Induced Natural Language Rationales and Interleaved Markup Tokens Enable Extrapolation in Large Language Models

Mirelle Bueno
University of Campinas

Carlos Gemmell
University of Glasgow

Jeffrey Dalton
University of Glasgow

Roberto Lotufo
University of Campinas
NeuralMind

Rodrigo Nogueira
University of Campinas
NeuralMind

Abstract

The ability to extrapolate, i.e., to make predictions on sequences that are longer than those presented as training examples, is a challenging problem for current deep learning models. Recent work shows that this limitation persists in state-of-the-art Transformer-based models. Most solutions to this problem use specific architectures or training methods that do not generalize to other tasks. We demonstrate that large language models can succeed in extrapolation without modifying their architecture or training procedure. Our experimental results show that generating step-by-step rationales and introducing marker tokens are both required for effective extrapolation. First, we induce a language model to produce step-by-step rationales before outputting the answer to effectively communicate the task to the model. However, as sequences become longer, we find that current models struggle to keep track of token positions. To address this issue, we interleave output tokens with markup tokens that act as explicit positional and counting symbols. Our findings show how these two complementary approaches enable remarkable sequence extrapolation and highlight a limitation of current architectures to effectively generalize without explicit surface form guidance. Code available at `https://github.com/MirelleB/induced-rationales-markup-tokens`

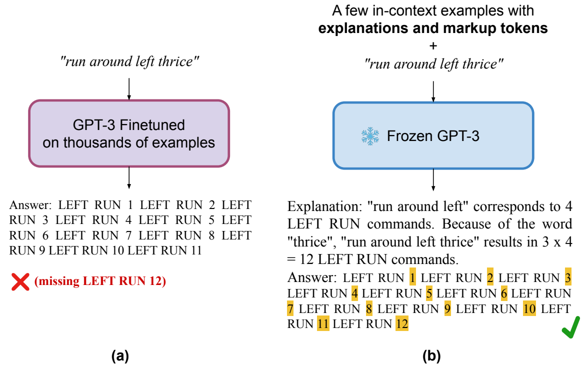

Figure 1: Answers produced by a GPT-3 model on the "length" split of the SCAN dataset when (a) fine-tuned on thousands of examples vs (b) induced via a few in-context examples to generate explanations and markup tokens (in yellow).

1 Introduction

The lack of compositional generalization of neural networks has been a long-standing limitation known for decades (Fodor and Pylyshyn, 1988; Schmidhuber, 1990; Marcus, 1998, 2018; Lake and Baroni, 2018; Liška et al., 2018; Keysers et al., 2019). This is often associated with their failure to extrapolate, i.e., the ability to work on sequences that are longer than those presented as training examples. Modern architectures such as the Transformer (Vaswani et al., 2017), which is the core component of state-of-the-art NLP models, perform poorly on this class of problems (Bhattamishra et al., 2020; Nogueira et al., 2021; Wang et al., 2021; Pal and Baral, 2021; Welleck et al., 2021; Bogin et al., 2022; Finlayson et al., 2022; Mittal et al., 2021). In Figure 1-(a), we illustrate how recent large language models such as GPT-3 fail at this task, even when fine-tuned on thousands of examples.

Architectures and training methods that target this specific problem are often developed based on synthetic tasks whose creation rules are known (Das et al., 1992; Li et al., 2019b; Russin et al., 2019; Andreas, 2020; Liu et al., 2020a; Chen et al., 2020; Herzig and Berant, 2021; Shaw et al., 2021; Zhu et al., 2021). Thus, they resort to techniques such as augmenting the training data or biasing the model's architecture to internally represent these rules. However, improvements obtained on one compositional generalization benchmark do not transfer to others (Furrer et al., 2020), i.e., they lose their ability to be used as competitive general-purpose models in real tasks, as these can seldom be solved with a small set of rules.

We study the behavior of Transformer models

Proceedings of the 1st Workshop on Mathematical Natural Language Processing (MathNLP), pages 17 - 24
December 8, 2022 ©2022 Association for Computational Linguistics

and demonstrate that this problem is not due to an intrinsic limitation of their training algorithm. We show that inducing autoregressive models to rationalize before making a prediction (Wang et al., 2022; Zelikman et al., 2022) is not enough to extrapolate on long sequences: to solve it, we introduce markup tokens (Nogueira et al., 2021; Kim et al., 2021). The two general approaches together allow the models to achieve remarkable extrapolation generalization without requiring changes to the model or architecture. These findings provide evidence that general-purpose models have the ability to both improve their effectiveness and interpretability *at the same time*. The need to markup tokens also suggests there are fundamental issues that need to be addressed in the Transformer architecture, particularly the need for better positional representations. Thus, our study confirms and supports recent results from previous work that positional embeddings used in current state-of-the-art Transformer models cannot precisely track of token positions or perform precise counting (Liu et al., 2020b; Thawani et al., 2021; Press et al., 2022).

2 Related Work

A long list of architectures and training methods attempt to improve the extrapolation capabilities of deep learning models. For instance, some are specifically designed to solve only a handful of tasks (Singh, 1992; Kaiser and Sutskever, 2015; Kalchbrenner et al., 2015; Price et al., 2016; Andreas et al., 2016, 2017; Trask et al., 2018). Pre-trained word embeddings find it difficult to extrapolate to unseen numbers in training (Wallace et al., 2019). Alternatives to improving the extrapolation ability of neural models include building neural models with a pre-training corpus of numerical text (Geva et al., 2020) or using scientific notation to represent numbers (Zhang et al., 2020). Likewise, better numerical and compositional skills can be achieved by supplementing input texts with pre-computed numerical calculations (Andor et al., 2019) or explicitly assuming rules or mathematical equations from natural language texts (Liu et al., 2019; Li et al., 2019a; Zou and Lu, 2019a,b; Shi, 2020; Qiu et al., 2021). Many of these models are capable of adding numbers larger than those seen during training. In contrast, more general-purpose architectures fail to extrapolate on numerical tasks (Joulin and Mikolov, 2015; Dehghani et al., 2018; Schlag et al., 2019).

Our work derives from recent findings that show that inducing the model to generate explanations in natural language leads to better performance in a wide variety of tasks (Recchia, 2021; Fernandes et al., 2022; Wang et al., 2022; Zelikman et al., 2022; Nye et al., 2022; Katz et al., 2022; Zhou et al., 2022; Khot et al., 2022). In particular, the work proposed by (Zhou et al., 2022) achieves state-of-the-art results in the extrapolation of tasks involving symbolic manipulation, compositional generalization and numerical reasoning. Tasks are solved via few-shot learning applied to a large language model (e.g. text-davinci-002) in two main steps. The first step consists of reducing the question into sub-questions, then, in the second phase, a new interaction is made with the model, now solving sequentially the sub-questions generated in the previous step.

The results shown in Zhou et al. (Zhou et al., 2022) corroborate our intuition that explanations alone are not enough to achieve extrapolations. By inducing the model to generate explanations *and* markup tokens, *we provide evidence that compositional generalization can be achieved without sacrificing the general applicability on other tasks*, which is often a feature that is lost with architectural modifications.

However, a limitation of Zhou et al.'s and our method is that both require a programmatic post-processing step: Zhou et al. use a python script to convert the model output (e.g., `3*["LEFT"]`), which is in python notation, into the expected format of the final answer (e.g., `LEFT LEFT LEFT`); in our method, we programmatically remove the markup tokens from the final answer. We argue that the need to call an external script exposes a limitation in the current Transformer architecture, namely, that it cannot handle long sequences of repeated tokens.

3 Methodology

In this section, we describe our proposed method for inducing explanations and markup tokens using in-context learning with a few examples. We first create a prompt $ic||oc$ that concatenates in-context training examples ic with a test example oc. The ic examples consist of N triples of "Instruction", "Explanation" and "Output", i.e., $ic = \{(i_1^*, e_1^*, o_1^*), ..., (i_N^*, e_N^*, o_N^*)\}$. The test example oc is made of only the "Instruction" field. When we feed $ic||oc$ to a language model, it should

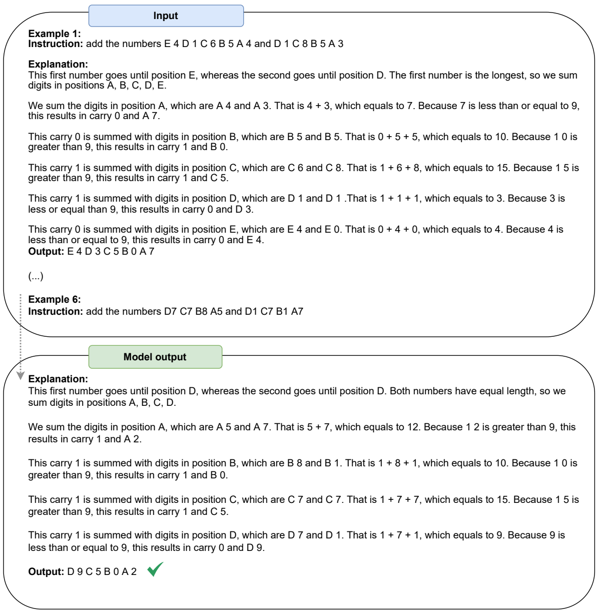

Figure 2: Example of a few-shot prompt and model completion for the addition task. First, a prompt composed of in-context (ic) samples are given, which are formed by $\{input, explanation, output\}$ triplets concatenated with an out-of-context (oc) test example that has only the "instruction" field. The model then completes the "explanation" and "output" fields from the test example as a result.

generate the remaining "Explanation" and "Output" fields for oc. Figure 2 illustrates the input prompt given to the model and the (correct) output given by the model.

We also interleave the tokens ic and oc with markup tokens that help the model to precisely identify the tokens in the input and output sequences (see Figure 1-(b) for an example). These tokens support the model in three ways: 1) They act as a form of working memory to indicate progress being made. 2) They act as sub-prompt anchors to inform the start of a known pattern. 3) They implicitly model a stopping condition should a certain amount of progress be reached. We programmatically include these markups in each test input and remove them from the output answers before comparing them with ground-truth ones.

Due to its few-shot nature, our method can be adjusted for different tasks. Likewise, our approach does not require any additional modifications to the language model such as pretraining or changes to the loss function.

4 Experimental Setup

We evaluate our method in two tasks that require extrapolation: 1) the length split of the SCAN data (Lake and Baroni, 2018) and 2) the addition of two numbers. In all experiments, we used the `text-davinci-002` model, available via a paid API provided by OpenAI. We report the accuracy of the test set.

4.1 SCAN

The SCAN synthetic dataset translates simple navigation commands into a sequence of actions (e.g., the input `jump thrice` results in the output `JUMP JUMP JUMP`). These commands are generated from the composition of a specific grammar, combining "primitive" commands such as `jump`, `walk`, `look`, `run` and `turn`; "modifiers" (`left`, `right`, `around`, `opposite`); repetition symbols like `twice`/`thrice`; "combiners" (`and`/`after`) that group two action sequences.

To construct the prompt, we generated nine in-context training examples, each made of three parts: an instruction, an explanation, and the desired output. The "Instruction" is a sequence of commands while the "Explanation" is a description, in natural language detailing the steps to generate the output. The "Output" corresponds to the expected answer to the instruction. In addition, in the output field, we inject markup tokens to delimit the end of a repeating sequence or sub-instruction. Therefore, to indicate each repetition of a given action, we use positive integers and at the end of a sequence of actions, we use the separator `||`. For example, for the input: `jump twice and walk twice`, we generate the output `JUMP 1 JUMP 2 || WALK 1 WALK 2`.

The target outputs of training examples have up to 22 actions. The test examples were drawn from the "Length" split provided by the authors.* This set has 3,920 examples whose target output varies between 24 and 48 actions. The instruction (input) of each test example is appended to the in-context training examples and the model is prompted to generate the "Explanation" and "Output" fields. Thus, since training examples are shorter than test ones, we are able to assess the compositional generalization of the model while extrapolating to larger unseen sequences. Due to the cost of using the GPT-3 API (approximately 0.10 USD per example),

*https://github.com/brendenlake/SCAN

Method	Acc.
Specialized Architectures	
Syntactic Attn. (Russin et al., 2019)	15.2
CGPS (Li et al., 2019b)	20.3
T5-base DUEL (Zhu et al., 2021)	45.0
LANE (Liu et al., 2020a)	100.0
NSSM (Chen et al., 2020)	100.0
SBSP (Herzig and Berant, 2021)	100.0
NQG (Shaw et al., 2021)	100.0
Synth (Nye et al., 2020)	100.0
General-purpose Architectures	
T5-base (Furrer et al., 2020)	14.4
T5-Large (Furrer et al., 2020)	5.2
T5-3B (Furrer et al., 2020)	3.3
T5-11B (Furrer et al., 2020)	2.0
GPT-3 Ada - fine-tuned	13.9
GPT-3 Curie - fine-tuned	6.4
GPT-3 Davinci - fine-tuned	8.2
Least-to-Most (Zhou et al., 2022)	99.7
—	
Ours (rationales only)	2.5
Ours (markups only)	22.5
Ours (rationales + markups, inverted prompt)	30.0
Ours (rationales + markups)	95.2

Table 1: Results on the "length" split of the SCAN dataset.

we evaluated the model on 400 randomly sampled examples from the test set.

4.2 Addition Task

Extrapolation abilities can also be tested with arithmetic tasks. For this, we built a prompt for the addition operation, where we present five in-context training examples with two numbers up to 5 digits and ask the model to generate the explanation and answer for a test set example made of numbers with 4 to 14 digits. We evaluate the model on 400 test samples automatically generated by the "balanced sampling" method from Nogueira et al. (Nogueira et al., 2021), which ensures that the set will have a roughly equal proportion of answers with d-digit numbers, with $d \in [4, 14]$.

We use a template similar to SCAN's to feed the in-context examples to the model. We manually generate the explanations for the training examples and inject markup tokens in the instructions and the target output. In the expected output, these tokens are used during the explanation steps. We illustrate in Figure 2 an example of a prompt followed by a completion of the model.

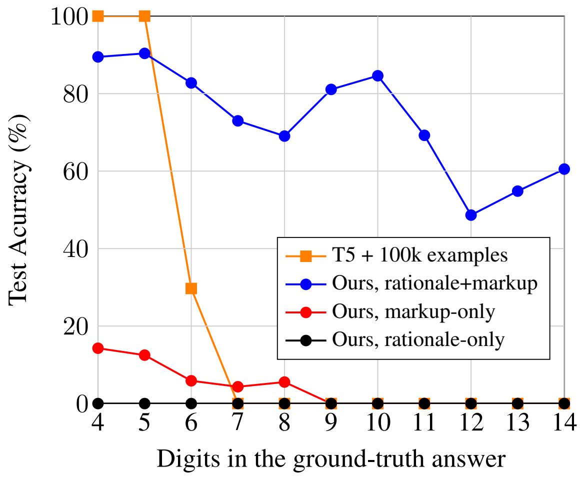

Figure 3: Test set accuracy in the addition task vs number of digits in the ground-truth answer.

5 Results

In Table 1 we show the results for the length split of the SCAN dataset. We see that specialized models like LANE, NSSM, and SBSP solve the compositional generalization proposed by SCAN, whereas generic architectures such as T5 (Raffel et al., 2020) or GPT-3 (Brown et al., 2020) fine-tuned on the task have poor performance.

We also show results for GPT-3's Ada (300M parameters), Curie (6B parameters) and Davinci (175B parameters) models fine-tuned on all 16,990 training examples of the SCAN dataset for 3 epochs. In these cases, we do not use in-context examples, explanations, or markup tokens. Our methodology of providing prompts with detailed explanations was shown to be more effective than finetuning on thousands of examples.

The same behavior is also observed in the addition task, as seen in Figure 3. Our approach with explanations and markup tokens (rationale + markup) shows that even with as few as 5 examples, the model can perform the task of adding numbers with more than 5 digits, reaching a performance of around 60% in numbers with up to 14 digits and an average accuracy of 73% considering all 400 examples in the test set.

We also investigated the performance of fine-tuning a general-purpose model on this task. We trained a T5-base with 100K samples on numbers with 2 to 5 digits per 10 epochs without adding explanations. We observe that the model reaches 100% accuracy with numbers of up to 5 digits, but fails to add numbers with more than 6 digits.

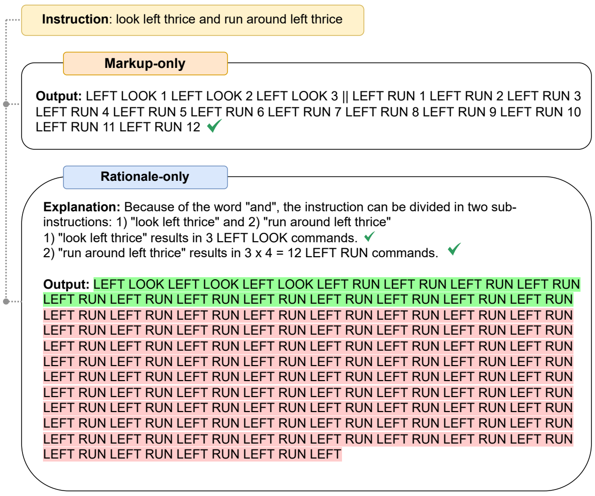

Figure 4: Model output differences between the markup-only and rationale-only approaches.

5.1 Ablation: Rationale-only vs. markup-only

We also investigate the impact of using explanation and markup tokens in isolation. We compare two scenarios: prompts without explanation (markups-only) and without markup tokens (rationales-only).

In Table 1, we see that the rationale-only and markup-only approaches have significantly lower test accuracy, demonstrating that it is not enough to explain how to solve the task, but it is also important to inject markup tokens. We believe that these tokens help the model generate repeated sequences of tokens.

In Figure 4, we provide qualitative evidence of this hypothesis: Without markup tokens, the model correctly generates the explanation but fails to finish the action sequence, therefore entering a loop.

5.2 Ablation: Inverted prompt

We also experimented with reversing the order in which the "explanations" and "outputs" fields are presented to the model. Therefore we provide the expected output first and then the explanation. The idea of this experiment was to verify if the order explanation followed by the output has an impact on the generation of the answer. In Table 1 we see that the performance drops from 95.2 to 30% (rationales + markups - inverted prompt). This empirical result agrees with the literature in terms that the model possibly processes the explanation before determining the final output.

6 Conclusion

In this work, we show how step-by-step rationales and positional markup tokens enable general-purpose architectures to extrapolate to sequences that are significantly longer than those provided as training examples. Rationales before the answer break down the problem into small executable chunks and markup tokens track the working progress as the output is generated. Importantly, we show how these methods are complementary and, when used together, enable remarkable extrapolation results on two synthetic tasks.

However, we note the use of markup tokens as a limitation of current models and subword tokenizers. Future models should be able to count tokens and keep track of individual tokens in long sequences without resorting to additional supporting tokens. As our qualitative analysis shows, most failure cases are due to one or two tokens generated incorrectly. We see the ability to automatically verify these errors, as proposed by Cobbe et al. (Cobbe et al., 2021), as a promising direction to improve the extrapolation capabilities of current models.

Acknowledgments

This research was partially funded by grants 2020/09753-5 and 2022/01640-2 from Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP). This work is also supported by grant EP/V025708/1 from the Engineering and Physical Sciences Research Council. We also thank Google Cloud for credits to support this work. R Lotufo is partially supported by CNPq (The Brazilian National Council for Scientific and Technological Development) under grant 310828/2018-0.

References

Daniel Andor, Luheng He, Kenton Lee, and Emily Pitler. 2019. Giving BERT a calculator: Finding operations and arguments with reading comprehension. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5949–5954.

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, Online. Association for Computational Linguistics.

Jacob Andreas, Dan Klein, and Sergey Levine. 2017. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pages 166–175. PMLR.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1545–1554.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. 2020. On the ability and limitations of transformers to recognize formal languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7096–7116.

Ben Bogin, Shivanshu Gupta, and Jonathan Berant. 2022. Unobserved local structures make compositional generalization hard. *arXiv preprint arXiv:2201.05899*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Xinyun Chen, Chen Liang, Adams Wei Yu, Dawn Song, and Denny Zhou. 2020. Compositional generalization via neural-symbolic stack machines. *Advances in Neural Information Processing Systems*, 33:1690–1701.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Sreerupa Das, C. Giles, and Gordon Sun. 1992. Learning context-free grammars: Capabilities and limitations of a recurrent neural network with an external stack memory.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2018. Universal transformers. In *International Conference on Learning Representations*.

Patrick Fernandes, Marcos Treviso, Danish Pruthi, André FT Martins, and Graham Neubig. 2022. Learning to scaffold: Optimizing model explanations for teaching. *arXiv preprint arXiv:2204.10810*.

Matthew Finlayson, Kyle Richardson, Ashish Sabharwal, and Peter Clark. 2022. What makes instruction learning hard? an investigation and a new challenge in a synthetic environment. *arXiv preprint arXiv:2204.09148*.

Jerry A Fodor and Zenon W Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. 2020. Compositional generalization in semantic parsing: Pre-training vs. specialized architectures. *arXiv preprint arXiv:2007.08970*.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 946–958.

Jonathan Herzig and Jonathan Berant. 2021. Span-based semantic parsing for compositional generalization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 908–921, Online. Association for Computational Linguistics.

Armand Joulin and Tomas Mikolov. 2015. Inferring algorithmic patterns with stack-augmented recurrent nets. *Advances in Neural Information Processing Systems*, 28:190–198.

Łukasz Kaiser and Ilya Sutskever. 2015. Neural GPUs learn algorithms. *arXiv preprint arXiv:1511.08228*.

Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. 2015. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*.

Uri Katz, Mor Geva, and Jonathan Berant. 2022. Inferring implicit relations with language models. *arXiv preprint arXiv:2204.13778*.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. 2019. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

Jeonghwan Kim, Giwon Hong, Kyung-min Kim, Junmo Kang, and Sung-Hyon Myaeng. 2021. Have you seen that number? investigating extrapolation in question answering models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7031–7037.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR.

Jierui Li, Lei Wang, Jipeng Zhang, Yan Wang, Bing Tian Dai, and Dongxiang Zhang. 2019a. Modeling intra-relation in math word problems with different functional multi-head attentions. In *Proceedings of the*

57th Annual Meeting of the Association for Computational Linguistics, pages 6162–6167.

Yuanpeng Li, Liang Zhao, Jianyu Wang, and Joel Hestness. 2019b. Compositional generalization for primitive substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4293–4302, Hong Kong, China. Association for Computational Linguistics.

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

Qian Liu, Shengnan An, Jian-Guang Lou, Bei Chen, Zeqi Lin, Yan Gao, Bin Zhou, Nanning Zheng, and Dongmei Zhang. 2020a. Compositional generalization by learning analytical expressions. *Advances in Neural Information Processing Systems*, 33:11416–11427.

Qianying Liu, Wenyv Guan, Sujian Li, and Daisuke Kawahara. 2019. Tree-structured decoding for solving math word problems. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2370–2379.

Xuanqing Liu, Hsiang-Fu Yu, Inderjit Dhillon, and Cho-Jui Hsieh. 2020b. Learning to encode position for transformer with continuous dynamical model. In *International Conference on Machine Learning*, pages 6327–6335. PMLR.

Gary Marcus. 2018. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.

Gary F Marcus. 1998. Rethinking eliminative connectionism. *Cognitive psychology*, 37(3):243–282.

Sarthak Mittal, Sharath Chandra Raparthy, Irina Rish, Yoshua Bengio, and Guillaume Lajoie. 2021. Compositional attention: Disentangling search and retrieval. *CoRR*, abs/2110.09419.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2021. Investigating the limitations of transformers with simple arithmetic tasks. *arXiv preprint arXiv:2102.13019*.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2022. Show your work: Scratchpads for intermediate computation with language models. In *Deep Learning for Code Workshop*.

Maxwell Nye, Armando Solar-Lezama, Josh Tenenbaum, and Brenden M Lake. 2020. Learning compositional rules via neural program synthesis. *Advances in Neural Information Processing Systems*, 33:10832–10842.

Kuntal Kumar Pal and Chitta Baral. 2021. Investigating numeracy learning ability of a text-to-text transfer model. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3095–3101.

Ofir Press, Noah Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*.

Eric Price, Wojciech Zaremba, and Ilya Sutskever. 2016. Extensions and limitations of the neural GPU. *arXiv preprint arXiv:1611.00736*.

Linlu Qiu, Peter Shaw, Panupong Pasupat, Paweł Krzysztof Nowak, Tal Linzen, Fei Sha, and Kristina Toutanova. 2021. Improving compositional generalization with latent structure and data augmentation. *arXiv preprint arXiv:2112.07610*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Gabriel Recchia. 2021. Teaching autoregressive language models complex tasks by demonstration. *arXiv preprint arXiv:2109.02102*.

Jake Russin, Jason Jo, Randall C O'Reilly, and Yoshua Bengio. 2019. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *arXiv preprint arXiv:1904.09708*.

Imanol Schlag, Paul Smolensky, Roland Fernandez, Nebojsa Jojic, Jürgen Schmidhuber, and Jianfeng Gao. 2019. Enhancing the transformer with explicit relational encoding for math problem solving. *arXiv preprint arXiv:1910.06611*.

Jürgen Schmidhuber. 1990. Towards compositional learning in dynamic networks.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2021. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 922–938.

Hongjie Shi. 2020. A sequence-to-sequence approach for numerical slot-filling dialog systems. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 272–277.

Satinder Pal Singh. 1992. Transfer of learning by composing solutions of elemental sequential tasks. *Machine learning*, 8(3):323–339.

Avijit Thawani, Jay Pujara, Filip Ilievski, and Pedro Szekely. 2021. Representing numbers in NLP: a survey and a vision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–656, Online. Association for Computational Linguistics.

Andrew Trask, Felix Hill, Scott E. Reed, Jack Rae, Chris Dyer, and Phil Blunsom. 2018. Neural arithmetic logic units. In *Advances in Neural Information Processing Systems*, pages 8035–8044.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? Probing numeracy in embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5310–5318.

Cunxiang Wang, Boyuan Zheng, Yuchen Niu, and Yue Zhang. 2021. Exploring generalization ability of pretrained language models on arithmetic and logical reasoning. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 758–769. Springer.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Sean Welleck, Peter West, Jize Cao, and Yejin Choi. 2021. Symbolic brittleness in sequence models: on systematic generalization in symbolic mathematics. *arXiv preprint arXiv:2109.13986*.

Eric Zelikman, Yuhuai Wu, and Noah D Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*.

Xikun Zhang, Deepak Ramachandran, Ian Tenney, Yanai Elazar, and Dan Roth. 2020. Do language embeddings capture scales? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 292–299.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Chi. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Wang Zhu, Peter Shaw, Tal Linzen, and Fei Sha. 2021. Learning to generalize compositionally by transferring across semantic parsing tasks. *arXiv preprint arXiv:2111.05013*.

Yanyan Zou and Wei Lu. 2019a. Quantity tagger: A latent-variable sequence labeling approach to solving addition-subtraction word problems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5246–5251.

Yanyan Zou and Wei Lu. 2019b. Text2Math: End-to-end parsing text into math expressions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5330–5340.