn-Dep</td>
<td>27.5 ± 0.7</td>
<td>32.9 ± 0.8</td>
</tr>
<tr>
<td>GRU +Attn</td>
<td>32.1 ± 0.3</td>
<td>37.5 ± 0.6</td>
</tr>
<tr>
<td>GRU +Attn-Dep</td>
<td>30.2 ± 0.3</td>
<td>35.9 ± 0.3</td>
</tr>
<tr>
<td>LSTM +Attn</td>
<td>31.5 ± 0.2</td>
<td>36.9 ± 1.1</td>
</tr>
<tr>
<td>LSTM +Attn-Dep</td>
<td>28.7 ± 0.2</td>
<td>34.0 ± 0.1</td>
</tr>
</tbody>
</table>

Table 9: Machine Translation: accuracy on eight novel sentences containing ‘daxy’ (‘daxiste’).

<table>
<thead>
<tr>
<th>Model</th>
<th>En-Fr</th>
<th>Fr-En</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN +Attn</td>
<td>79.2 ± 15.6</td>
<td>41.7 ± 5.9</td>
</tr>
<tr>
<td>RNN +Attn-Dep</td>
<td>66.7 ± 5.9</td>
<td>41.7 ± 5.9</td>
</tr>
<tr>
<td>GRU +Attn</td>
<td>70.8 ± 11.8</td>
<td>54.2 ± 5.9</td>
</tr>
<tr>
<td>GRU +Attn-Dep</td>
<td>58.3 ± 5.9</td>
<td>45.8 ± 11.8</td>
</tr>
<tr>
<td>LSTM +Attn</td>
<td>75.0 ± 10.2</td>
<td>41.7 ± 15.6</td>
</tr>
<tr>
<td>LSTM +Attn-Dep</td>
<td>50.0 ± 10.2</td>
<td>41.7 ± 5.9</td>
</tr>
</tbody>
</table>

Table 10: Machine Translation: accuracy on eight novel sentences containing ‘tired’ (‘fatigué’).

<table>
<thead>
<tr>
<th>Model</th>
<th>En-Fr</th>
<th>Fr-En</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN +Attn</td>
<td>66.7 ± 5.9</td>
<td>20.8 ± 5.9</td>
</tr>
<tr>
<td>RNN +Attn-Dep</td>
<td>66.7 ± 5.9</td>
<td>29.2 ± 15.6</td>
</tr>
<tr>
<td>GRU +Attn</td>
<td>62.5 ± 0.0</td>
<td>33.3 ± 5.9</td>
</tr>
<tr>
<td>GRU +Attn-Dep</td>
<td>66.7 ± 5.9</td>
<td>25.0 ± 20.4</td>
</tr>
<tr>
<td>LSTM +Attn</td>
<td>66.7 ± 5.9</td>
<td>25.0 ± 10.2</td>
</tr>
<tr>
<td>LSTM +Attn-Dep</td>
<td>62.5 ± 0.0</td>
<td>25.0 ± 17.7</td>
</tr>
</tbody>
</table>