LET's Test Compositionality

Michael Neely
University of Amsterdam
Student number: 12547190
michael.neely@student.uva.nl

Leila Talha
University of Amsterdam
Student number: 10756922
leila.talha@student.uva.nl

Abstract

In this research, we evaluate the compositionality of a recurrent and an attention-based neural model in terms of both localism and systematicity in the context of a sequence-to-sequence machine translation task on a new artificial compositional language. Models achieve high performance on the task, but fail to exhibit compositional behavior.

1 Introduction

In many Natural Language Processing (NLP) tasks, data-driven deep learning models have proven more effective than their traditional, symbolic counterparts. Neural networks can process large amounts of data and are therefore more robust to the intrinsic noise of natural language. However, does their success on isolated tasks mean they genuinely understand the languages they are processing? Or perhaps, are they just exploiting statistical patterns?

The principle of compositionality is a fundamental feature of natural language and serves as a rigorous test of the robustness of a model's learned representation. In its most broad definition, compositionality refers to "the meaning of a whole is a function of the meanings of the parts and of the way they are syntactically combined" (Kamp and Partee, 1995). Note that this statement only refers to language itself, not the behavior of an entity using the language. To address this gap, (Hupkes et al., 2019) propose a suite of behavioral tests designed to evaluate the compositionality of a neural model across five dimensions. Of the five aspects, systematicity — whether models systematically recombine known parts and rules — and localism — whether models' composition operations are local or global — are of particular note.

In this research, we evaluate the compositionality of recurrent and attention-based neural models in terms of both localism and systematicity in the context of a sequence-to-sequence machine translation task on an artificial compositional language, which we call PFCG-LET.

2 Related Work

It is common practice to evaluate the compositionality of neural models in isolation by carefully designing artificial languages which exhibit desirable compositional phenomena.

(Bowman et al., 2015) present a logical inference task with a limited vocabulary of six unanalyzed word types and three logical relationships. They examine sequence and tree-based versions of recurrent (RNN) and gated-recurrent (LSTM) neural networks and demonstrate the capability of models to generalize effectively. They note the bias of the tree-based models as a factor that significantly improves efficiency, highlighting the importance of selecting the appropriate model architecture for a given task.

(Veldhoen et al., 2016) define an arithmetic language for adding and subtracting integers. Noting the existence of two effective strategies to solve the task — recursive and iterative — they examine the internal structure of both categories of models. They find that their tree-based model adopts a recursive strategy, processing sequences through a repeated application of project, sum, and squash steps. Using diagnostic classifiers, they find their sequential models to favor an iterative approach that struggles with right-branching sequences.

(Lake and Baroni, 2017) propose the recursionless SCAN grammar, featuring navigational commands which are turned into action sequences through the compositional application of interpretation functions. They report sequence-to-sequence recurrent models to be highly unsystematic in their completion of the task. Their conclusion is made more optimistic by (Loula et al., 2018) and (Dessì and Baroni, 2019), which find more evidence of systematic behavior by rearranging splits and using convolutional networks.

Finally, (Hupkes et al., 2019) introduce the sequence-to-sequence probabilistic context-free grammar string-edit-task (PCFG-SET), which features unary and binary functions for string manipulation. They analyze gated recurrent (LSTM), convolutional, and transformer models through the aforementioned compositionality test suite.

3 Approach

Motivated by (Hupkes et al., 2019), we introduce a new artificial sequence-to-sequence translation task, which we dub the list-edit-task (PCFG-LET).

3.1 PCFG-LET: syntax and semantics

Like (Hupkes et al., 2019), the input alphabet of PCFG-LET consists of four categories of words: numerical characters which define a list, unary and binary functions which manipulate those lists, a comma (,) which separates binary arguments, and a special symbol `E` that represents an empty list. These words are combined to generate input sequences that define a series of operations applied to the list argument(s). Note that the empty token can only appear in intermediate or output sequences.

The task for a particular network is then to translate inputs into interpreted outputs by recursively applying the interpretation functions. Success should only be possible if networks are **local** in their operation and **systematic** in their approach since the functions are highly sensitive to the order of lists and arguments. The grammar and interpretation functions are shown in figures 1 and 2, respectively. The functions of the list-edit-task prevent sequence explosion, the phenomenon in which applications of interpretation functions lead to excessively long sequences. (Hupkes et al., 2019) highlight this as an unfortunate side-effect of their PCFG-SET.

Non-terminal rules			
S	$\rightarrow$	$F_U\ S \mid F_B\ S\ ,\ S$	0.3 | 0.3
S	$\rightarrow$	X	0.4
X	$\rightarrow$	$X\ X$	0.45
Lexical rules			
F_U	$\rightarrow$	`min` | `max` | `mirror` | `remove_repeated` | `remove_unique`	0.2 | ... | 0.2
F_B	$\rightarrow$	`filter` | `difference` | `intersection` | `union`	0.25 | ... | 0.25
X	$\rightarrow$	1 | 2 | ... | 518 | 519	0.001 | ... | 0.001

Figure 1: The probabilistic context free grammar that defines all possible input sequences in the PCFG-LET task

3.2 PCFG-LET: data

Like (Hupkes et al., 2019), we select the size of the numerical alphabet to be 520 (the numbers `1` through `519` and the empty token `E`) and create a dataset of 100,000 (one-hundred-thousand) input-output pairs by sampling from the grammar. To eliminate any possibility of memorization, we ensure each sample is unique. By unique, we mean

Unary functions F_U:		
`min` $x_1\ x_2\ \ldots\ x_n$	$\rightarrow$	$x_i, \forall_{i,j}\ x_i \leq x_j$
`max` $x_1\ x_2\ \ldots\ x_n$	$\rightarrow$	$x_i, \forall_{i,j}\ x_i \geq x_j$
`mirror` $x_1\ x_2\ \ldots\ x_c\ \ldots\ x_{n-1}\ x_n$	$\rightarrow$	$\ldots\ \ x_2\ x_1 x_c\ x_n\ x_{n-1}\ \ldots$
`unique` $x_1\ x_2\ \ldots\ x_n = \boldsymbol{x}$	$\rightarrow$	$\{\boldsymbol{x}\}$
`remove_repeated` $x_1\ x_2\ \ldots\ x_n$	$\rightarrow$	$\{x_i \mid \forall_{j \neq i}\ x_j \neq x_i\}$
`remove_unique` $x_1\ x_2\ \ldots\ x_n$	$\rightarrow$	$\{x_i \mid \exists_{j \neq i}\ x_j = x_i\}$
Binary functions F_B:		
`filter` $\boldsymbol{x}, \boldsymbol{y}$	$\rightarrow$	$\{\boldsymbol{x}\} \setminus \{\boldsymbol{y}\}$
`difference` $\boldsymbol{x}, \boldsymbol{y}$	$\rightarrow$	$(\{\boldsymbol{x}\} \setminus \{\boldsymbol{y}\}) \cup (\{\boldsymbol{y}\} \setminus \{\boldsymbol{x}\})$
`intersection` $\boldsymbol{x}, \boldsymbol{y}$	$\rightarrow$	$\{\boldsymbol{x}\} \cap \{\boldsymbol{y}\}$
`union` $\boldsymbol{x}, \boldsymbol{y}$	$\rightarrow$	$\{\boldsymbol{x}\} \cup \{\boldsymbol{y}\}$

Figure 2: The interpretation functions for the PCFG-LET input sequences. The notation $\{.\}$ denotes a transformation from a list to a set, and the subscripts i and j denote the items enumerated on the left-hand side of the rules, for which conditions specified in the rules for unary functions should hold. Except for mirror, all functions return a set (possibly a singleton, or empty).

that the arguments to any given function are **never repeated**. We do **not** perform any form of naturalization, and do **not** limit the length of list arguments given to the functions, but do cap sequence length to 50 (fifty) tokens for performance. The distribution of token length and function depth is displayed in Appendix A. Due to the lack of naturalization, sequences tend to be short in both length and depth. Data is split into portions of 85%, 5%, and 10% for training, validation, and testing.

3.3 Testing localism

We test the localism of trained models similarly to (Hupkes et al., 2019) by unrolling computations and comparing model's successive local predictions to their global ones. The final prediction is **consistent** if it matches the global prediction, and **accurate** if it matches the target.

3.4 Testing systematicity

Again, like (Hupkes et al., 2019), we investigate the ability of models to recombine known parts and rules by focusing on words w_1 and w_2 that never co-occur in the training corpus. We decompose the systematic treatment of input sequences into three parts: the **empty token**, **numbers**, and **functions**.

To accomplish this, new models are trained on a dataset manipulated to prevent `mirror` from co-occurring with: (1) the empty token `E`, (2) the numbers `7` and `13`, and (3) the `union` function, respectively. We first test the models on a similar but non-overlapping test set to form a baseline, we then evaluate each sub-task by measuring sequence accuracy on test sets where every sequence includes the pair of words w_1 and w_2 excluded in the training set. The manipulated datasets use a smaller vocabulary size of 27 but are otherwise equivalent to the original PCFG-LET task in terms of both command length and depth.

4 Experiments and Results

Models examined in the literature tend to fall into three categories: recurrent, convolutional, or purely attention-based. Initial experimentation with convolutional models led to consistently poor results. Taking this as evidence of the order-sensitive nature of the task, we chose to study the other architectures, which we implemented using the popular OpenNMT-py (Klein et al., 2017) library.

4.1 Model 1: LSTMS2S

The sequential processing nature of LSTM models (Hochreiter and Schmidhuber, 1997) is ideal for PCFG-LET. We select a fully recurrent, bidirectional model with attention similar to the LSTMS2S architecture of (Hupkes et al., 2019). With careful hyperparameter tuning, we decide on a 512-dimensional word vector size with scaled dot-product attention (Vaswani et al., 2017) and a batch size of 64 sequences. We train the model for 25 epochs or until convergence (five successive epochs with no improvement in word accuracy and perplexity on the validation set), using the Adam optimizer (Kingma and Ba, 2014) with a standard learning rate of 0.001.

4.2 Model 2: Transformer

Transformers (Vaswani et al., 2017) discard recurrent cells in favor of a purely attention-based approach. Such a design is advantageous for processing longer sequences. By inferring order from position encodings, the cost of relating tokens remains uniform regardless of their distance. Unlike LSTMs, additional stacked layers in a transformer improve hierarchical modeling capability. A drawback of the Transformer model is its sensitivity to hyperparameter tuning. We experimented with different settings, but ultimately found the set chosen by (Hupkes et al., 2019), to be optimal. The only difference between our Transformer Model and Hupkes et al. 's is a reduced word vector dimensionality of 256. The transformer model is trained for 25 epochs (or until convergence), similarly to the LSTMS2S.

4.3 Task Accuracy

Results are shown in table 1. Models achieve higher accuracy scores than (Hupkes et al., 2019), which is likely due to the lower average length and depth of PCFG-LET sequences. Consistent with the results of (Hupkes et al., 2019), models struggle with complex sequences of higher length and depth (see figure 3).

Experiment	LSTMS2S	Transformer
Task Accuracy*	0.8839 ± 0.0051	0.8569 ± 0.0010
Localism*	0.6704 ± 0.0079	0.6371 ± 0.0097
Localism†	0.7232 ± 0.0066	0.6925 ± 0.0096
Systematicity*, baseline	0.7150 ± 0.0014	0.7487 ± 0.0090
Systematicity*, empty token	0.4233 ± 0.0047	0.2237 ± 0.1628
Systematicity*, numbers	0.4190 ± 0.0050	0.3803 ± 0.0135
Systematicity*, functions	0.4687 ± 0.0091	0.4317 ± 0.0155

Table 1: General task performance and performance per test for PCFG-LET, averaged over three runs with standard deviation indicated. Sequence accuracy is denoted with *, and consistency score with †.

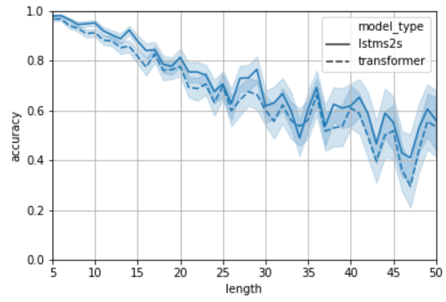

Figure 3: Average sequence accuracy as a function of input length. A similar pattern is seen with depth.

4.4 Localism

Consistency scores are inversely correlated with the length of input sequences, depth of input sequences, and length of target sequences. At first glance, both models appear to achieve higher consistency scores than those reported by (Hupkes et al., 2019). However, when exclusively considering sequences of depth greater than one, the LSTMS2S only achieves an average consistency score of 0.4493 ± 0.0131, and the Transformer, a score of 0.3883 ± 0.0192.

4.5 Systematicity

On the baseline test, both models achieve reasonable results in terms of sequence accuracy, with the Transformer outperforming LSTMS2S. For the three subtests of systematicity however, the LSTMS2S consistently outperforms the Transformer with an average sequence accuracy of 0.4370 ± 0.0066 and 0.3452 ± 0.0947, respectively. This is a significant drop in performance compared to the baseline for both models.

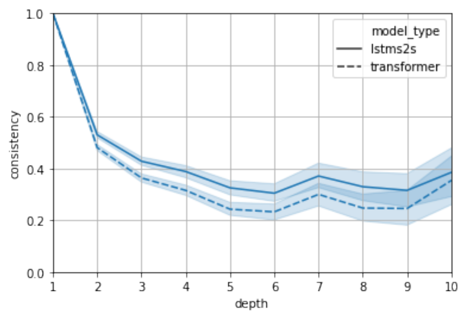

Figure 4: Average consistency score as a function of input depth.

5 Discussion

The LSTMS2S performs better than the Transformer in almost every test, invalidating our earlier assumptions. This discrepancy could be due to the order-sensitive nature of the task, the short average sequence length, or insufficient Transformer hyperparameter tuning. Regardless of these factors, and despite high task accuracy scores, it is clear both models do not exhibit compositional behavior. In the localism test, models suffer an average drop in accuracy of 25% when performing unrolled computations. Consistency and performance scores on deep sequences never rise above 50%, indicating the model is either not applying a recursive strategy or is struggling with interpretation functions.

To further test this hypothesis, we examine model performance on individual functions. The high scores on `remove_unique` and `intersection` shown in figure 5 are surprising, since we consider these to be more difficult than functions which have lower scores like `max` and `min`. We probe further by examining individual function performance where the correct target is not an empty token. Figure 6 reveals that models adopt a strong bias for predicting empty tokens as the output of these functions. So strong, they fail to predict the correct output almost every time. There are 491 primitive (the only function in the sequence) instances of `remove_unique` in the training set which do not map to an empty token, and 111 such instances of `intersection`. Poor model performance on these functions could be fixed by ensuring a balanced distribution of instances. However, truly compositional models should be able to infer the behavior of interpretation functions from a small number of examples.

In the systematicity test, performance of the LSTMS2S on the subtests with numbers and the empty token does not differ significantly, as opposed to performance of the Transformer. This could indicate that a recurrent-based model is more likely to learn compositionally than a model that is attention-based. However, performance worsens by 39% and 54% on average for LSTMS2S and the Transformer, respectively. This drop could be explained by the selection of the `mirror` function for the tests and having provided an insufficient amount of examples to learn its interpretation. The correct output of the `mirror` function is difficult to predict since it is the only interpretation function in PCFG-LET that usually returns a list with the same number of elements as its input. A different, more likely, explanation is that the models simply fail to generalise in a compositional manner.

Based on our experiments in the setting of PCFG-LET, we conclude the tested models resort to naive exploitation of statistical patterns. In future work, we would like to investigate Memory-Augmented RNNs. Since these networks are capable of learning Dyck languages and solving palindrome and string reversal tasks (Suzgun et al., 2019), we hypothesize their increased modeling capacity will lead to improved representations of hierarchical composition.

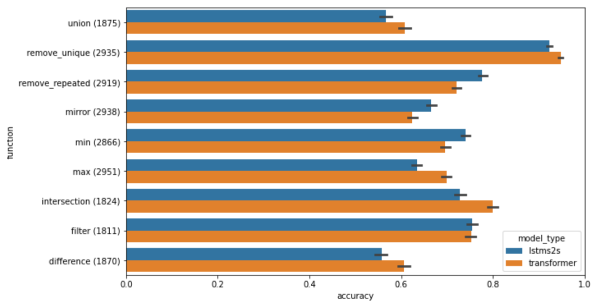

Figure 5: Average sequence accuracy per function (number of occurrences shown in parentheses)

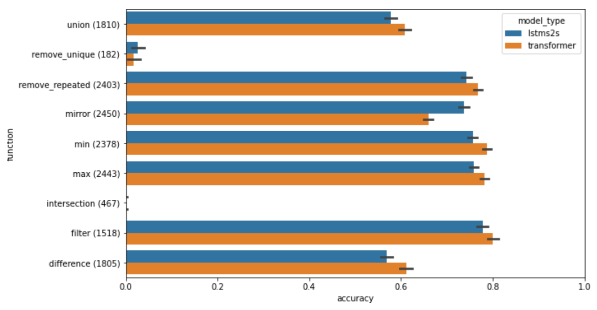

Figure 6: Average sequence accuracy per function where the target is not an empty token (number of occurrences shown in parentheses)

References

Samuel R. Bowman, Christopher D. Manning, and Christopher Potts. 2015. Tree-structured composition in neural networks without tree-structured architectures.

Roberto Dessì and Marco Baroni. 2019. CNNs found to jump around more skillfully than RNNs: Compositional generalization in seq2seq convolutional networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3919–3923, Florence, Italy. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2019. Compositionality decomposed: how do neural networks generalise?

Hans Kamp and Barbara Partee. 1995. Prototype theory and compositionality. *Cognition*, 57(2):129 – 191.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Brenden M. Lake and Marco Baroni. 2017. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks.

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

Mirac Suzgun, Sebastian Gehrmann, Yonatan Belinkov, and Stuart M. Shieber. 2019. Memory-augmented recurrent neural networks can learn generalized dyck languages.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Sara Veldhoen, Dieuwke Hupkes, and Willem Zuidema. 2016. Diagnostic classifiers: revealing how neural networks process hierarchical structure.

A Distribution of lengths and depths of PCFG-LET commands

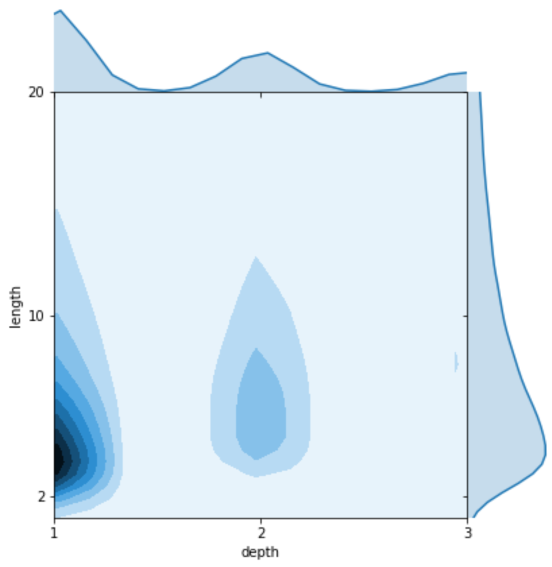

Figure 7: Distribution of lengths and depths in the generated PCFG-LET dataset.