Curriculum Learning with TinyStories

Stanford CS224N Custom Project

Mentor: Shikhar Murty

Michail Christiaan Melonas
Department of Computer Science
Stanford University
michail@stanford.edu

Abstract

Language model performance improves predictably as model size, training data, and compute resources are increased (Kaplan et al., 2020). Performance has also been shown to be affected by the quality (Gunasekar et al., 2023), breadth and diversity (Eldan and Li, 2023) of the training corpora. In this work, we explore whether the order in which training examples are utilised affect performance. Our focus is on the emergence of language abilities such as grammar, creativity, and consistency. We implement a curriculum learning strategy that ranks inputs using a perplexity based measure. Our learning schedule proceeds according to performance on a separate validation dataset. Experimental results shows that under some circumstances such a strategy is beneficial. However, we also find evidence that our strategy can be harmful compared to conventional random data shuffling.

1 Introduction

Language model (LM) performance improve as we increase model size, training data, and compute resources. Kaplan et al. (2020) found that cross-entropy loss scales as a power-law with these factors. Performance has also been shown to be affected by the characteristics of the training corpora. Gunasekar et al. (2023) showed that the usage of textbook quality training data can result in models that outperform counterparts which have orders of magnitude more parameters and that have been trained on much larger datasets. Similarly, Eldan and Li (2023) found that the breadth and diversity of the corpora affect the scale at which language abilities start to emerge. In particular, it was found that models with as few as 10 million parameters can produce coherent English text with near perfect grammar when trained on an appropriately narrow dataset. This is in contrast to models such as GPT-2 (117M) (Radford et al., 2019) which have been trained on internet datasets. Said model has in excess of 100 million parameters and has been trained on more than 8 million documents, yet struggles to produce coherent text beyond a few words. Understanding the factors which govern model performance is crucial for the development of improvements.

LMs have achieved impressive results on various natural language processing (NLP) tasks. However, most notable models are very large. The enormous computational cost of modern LMs prohibit their adoption and therefore limits their utility. The efforts of Gunasekar et al. (2023) and Eldan and Li (2023) are interesting since they indicate that LMs can be useful at a much smaller scale than previously thought. In this paper, we investigate another such avenue. We explore whether the order in which training examples are utilised affect performance. That is, we apply the technique of curriculum learning to language modelling. To this end, we make use of the TinyStories dataset (Eldan and Li, 2023). Our focus is on the emergence of language abilities such as grammar, creativity, and consistency. We compare our results to that of a baseline LM trained according to conventional random data shuffling. Overall, we find mixed evidence regarding the benefit of CL.

2 Related Work

2.1 Curriculum Learning

CL involves training machine learning models in a meaningful order. It requires sorting training examples from "easy" to "hard", and presenting them to a given model in a way that is more "educational" than conventional random data shuffling. Arguably, this approach to learning is closer to that of humans which is highly structured and organised. CL was first introduced by Bengio et al. (2009) who presented the technique as a continuation method for non-convex optimisation. Said authors applied CL to both a computer vision problem (involving the classification of geometric shapes) and to language modelling. An increase in the speed of convergence of the training process as well as better generalisation performance was found. Another notable example is due to Zaremba and Sutskever (2015) who employ CL when training an LSTM model tasked with evaluating short Python programs. The authors are able to improve accuracy by around 10% using the technique.

2.2 TinyStories

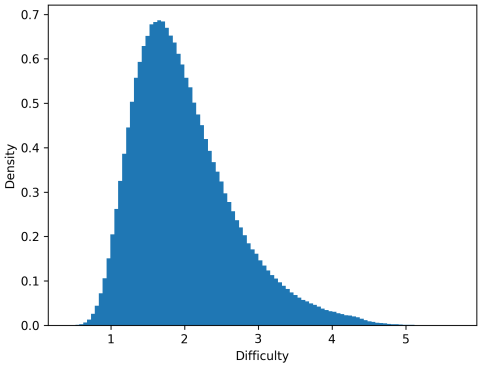
Internet datasets like WebText (Radford et al., 2019) cover a very large amount of information from numerous domains. When training models on such datasets, we are not only learning about language, but also about the various topics and information contained in the data. As such, Eldan and Li (2023) observe that the necessary scale for the emergence of abilities that LMs have become known for, such as the generation of coherent text, reasoning, and instruction following, is large. In response said authors developed the TinyStories dataset. TinyStories is a high quality synthetic dataset generated via GPT-4. TinyStories is purposefully simple and narrow, while still reflecting the basics of natural language: grammar, vocabulary, facts, and reasoning. The dataset consists of more than 2.7 million short stories (see Figure 1a) at the level of comprehension of a typical 3 or 4 year-old child. The idea of the dataset is to allow studying the minimal requirements of a language model to generate coherent text. Its size and simplicity means that small models can fit it well.

One day, Tom and Sue were playing outside. They loved to run and play in the sun. They saw big, dark clouds in the sky. They knew rain was coming. Sue said, "Let's run home before the rain comes!" Tom said, "No, let's keep playing!"

The rain came down fast and fierce. Tom and Sue got very wet. They were not happy. They tried to run home, but the rain was too strong. They saw a big tree and ran under it to stay dry. They waited for the rain to stop.

While they were under the tree, they heard a loud noise. It was a big, scary bear! The bear was also hiding from the rain. Tom and Sue were very scared. They did not know what to do. The bear saw them and roared, "Go away! This is my tree!" Tom and Sue ran away in the rain, but they could not find their way home.

(a) Example entry from the TinyStories dataset



(b) Distribution of story difficulty

Figure 1: TinyStories dataset example and story difficulty distribution

2.3 GPT-Eval and BLiMP

Evaluating the quality of open-ended language generation is challenging. This follows since there are typically multiple correct completions associated with a given prompt. A well-known (but perhaps less extreme) example of this problem in NLP is with the evaluation of machine translation (MT) systems. For a given input sequence, there are often many equally valid alternative translations. BLEU, the most common evaluation metric for MT, is flawed since it is dependent on such alternatives being available as reference translations. Consequently, a good translation can get a poor score. GPT-Eval (Eldan and Li, 2023) is a general framework that can be used evaluate any arbitrary language generation. Said framework leverages an existing large LM, such as GPT-4 (OpenAI et al.,

2024), to evaluate the quality of completions generated by a given LM. We uilise GPT-Eval to grade the grammar, creativity, and content self-consistency of completions due to our LMs.

The Benchmark of Linguistic Minimal Pairs for English (BLiMP) (Warstadt et al., 2023) is a framework used to evaluate the grammatic ability of LMs. BLiMP consists of a suite of 67000 pairs of minimally different sentences. Each pair consists of a "correct" sentence and a slightly perturbed "incorrect" sentence, and represents a particular aspect of English grammar. The BLiMP score assigned to a given LM is the proportion of pairs for which the model assigns a higher probability to the "correct" sentence than the "incorrect" counterpart.

3 Approach

A particular CL approach is characterised by (a) the choice of difficulty measure used to rank input examples and (b) the schedule according to which examples of various levels are introduced during training. In our approach, the difficulty of training examples is measured via perplexity. That is, supposing that a given text example consists of $t+1$ tokens, $x = \{x_0, x_1, \ldots, x_t\}$, the perplexity according to some LM, say p_θ, is given as:

$$\text{PPL}(x|p_\theta) = e^{-\frac{1}{t}\sum_{j=1}^{t} \ln p_\theta(x_j|x_0,x_1,\ldots,x_{j-1})}. \tag{1}$$

Here, $p_\theta(x_j|x_0, x_1, \ldots, x_{j-1})$ indicates the probability of x_j conditional on all previous tokens. Perplexity is a measure of a LM's ability to predict the next token in a sequence of text. Clearly, lower values indicate improved prediction performance, and the smallest possible perplexity is 1. Noting that we have access to existing LMs trained on TinyStories, we assign a difficulty score to each input as the perplexity delta between a large model, say p_γ, and a small model, say p_α:

$$\Delta(x) = \text{PPL}(x|p_\alpha) - \text{PPL}(x|p_\gamma). \tag{2}$$

The intuition for measuring difficulty in this way is that for "easy" input examples, the perplexity of p_γ and p_α should be approximately equal, whereas for "difficult" examples, the perplexity of p_γ should be lower than that of p_α. This is informed by the assumption that larger models better fit the dataset. Using 2 as the choice of difficulty measure was suggested by an author of TinyStories. The models we use for this calculation are available on Huggingface as `roneneldan/TinyStories-3M` (p_α) and `roneneldan/TinyStories-33M` (p_γ). After a manual inspection of the scores due to 2, we noticed that stories with large absolute perplexity were sometimes left with a much smaller score (due to little difference between $\text{PPL}(x|p_\alpha)$ and $\text{PPL}(x|p_\gamma)$) than stories with much smaller absolute perplexities. Similarly, stories with relatively small absolute perplexity sometimes had very large scores. Given that perplexity itself can be seen as a measure of story "difficulty", we adjusted 2 to reflect both the perplexity delta as well as the absolute perplexity (we chose the smaller model, p_α, for this). Supposing we have n stories in our dataset, the final difficulty score assigned to a given story is the normalised sum of 2 and 1:

$$\text{score}(x_i) = \frac{\Delta(x_i)}{\frac{1}{n}\sum_{z=1}^{n}\Delta(x_z)} + \frac{\text{PPL}(x_i|p_\alpha)}{\frac{1}{n}\sum_{z=1}^{n}\text{PPL}(x_z|p_\alpha)} \tag{3}$$

for $i = 1, 2, \ldots, n$.

Our approach to CL is inspired by Zaremba and Sutskever (2015). After ranking training examples according to 3, we start with the easiest 10%. We initiate training with this subset, and periodically monitor performance on a validation set. Once the validation loss is no longer decreasing (the criteria for which is three consecutive values being larger than the smallest loss up to that point), we add the next 10% of examples to the training data. This is continued until the training set matches the full dataset. With this approach, the training distribution gradually starts to approximate that of the baseline model, which is trained according to random data shuffling using the full dataset.

4 Experiments

4.1 Data

We utilise the TinyStories dataset described in section 2.2. Upon a quality inspection, we found examples of stories that contain non-English characters, and stories that consisted of long sequences

of repetitive text. We also found stories composed of only a single word (e.g. *Once*), and stories that were otherwise inappropriate (e.g. *They put water and salt and sugar and flour and carrots and apples and cheese and cookies and milk and eggs and bananas and pasta and bread and jam and butter ...*). It was found that perplexity correlates well with such cases. Around $300k$ stories were removed by dropping those for which the perplexity due to p_α was larger than 10 or for which perplexity due to p_γ was larger than 5. The distribution of difficulty for the remaining 2.4 million stories using 3 is given in Figure 1b. We found a weak positive correlation ($\rho = 0.2742$) between the number of words in given story and 3. For the purpose of CL, we constructed a held out set consisting of $10k$ stories. These inputs were taken from the cleaned dataset, and contained stories of all difficulties.

4.2 Evaluation method

Our goal is to investigate the effect of CL on the emergence of the ability to generate coherent English text. To this end, the main evaluation method is that of GPT-Eval described in section 2.3. We evaluate the completions of our models on two sets of prompts. The first consists of the single short prompt, "Once upon a time". The second is a hand-selected subset of evaluation prompts from the TinyStories paper, composing of the following much longer prompts:

- "Once upon a time there was a curious boy who lived in a house with a big garden. Every day he explored the garden he found new surprises. But one day, it was raining so hard that his mother told him"
- "One day a girl walked into the living room and noticed something very strange. There was a huge cabinet standing in the corner. It looked very old and heavy. She walked over and tried to open it, when suddenly"
- "Alice wanted to play with her doll, but she couldn't remember where she had put it. She looked all around the house but couldn't find it, so she decided".

After generating a completion from a given model, GPT-Eval grades the grammar, creativity, and self-consistency with a score between 1 and 10. We compare the evolution of these values between our CL effort and a corresponding baseline. We also evaluate models using the BLiMP score and look at how the average validation loss compares on a disparate set of approximately $20k$ stories.

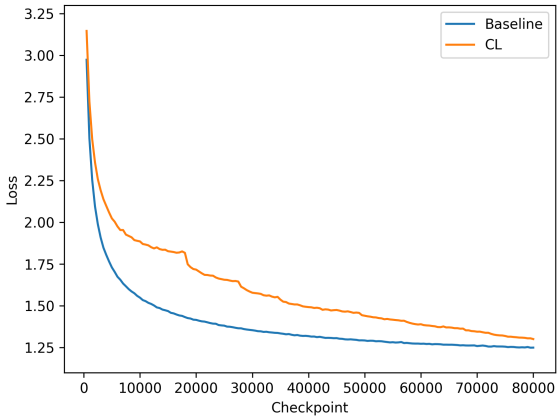


Figure 2: Average cross-entropy loss on TinyStories validation set.

4.3 Experimental details

We train a custom byte-pair encoding tokenizer on the TinyStories dataset with a vocabulary size of $10k$. This is done with the help of the `transformers` library. We make use of the GPT-2 (Radford et al., 2019) architecture, with a context window and embedding dimension of 512, 8 transformer blocks, and 8 attention heads. This gives a model with 30.5 million total parameters. Most of the

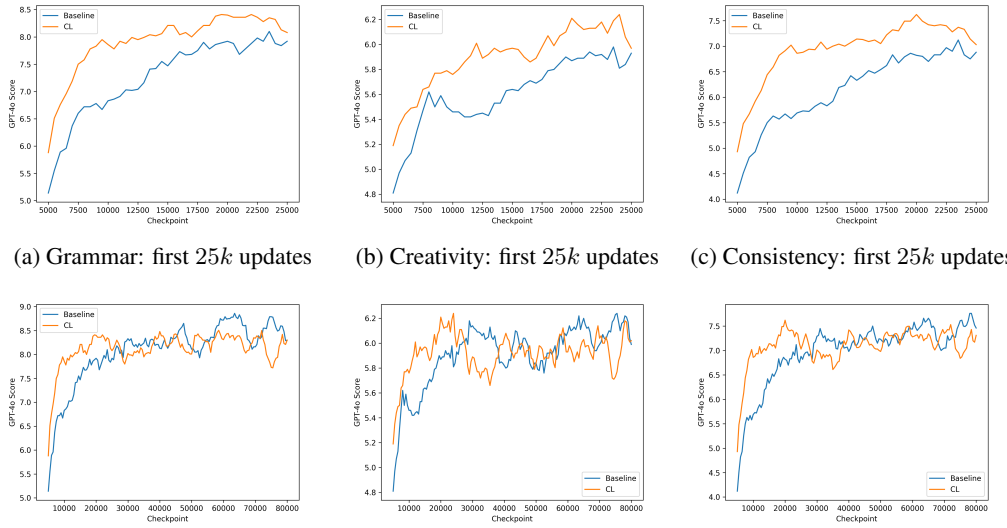
GPT-2 implementation is borrowed from `nanoGPT`. We use cross-entropy as choice of loss function, and train with a constant learning rate of 0.0001. We rely on the AdamW optimizer (Loshchilov and Hutter, 2019) with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-6}$, and weight decay of 0.1, and use gradient norm clipping of 1. A batch size of 64 is used. All training is done using a single NVIDIA A100 GPU.

4.4 Results

We train two models: one with conventional random data shuffling (`Baseline`), and one according to our CL methodology (`CL`). Both models are trained for $80k$ steps. After every 500 updates we checkpoint weights. Figure 2 shows the evolution of the validation loss over the training updates. Looking at `CL`, we notice clear step changes: these are the points at which the curriculum was expanded. The validation set contains examples of all levels of difficulty. The two models appear to converge to the same value, with the `Baseline` reaching convergence much sooner than `CL`. This makes sense given that `Baseline` was trained on a dataset with the same distribution as the validation set.

Figures 3 and 4 contain the results due to GPT-Eval. For each of the prompts described in section 4.2, we generate 10 completions at each checkpoint. Sampling is done with temperature 0.75. For each set of prompts the scores are averaged at a given checkpoint. The provided figures are the rolling averages (of said averages) using a window of size 10. The results due to the simple prompt "Once upon a time" indicate that language ability emerges sooner when using CL. Both `Baseline` and `CL` converge to scores of approximately $8.3/10$, $6.0/10$, and $7.35/10$ for each of grammar, creativity, and content self-consistency, respectively. `CL` requires roughly $15k$ fewer updates ($25k$ versus $10k$ steps) than `Baseline` to converge. Looking at Figure 4, however, we appear to find the opposite result. Here, `CL` outperforms `Baseline` over the first $10k$ updates. After this point, the scores for `CL` are much lower. `Baseline` clearly outperforms the `CL` counterpart, reaching final scores of $6.39/10$, $5.66/10$, and $4.87/10$ versus $5.94/10$, $5.75/10$, and $4.5/10$ for each of the three categories. Thus, for the longer prompts, `CL` is only slightly better in one category (1.59%) and much worse in the two other (7.58% and 8.22%). Additionally, in Figure 4, it is clear that `Baseline` acquires a given level of grammar, creativity, and consistency much sooner than `CL` (after the initial $10k$ steps).

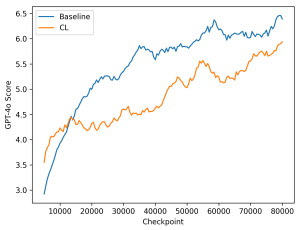
Finally, we obtain BLiMP scores on the last checkpoints of 0.5646 and 0.5721 for `CL` and `Baseline`, respectively. We note that these scores are only slightly better than random guessing (GPT-2 achieves 0.83 (Warstadt et al., 2023)). Additionally, the score due to `Baseline` is higher than that of `CL`.



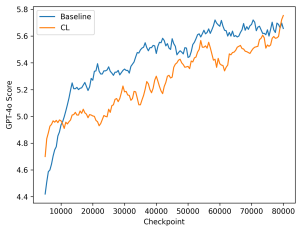
(a) Grammar: first $25k$ updates (b) Creativity: first $25k$ updates (c) Consistency: first $25k$ updates

(d) Grammar: all $80k$ updates (e) Creativity: all $80k$ updates (f) Consistency: all $80k$ updates

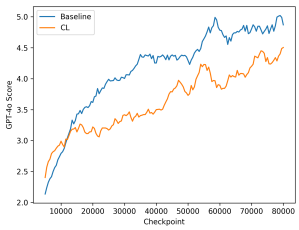
Figure 3: GPT-Eval scores when using the prompt, "Once upon a time"



(a) Grammar: all $80k$ updates



(b) Creativity: all $80k$ updates



(c) Consistency: all $80k$ updates

Figure 4: GPT-Eval scores for three hand-selected prompts

5 Analysis

The evidence presented in section 4.4 suggests that the order in which training examples are utilised does affect LM performance. However, the results are clearly mixed. Some of the evidence indicate that CL has a strong positive impact on the emergence of language abilities (Figure 3). Yet, other evidence (Figure 4 and the BLiMP scores) indicate the opposite. We observe that the overall scores for grammar, creativity, and consistency are lower for the longer prompts compared to the short prompt, "Once upon a time". It is also interesting to note that the scores due to the longer prompts continue to improve until the final training step. Those due to the short prompt converge to their final values early on in the training process. Generating completions for the longer prompts is thus a more challenging task, perhaps due to the long-term dependencies present in the sequences. We suspect the difference in performance on the two sets of prompts can be explained, in part, by the positive correlation between the number of words in a story and its difficulty score. However, as noted before, said correlation is not very strong, and can only explain part of the result.

Looking at the evolution of scores due to the set of longer prompts, our LM may well have benefited from a CL strategy close to what Bengio et al. (2009) used when training a model for shape classification. Noticing that `CL` is superior over the first $10k$ steps, a schedule whereby we start training with a set of easy stories, and then abruptly shift to the full dataset, could have delivered results consistent with those due to "Once upon a time". This is in contrast to our approach of gradually increasing the difficulty of stories. Additionally, we might also have benefited from using a different measure of difficulty than 3. Possibilities include using the length of stories or the density of infrequent words as measures.

6 Conclusion

Our approach to CL with TinyStories was a partial success. We demonstrated that under some circumstances the ability to produce coherent English emerges after significantly fewer parameter updates, and requires fewer unique training examples, when utilising inputs in a meaningful order. However, we also found evidence that CL can be harmful. In our case, this occurred when evaluating completions due to longer prompts. Despite mixed results, we maintain that CL is a promising avenue for reducing the necessary scale for LMs to be useful. In future work, we would like to study CL strategies that utilise alternative difficulty measures and learning schedules.

7 Ethics Statement

An obvious ethical dilemma with any machine learning research is selective reporting of results in favour of some claim. For example, we could have given a skew impression of the success of our CL methodology by only reporting on the results related to the single short prompt, and withholding those related to the set of longer prompts. Although the strength of such evidence can be criticised as weak, we can still create a false impression when only reporting on the successes (and not the failures) of our experiments. A mitigation strategy to this problem is a commitment to full transparency of all results.

Another possible ethical consideration relates to the environmental impact of machine learning. In this context, both the electricity and hardware required for training are relevant. In our case, each model required around 5 hours of training and 2 hours of evaluation on a large GPU. We also conducted multiple smaller scale experiments the in a build-up to our final training. Arguably, however, research such as ours, which is presented as an effort into reducing the necessary scale for LMs to be useful, is a step in the right direction.

References

Y. Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. volume 60, page 6.

Ronen Eldan and Yuanzhi Li. 2023. Tinystories: How small can language models be and still speak coherent english?

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. 2023. Textbooks are all you need.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted

Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2023. Blimp: The benchmark of linguistic minimal pairs for english.

Wojciech Zaremba and Ilya Sutskever. 2015. Learning to execute.