BabyLM Challenge: Encouraging Tree-Structured Calculations in Transformers

Project Advisors: Shikhar Murty and Jesse Mu

Thomas Little
tjlittle@stanford.edu

Vincelot Ravoson
vravoson@stanford.edu

Abstract

In this paper we investigate the implications of training a language model on extremely limited text and explore methods of manual intervention in attempts to increase natural language understanding. Specifically, we investigate the effects of imposed tree-structured computations on transformers' ability to understand the compositionality of language. While previous studies have shown correlation between tree-structured computations and improved model performance, it remains unclear whether these results are causal or instead due to orthogonal properties in the dataset. To address this gap, we examine the performance of models with and without intervention at the transformer level and correlate them with their ability to learn hierarchical methods of composition. To measure this ability, the models are run through several methods of syntactic evaluation on existing language comprehension datasets.

1 Introduction

Over the past several years language models have shown continual improvement in performance, boasting massive training corpuses that are increasingly out of reach for individuals and smaller research teams. In the name of democratizing language model research, the BabyLM challenge[1] proposes a task to train models from scratch using limited, child-directed text. The purpose of this is to incentivize researchers with interest in cognitive modeling to focus efforts on optimizing pretraining data rather than increasing the size of the training corpus. This paper addresses the BabyLM challenge and focuses on implementing novel intervention methods in hopes of increasing model performance without the need for excessive training data.

Large scale transformer-based language models have been able to outperform many models in various language processing tasks over the last several years. However, when transformers are trained, it is natural to ask whether they process language in the same way as humans. More precisely, it is generally assumed that languages like English are *compositional*: the meaning of a complex sentence can be understood as the aggregated meaning of each sub-component. In other words, it is often assumed that humans process languages through hirearchical, tree-based segmentation. Compositionality is what allows us to understand novel sentences and generalize our understanding of language to complete previously unseen linguistic tasks. A recent paper by Murty et al.[2] introduces a metric, the t_{score}, to measure the tree-structureness of transformers, and experiments suggest a correlation between increased t_{score} and model performance.

Another issue to consider is how transformers perform when trained on small-scale datasets. In fact, typical 13-year-olds hear only about 50 million words ([3]), but they are still able to complete many tasks as well if not better as transformers. This order of magnitude is very small compared to the billions of trillions of tokens seen during the learning process of state-of-the-art language models such as GPT-4 ([4]) or Chinchilla ([5]).

In this paper, we combine these two questions to investigate whether transformers are able to learn tree-like representations of language when trained on relatively small datasets compared to current large language models. In addition, we also attempt to modify the loss function of the

transformer architecture to encourage compositionality during the learning process. By encouraging transformers to take into account the syntactic structure of sentences, we hope to improve their performance on standard NLP tasks, albeit trained on smaller datasets.

The ultimate goal of this project is to see if models trained on small datasets are able to achieve better performance, with future hopes of investigating whether their capabilities could be further enhanced when trained on larger datasets. These models could also be used to develop new language models for languages for which we do not have much data.

2 Related Work

Tree-based linguistic structures have been shown to be a common way for humans to understand language[6][7][8] and they have been correlated with increased compositional generalization in models[2]. In order to measure how tree-structured transformer computations are, Murty et al.[2] introduces the t_{score} to evaluate the tree-structuredness of a language model. The t_{score} is defined as follows:

$$t_{\text{score}} \triangleq \frac{1}{|\mathcal{D}|} \sum_{S \in \mathcal{D}} \left(\mathbb{E}_T \left[\text{SCI}(S, T) \right] - \text{SCI}\left(S, \widehat{T}_{\text{proj}}\left(S\right) \right) \right).$$

Where $\mathcal{D}$ is a collection of spans S, and T is a function that produces binary tree for any $S \in \mathcal{D}$. Here SCI is defined as span contextual invariance, which measures how dependent a given set of tokens are on their surrounding context. This function allows the model to search for bracketings of input sentences where spans have maximal contextual invariance, allowing the model to decompose the input into linguistic units with intrinsic meaning. A visual representation of a transformer comparing a bracketed embedding to the original embedding can be seen in Figure 1 below.

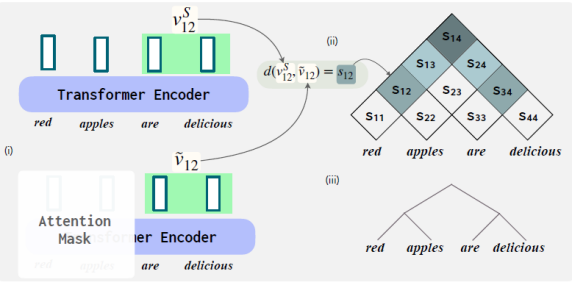


Figure 1: Tree-based hierarchical composition (Murty et al. (2022) [2])

This metric also defines $\widehat{T}_{\text{proj}}$, which is a function that approximates a transformer model as a tree (for additional information on how this projection is defined, please refer to Murty et al.[2]). The t_{score} metric computes the average SCI score of induced trees ($\widehat{T}_{\text{proj}}$), normalized against the expected SCI score under a uniform distribution over trees, which was found to be necessary to prevent the method from falsely assigning high t_{score} values to entirely context-free encoders, which would learn high SCI scores for all trees.

In a second part, the paper trains transformer architectures with 2,4 and 6 layers on three datasets commonly used for benchmarking compositional generalization. For two out of these three datasets, the t_{score} increases with the number of training iterations and suggests that transformers learn a tree-structured representation of the data, and the higher the number of layers, the better the learning of the tree-structure. Moreover, the accuracy of the transformers also increases during the training process, which suggests a correlation between increased t_{score} and model performance.

3 Approach

3.1 Baseline language model

We started by training a baseline model on our initial dataset ($\sim$ 10 million tokens), namely GPT-2 small with 12 layers of transformer blocks and 117M parameters, and a modified version of the same model with 6 layers of transformer blocks and 72M parameters. The model is trained to minimize the cross entropy loss function between the actual and the predicted token distributions, and the overall model quality is measure by the perplexity, defined as:

$$\mathcal{P}(X) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log p_\theta\left(x_i \mid x_{<i}\right)\right),$$

where p_θ is the language model probability distribution, and x_i's the actual token in the corpus.

3.2 Evaluation on grammar tasks

Then, the baseline model is evaluated using BLiMP (the Benchmark for Linguistic Minimal Pairs) framework, a grammar task used to evaluate what language models know about the most common grammatical phenomena in English. More precisely, for each sentence in one of the 67 sub-datasets of BLiMP, the model has to choose between two sequences, one of which is grammatically correct and the other one is incorrect (Figure 2). We expect our model to perform better than random chance, i.e. to get a BLiMP score greater than 0.5.

sentence_good (string)	sentence_bad (string)
"Who should Derek hug after shocking Richard?"	"Who should Derek hug Richard after shocking?"

Figure 2: Task example in the BLiMP dataset

3.3 Tree-structuredness of the baseline language model

As a next step, we take inspiration from the code of Murty et al. ([2]) to measure how tree-structured our baseline model is when it is trained on the 10M tokens dataset. To do so, we have to add the option for GPT-2 to apply tree-masks as described in the paper, and integrate this new transformer architecture to the code to compute the tree-projections of the model and draw the evolution of the t_{score} over the course of the training process.

3.4 Encouraging tree-structured computations

Then, we integrate the code used to compute tree-projections to the source code of GPT-2 to encourage the model to learn tree-stuctured representations of its inputs. More precisely, we augment the cross-entropy loss $\mathcal{L}_{\text{entropy}}$ used in the baseline model by adding an additional regularized loss term:

$$\mathcal{L}_{\text{entropy}} + \lambda\mathcal{L}_{\text{tree}}$$

to encourage the transformer to learn tree-structured representations of the inputs.

To calculate the additional $\mathcal{L}_{\text{tree}}$ loss term, we first determine a SCI score for all possible splits of the input sentence. We then recursively search each split to identify the optimal decomposition for the input sentence that maximizes the SCI score. Finally, we compare this optimal decomposition with the SCI score of a randomly selected split using the following formula:

$$\mathcal{L}_{\text{tree}} = \max\left(\left(\text{SCI}_{\text{random}} + \beta - \text{SCI}_{\text{best}}\right), 0\right).$$

This aims to train the model to prefer the optimal splits over randomly selected splits (here β is a hyperparameter currently set to 0.1).

4 Experiments

4.1 Data

In all our experiments, we trained our models on the 10M token datasets provided by the BabyLM Challenge organizers. This data is an aggregation of text corpora from several websites, books and dialogue transcripts (Table 1).

Dataset	Domain	# Words		Proportion
		Strict-SMALL	STRICT	
CHILDES (MacWhinney, 2000)	Child-directed speech	0.44M	4.21M	5%
British National Corpus (BNC), [1] dialogue portion	Dialogue	0.86M	8.16M	8%
Children's Book Test (Hill et al., 2016)	Children's books	0.57M	5.55M	6%
Children's Stories Text Corpus [2]	Children's books	0.34M	3.22M	3%
Standardized Project Gutenberg Corpus (Gerlach and Font-Clos, 2018)	Written English	0.99M	9.46M	10%
OpenSubtitles (Lison and Tiedemann, 2016)	Movie subtitles	3.09M	31.28M	31%
QCRI Educational Domain Corpus (QED; Abdelali et al., 2014)	Educational video subtitles	1.04M	10.24M	11%
Wikipedia [3]	Wikipedia (English)	0.99M	10.08M	10%
Simple Wikipedia	Wikipedia (Simple English)	1.52M	14.66M	15%
Switchboard Dialog Act Corpus (Stolcke et al., 2000)	Dialogue	0.12 M	1.18	1%
Total	-	9.96M	98.04M	100%

Table 1: BabyLM Challenge training dataset.

5 Results & Analysis

5.1 Baseline language model

For our baseline models we used datasets described in 4.1. to start pretraining causal language models and get initial results. Specifically, our models were trained on a 10 million word dataset containing sources with various child-directed text[1]. This dataset was already split into training, test, and development sets by the BabyLM Challenge team.

Perplexity was used as one of the main metrics to assess the performance of the model. Our goal was to train a model with a low perplexity, i.e. a model for which the probability to predict the correct next word given the previous words is high. Two language models were tested: GPT-2 small with 12 layers of transformer blocks and 117M parameters, and a modified version of the same model with 6 layers of transformer blocks and 72M parameters. The hyperparameters of each model (aside from the number of transformer layers) are the same as the vanilla GPT-2 model as described in the original GPT-2 inception paper [9]. The rationale for these relatively small models was to get an idea of how "standard" models behave on our small dataset while still maintaining the ability to iterate through model changes quickly.

In Figure 3, we observe that both models perform similarly, with the larger model providing marginally better results.

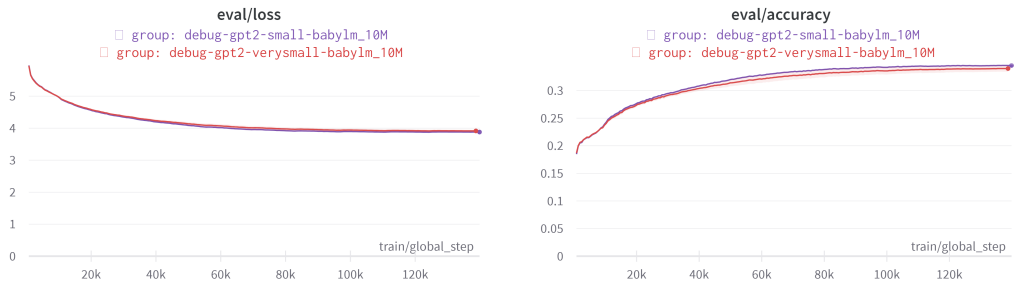


Figure 3: Evaluation loss and model accuracy during the training process.

5.2 Evaluation on grammar tasks

In a second part, we ran an ensemble of syntactic evaluations using BLiMP, the score of which can be seen in the Table 2 and Figure 4 below. The baseline model obtains an average BLiMP score of 0.69, which is above the random chance score of 50%. It means that the model has been trained on

enough training samples to recognize several grammatical phenomena in English. However, it does not perform as well as the pretrained GPT-2 model, which is trained on way more examples.

Model	BLiMP Overall Score
GPT2-BabyLM-Baseline	0.69
GPT2-Pretrained	0.83
Human	0.89

Table 2: BLiMP Scores

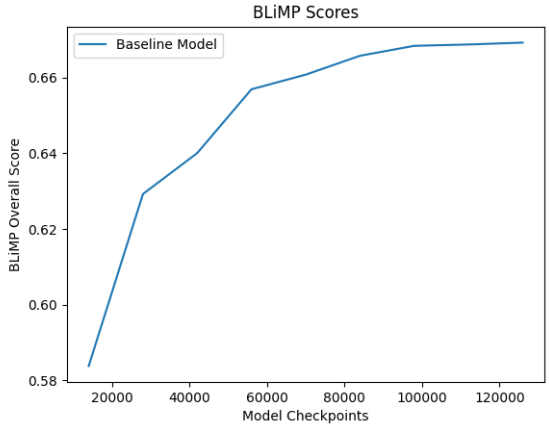


Figure 4: BLiMP Evaluation

5.3 Tree-structuredness of the baseline language model

When the baseline model is trained on the 10M word dataset described in 4.1, the training process shows an increase in tree-structuredness as training progresses and model performance increases (Figure 5). This mirrors the correlation described by Murty et al. (2022) [2] but still does not show causation.

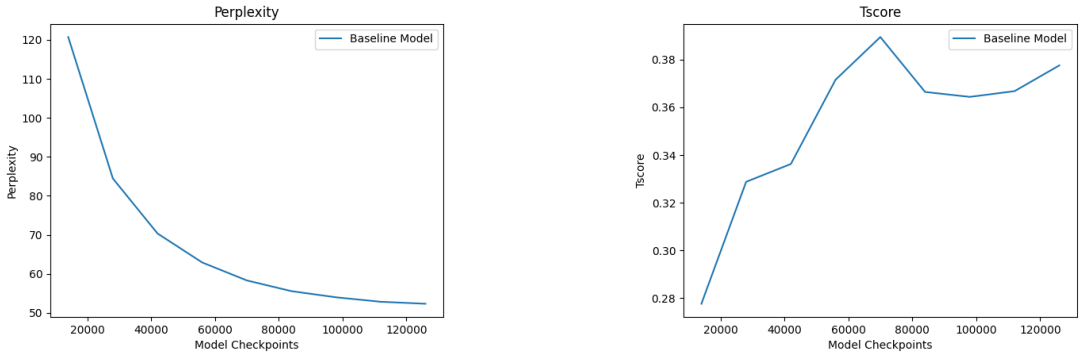


Figure 5: Baseline Model Results [Left: Perplexity | Right: t_{score} evolution]

5.4 Encouraging tree-structured computations

In order to encourage the model to learn more tree-structured representations of tokens, we augmented the cross-entropy loss of the baseline model by adding a regularized loss term: $\lambda\mathcal{L}_{\text{tree}}$. To calculate the additional loss term, we first determine a SCI score for all possible splits of the input sentence. We then recursively search each split to identify the optimal decomposition for the input sentence that maximizes the SCI score. Finally, we compare this optimal decomposition with the SCI score of a randomly selected split using the following formula:

$$\mathcal{L}_{\text{tree}} \triangleq \max((\text{SCI}_{random} + \beta - \text{SCI}_{best}), 0)$$

This aims to train the model to prefer the optimal splits over randomly selected splits (here β is a hyperparameter currently set to 0.1).

The updated GPT-2 architecture with a tree-regularizing loss function was retrained on the same data as the baseline model. Its evaluation accuracy is plotted in red below against the 6 layer encoder baseline model. Perplexity, BLiMP score, and t_{score} plots are omitted in this section as they show stochastic behavior as seen in Figure 6 below and do not provide further insight into the behavior of the model.

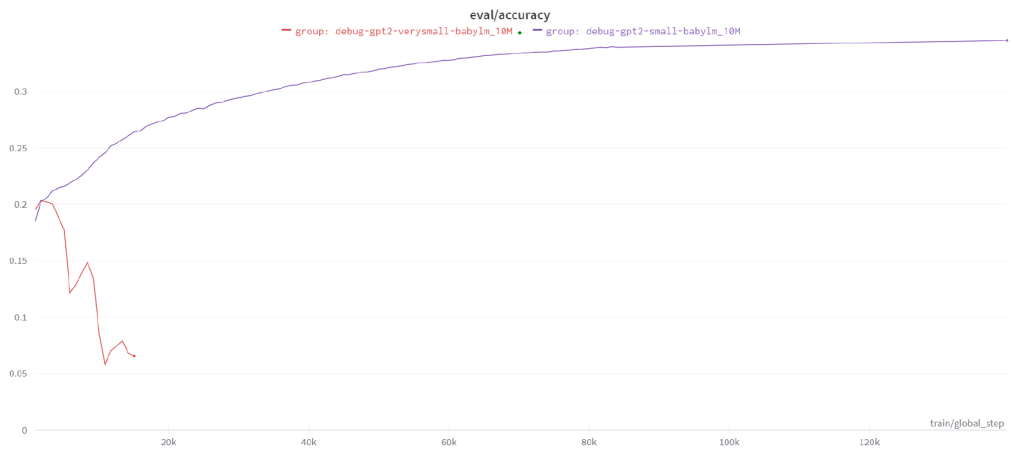
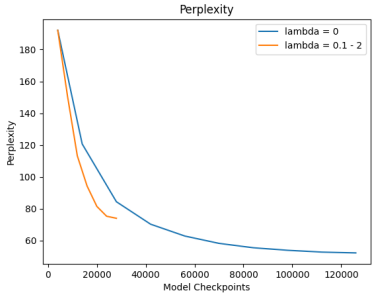


Figure 6: Evaluation accuracy (red: regularized model with $\lambda = 10$, purple: basline model).

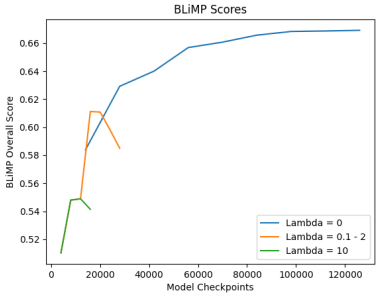
We observe that with a penalty coefficient $\lambda = 10$, adding the regularization term does not result in a successful improvement of the accuracy. We tried to run the model with several values of λ, ranging from 0.1 to 10 and observed the accuracy after a thousand epochs. When the penalty λ is low, i.e. around $\sim$0.1, the regularized loss takes values around $\sim$0.01, which is negligible compared to the typical values of $\sim$5 taken by the cross-entropy loss, and the accuracy curve closely follows the purple line showed in Figure 4. However, when the value of λ increases to 10 so that the two regularization terms have the same order of magnitude, we observe that the accuracy of the regularized model suddenly drops after a few number of training iterations. In that case, it means that the model only learns how to perform tree-like computations, but does not learn how to predict the next word in a sentence accurately, resulting in a high cross-entropy loss. Intermediate values of λ lead to curves that start to increase with the same trend than the purple line, before decreasing like in Figure 6.

We can further analyze the performance of this regularization effect by studying the plots below. Note that baseline model performance is shown in blue, while a t_{score} regularization of 10 and 0.1 are shown in green and orange, respectively. Additionally, please note that the model with a regularization factor of 10 is omitted from the perplexity chart as its perplexity increased at a rate that would make the other two models unreadable, however its poor performance is captured by the other two plots. Additionally, due to limitations in the algorithmic speed of the applied tree-regularization, the regularized experiments were only run until performance appeared to degrade. This is a limitation

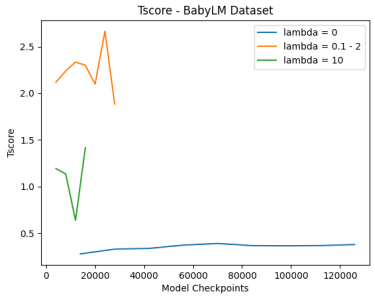
in this study, as these local drops are not necessarily indicative of the final model performance, however, this was done to be able to run experiments with different regularization factors, rather than running a single experiment for a longer time.



(a) Perplexity



(b) BLiMP Score



(c) T-score

Figure 7: Regularized Model Results

Initially, we can see that indeed, larger regularization values do increase t_{score} by a significant margin, indicating that the model is learning a tree-based method of composition. However, by looking at Figure 7.b, we can see that the highly regularized model has a maximum BLiMP score of 0.54, which is barely above the random chance baseline of 0.5. The model with a regularization of 0.1 seems to perform well until reaching a model checkpoint of 16000, but neither model performs as well as the baseline. One note to make on the model described in orange is that represents a combination of regularization factors. We found that large regularization factors decreased overall performance on syntactic evaluations, but resulted in higher t_{scores}. One reason for this could be that the model is too focused on optimizing the t_{scores} without maintaining a solid understanding of language. To mitigate this, we looked at a scheduled regularization factor, with λ set to 0.1 until model checkpoint 16000, when it was increased to λ of 2. Further experiments would address more refinement of this scheduling, with the general idea being that the model can first learn a fundamental understanding of language before attempting to shape it into more tree-structured compositions.

It is important to note that there are other factors that may also be playing a role in these results. One potential cause of these issues likely arises from the limiting factors imposed on our implementations to conduct tractable experiments given our time frame. In fact, the complexity of the recursive chart used to calculate $\mathcal{L}_{\text{tree}}$ is $O(n^3)$, where n is related to the length of the input sequence as well as the number of input sentences analyzed per batch. To allow our GPU instance to run in a reasonable amount of time, we truncated all input sentences to a maximum of 10 words and set our maximum sentence batch size to 50.

Both these assumptions likely have large impacts on model performance, as they increased the inherent stochaticity in the gradient updates, since each update is based on a very small percentage of the overall dataset. Accounting for the fact that our training dataset contained 650,000 input sentences, a batch size of 50 is only 0.008% of the overall dataset. We attempted using 1% of the

overall dataset to calculate $\mathcal{L}_{\text{tree}}$ at each step, however, training was estimated to take over 600 hours (again due to the $O(n^3)$ time).

6 Conclusion

Our study shows that transformer models trained on small datasets inspired by the limitations of human development increasingly perform tree-like computations, which mirrors the results of Murty et al. (2022) [2]. However, this result only implies a correlation between high tree-stucturedness and high performance, but not causation. In order to prove the causation between these two observations, we regularized the loss of the transformer architecture to encourage the model to learn tree-structured representations of the input data. Our experiments suggest that adding the regualarized tree-loss does not result in an improvement of the model accuracy, due to the inability of the model to optimize both cross-entropy and tree-loss during the training process.

However, these results need to be nuanced due to limiting constraints in our implementations to reduce the algorithmic runtime. In fact, as it is currently, using our method (even for small-scale datasets) presents challenges for individuals wishing to train models. It is possible to reduce the runtime to at least $O(n^2)$, which would increase the feasibility of using this method, and would likely result in improvements over the performance seen in this study. Hopefully this avenue is explored in the future so that further research can be conducted on the impact of imposed tree-structured computations in transformer models and their abilities to gain increased compositional generalization.

References

[1] Alex Warstadt, Leshem Choshen, Aaron Mueller, Adina Williams, Ethan Wilcox, and Chengxu Zhuang. Call for papers – the babylm challenge: Sample-efficient pretraining on a developmentally plausible corpus, 2023.

[2] Shikhar Murty, Pratyusha Sharma, Jacob Andreas, and Christopher D. Manning. Characterizing intrinsic compositionality in transformers with tree projections, 2022.

[3] Betty Hart and Todd R Risley. The early catastrophe: The 30 million word gap by age 3. *American educator*, 27(1):4–9, 2003.

[4] OpenAI. Gpt-4 technical report, 2023.

[5] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*.

[6] Stephen Crain and Mineharu Nakayama. Structure dependence in grammar formation. *Language*, 63(3):522–543, 1987.

[7] John Hale, Chris Dyer, Adhiguna Kuncoro, and Jonathan Brennan. Finding syntax in human encephalography with beam search. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2727–2736, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[8] Christophe Pallier, Anne-Dominique Devauchelle, and Stanislas Dehaene. Cortical representation of the constituent structure of sentences. *Proceedings of the National Academy of Sciences*, 108(6):2522–2527, 2011.

[9] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.