Scaling Data-Constrained Language Models

Niklas Muennighoff [1] **Alexander M. Rush** [1] **Boaz Barak** [2] **Teven Le Scao** [1]
Aleksandra Piktus [1] **Nouamane Tazi** [1] **Sampo Pyysalo** [3] **Thomas Wolf** [1] **Colin Raffel** [1]
[1] Hugging Face [2] Harvard University [3] University of Turku
n.muennighoff@gmail.com

Abstract

The current trend of scaling language models involves increasing both parameter count and training dataset size. Extrapolating this trend suggests that training dataset size may soon be limited by the amount of text data available on the internet. Motivated by this limit, we investigate scaling language models in data-constrained regimes. Specifically, we run a large set of experiments varying the extent of data repetition and compute budget, ranging up to 900 billion training tokens and 9 billion parameter models. We find that with constrained data for a fixed compute budget, training with up to 4 epochs of repeated data yields negligible changes to loss compared to having unique data. However, with more repetition, the value of adding compute eventually decays to zero. We propose and empirically validate a scaling law for compute optimality that accounts for the decreasing value of repeated tokens and excess parameters. Finally, we experiment with approaches mitigating data scarcity, including augmenting the training dataset with code data or removing commonly used filters. Models and datasets from our 400 training runs are freely available at https://github.com/huggingface/datablations.

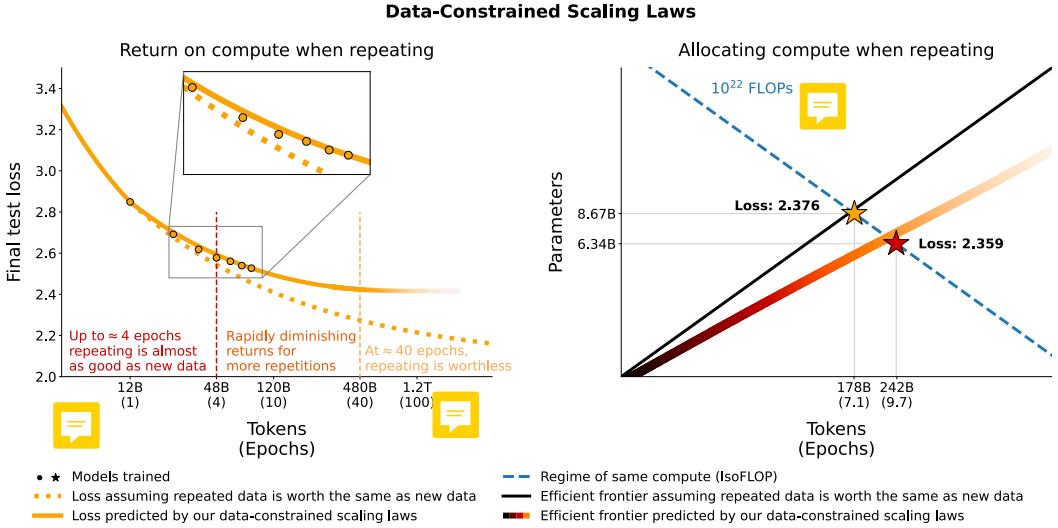


Figure 1: ***Return* and *Allocation* when repeating data.** *(Left):* Loss of LLMs (4.2B parameters) scaled on repeated data decays predictably (§6). *(Right):* To maximize performance when repeating, our data-constrained scaling laws and empirical data suggest training smaller models for more epochs in contrast to what assuming Chinchilla scaling laws [42] hold for repeated data would predict (§5).

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

1 Introduction

Recent work on compute-optimal language models [42] shows that many previously trained large language models (LLMs, which we define as having more than one billion parameters) could have attained better performance for a given compute budget by training a smaller model on more data. Notably, the 70-billion parameter Chinchilla model [42] outperforms the 280-billion parameter Gopher model [89] while using a similar compute budget by being trained on four times more data. Extrapolating these laws for compute allocation (hereafter "Chinchilla scaling laws") to a 530 billion parameter model, such as the under-trained MT-NLG model [99], would require training on a massive 11 trillion tokens, corresponding to more than 30 terabytes of text data. For most languages, available data is several orders of magnitude smaller, meaning that LLMs in those languages are already data-constrained. Villalobos et al. [112] estimate that even high-quality English language data will be exhausted by the year 2024 given the Chinchilla scaling laws and the trend of training ever-larger models. This motivates the question [112, 81]: what should we do when we run out of data?

In this work we investigate scaling large language models in a data-constrained regime, and whether training an LLM with multiple epochs of repeated data impacts scaling. Using multiple epochs is, of course, standard in machine learning generally; however, most prior large language models have been trained for a single epoch [51, 15] and some work explicitly advocates against reusing data [40]. An exception is the recent Galactica models [108] that were trained for 4.25 epochs and exhibit continually decreasing validation loss and improving downstream performance throughout training. However, the experiments of Galactica do not compare this setup to an alternative non-data-constrained model trained for one epoch on unique data. Without this comparison, it is difficult to quantify the trade-off between additional compute versus additional data collection.

Our main focus is to quantify the impact of multiple epochs in LLM training such that practitioners can decide how to allocate compute when scaling models. Toward this end, we assembled a battery of empirical training runs of varying data and compute constraints. Specifically, we train more than 400 models ranging from 10 million to 9 billion parameters for up to 1500 epochs and record final test loss. We use these results to fit a new *data-constrained scaling law* that generalizes the Chinchilla scaling law [42] to the repeated data regime and yields a better prediction of loss in this setting. Figure 1 summarizes our main results targeting the value of repeated data (*Return*) and optimal allocation of resources in that regime (*Allocation*). We find that, while models trained for a single epoch consistently have the best validation loss per compute, differences tend to be *insignificant* among models trained for up to 4 epochs and do not lead to differences in downstream task performance. Additional epochs continue to be beneficial, but returns eventually diminish to zero. We find that, in the data-constrained regime, allocating new compute to both more parameters and epochs is necessary, and that epochs should be scaled slightly faster. These findings suggest a simple way to continue scaling total training compute budgets further ahead in the future than the previously anticipated limits.

Finally, given the challenges imposed by data constraints, we consider methods complementary to repeating for improving downstream accuracy without adding new natural language data. Experiments consider incorporating code tokens and relaxing data filtering. For code, English LLMs, such as PaLM [19] or Gopher [89], are trained on a small amount of code data alongside natural language data, though no benchmarking was reported to justify that decision. We investigate training LLMs on a mix of language data and Python data at 10 different mixing rates and find that mixing in code is able to provide a $2\times$ increase in effective tokens even when evaluating only natural language tasks. For filtering, we revisit perplexity and deduplication filtering strategies on both noisy and clean datasets and find that data filtering is primarily effective for noisy datasets.

2 Background

Predicting the scaling behavior of large models is critical when deciding on training resources. Specifically, two questions are of interest: *(Allocation)* What is the optimal balance of resources? *(Return)* What is the expected value of additional resources? For scaling LLMs, the resource is compute (measured in FLOPs), and it can be allocated to training a larger model or training for more

N: model parameters
D: training tokens

steps.[1] The metric used to quantify progress is the model's loss on held-out data, i.e. the ability to predict the underlying data as measured in the model's cross-entropy [2, 42]. We aim to minimize the loss (L) subject to a compute resource constraint (C) via optimal allocation to N and D as:

$$\underset{N,D}{\text{argmin}}\, L(N, D) \text{ s.t. } \text{FLOPs}(N, D) = C \tag{1}$$

Currently, there are established best practices for scaling LLMs. *Return* follows a power-law: loss scales as a power-law with the amount of compute used for training [39, 46, 6, 35, 7, 41]. *Allocation* is balanced: resources are divided roughly equally between scaling of parameters and data [42]. These scaling laws were established empirically by training LLMs and carefully extrapolating behavior.

Chinchilla [42] uses three methods for making scaling predictions:

N D
- (*Fixed Parameters*) Train with a fixed model size but on varying amounts of data.
C D
- (*Fixed FLOPs*) Train with fixed computation while parameters and training tokens vary.
- (*Parametric Fit*) Derive and fit a formula for the loss.

For the parametric fit, the loss (L) is a function of parameters (N) and training tokens (D):

$$L(N, D) = \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}} + E \tag{2}$$

Where $\{A, \alpha, B, \beta, E\}$ are learned variables fit using the training runs from the first two approaches [42]. Using these learned variables, they propose calculating the optimal allocation of compute (C) to N and D as follows:

$$\begin{gathered} N_{opt}(C) = G(C/6)^{a} \quad D_{opt}(C) = G^{-1}(C/6)^{b} \\ \text{where} \quad G = \left(\frac{\alpha A}{\beta B}\right)^{\frac{1}{\alpha+\beta}} \quad a = \frac{\beta}{\alpha+\beta} \quad b = \frac{\alpha}{\alpha+\beta} \end{gathered} \tag{3}$$

These methods lead to the conclusion that $\alpha \approx \beta$ and hence N and D should be scaled proportionally for compute-optimal training. As loss can be an imperfect proxy for performance on natural language tasks [123, 97, 105], they also validate their conclusions on various downstream tasks.

3 Method: Data-Constrained Scaling Laws

We are interested in scaling behavior in the data-constrained regime. Specifically, given a limited amount of unique data, what is the best *Allocation* of and *Return* for computational resources. Prior work [46, 42] assumes that the necessary data to support scaling is unlimited. Our aim is therefore to introduce a modified version of Equation 2 that accounts for data constraints and fit the terms in the modified scaling law to data from a large body of experiments.

The primary method we consider is *repeating* data, i.e. allocating FLOPs to multiple epochs on the same data. Given a budget of unique data D_C, we split the Chinchilla total data term D into two parts: the number of unique tokens used, U_D, and the number of repetitions, R_D (i.e. epochs - 1). Given total training tokens D and data budget D_C these terms are simply computed as $U_D = \min\{D_C, D\}$ and $R_D = (D/U_D) - 1$. When training for a single epoch like done in prior scaling studies, $R_D = 0$. We are thus interested in minimizing Equation 1 with the additional constraint of a data budget D_C:

$$\underset{N,D}{\text{argmin}}\, L(N, D) \text{ s.t. } \text{FLOPs}(N, D) = C, U_D \leq D_C \tag{4}$$

Symmetrically, for mathematical convenience, we split the parameter term N into two parts: the base number of parameters needed to optimally fit the unique tokens U_N, and the number of times

[1] In this work we use [46]'s approximation for the compute cost: $\text{FLOPs}(N, D) \approx 6ND$, where N denotes the number of model parameters and D denotes the number of tokens processed.

to "repeat" this initial allocation, R_N. We compute U_N by first rearranging Equation 3 to find the optimal compute budget for the unique tokens used (U_D). We input this value into the N_{opt} formula of Equation 3 to get $U_N = \min\{N_{opt}, N\}$. U_N thus corresponds to the compute-optimal number of parameters for U_D or less if $N < N_{opt}$. Once we have U_N, we compute the repeat value as $R_N = (N/U_N) - 1$.

To empirically explore the scaling behavior in a data-limited setting we train LLMs under these constraints. We consider three different experimental protocols in this work:

- (*Fixed Unique Data*) In §5 we fix the data constraint D_C and train models varying epochs and parameters. These experiments target *Allocation*, specifically tradeoff of D and N.
- (*Fixed FLOPs*) In §6 we fix the computation available and vary D_C (and thus also U_D and U_N). These experiments target *Return*, i.e. how well does repeating scale compared to having more unique data.
- (*Parametric Fit*) We fit a formula introduced in §3.1 on all our training runs and evaluate its predictive capability throughout §5 and §6.

Before discussing experimental results we describe the parametric assumptions.

3.1 Parametric Fit

To extrapolate scaling curves, it is necessary to incorporate repetition into the Chinchilla formula (Equation 2). We generalize Equation 2 by replacing D and N with terms corresponding to the *effective data* (D') and *effective model parameters* (N').

$$L(N, D) = \frac{A}{N'^{\alpha}} + \frac{B}{D'^{\beta}} + E$$

Intuitively, D' should be smaller or equal to D where D is the total number of processed tokens since repeated tokens provide less useful information to the model than new ones. We use an *exponential decay* formulation, where the value of a data token processed loses roughly $(1 - 1/R_D^*)$ fraction of its value per repetition, where R_D^* is a learned constant. After some derivations and approximations (see Appendix A), this boils down to

$$D' = U_D + U_D R_D^* (1 - e^{\frac{-R_D}{R_D^*}}) \,. \tag{5}$$

Note that for $R_D = 0$ (no repetitions), $D' = U_D = D$. For $R_D \ll R_D^*$, $e^{-R_D/R_D^*} \approx 1 - \frac{R_D}{R_D^*}$ and so

$$D' \approx U_D + U_D R_D^* (1 - 1 + R_D/R_D^*) = U_D(1 + R_D) = D$$

and hence in this case, repeated data is worth almost the same as fresh data. (This is also consistent with the predictions of the "deep bootstrap" framework [76].) As R_D grows, the value of repeated tokens tends to zero, and the effective data D' becomes much smaller than D. The formula implies that no matter how many times we repeat the data, we will not get a better loss than could be obtained with a single epoch on $U_D + U_D R_D^*$ fresh tokens.

Just as processing repeated tokens yields a diminishing return, both intuitively and empirically, models with sizes that vastly outstrip the available data also offer diminishing returns per parameter. Hence we use a symmetric formula for the number of effective parameters, where again R_N^* is learned,

$$N' = U_N + U_N R_N^* (1 - e^{\frac{-R_N}{R_N^*}}) \,. \tag{6}$$

The learned constants R_D^*, R_N^* roughly correspond to the "half-life" of repeated data and excess parameters. For example, at $R_D = R_D^*$, the number of effective tokens D' is $U_D + U_D R_D (1 - e^{-1})$ which means that the $U_D R_D$ repeated tokens are worth on average $1 - 1/e$ fraction of fresh ones.

Using a methodology similar to [42], R_N^* and R_D^* can be fit on empirical measurements, which yields data-driven estimates. See Appendix A for more details on the derivations and the fitting procedure.

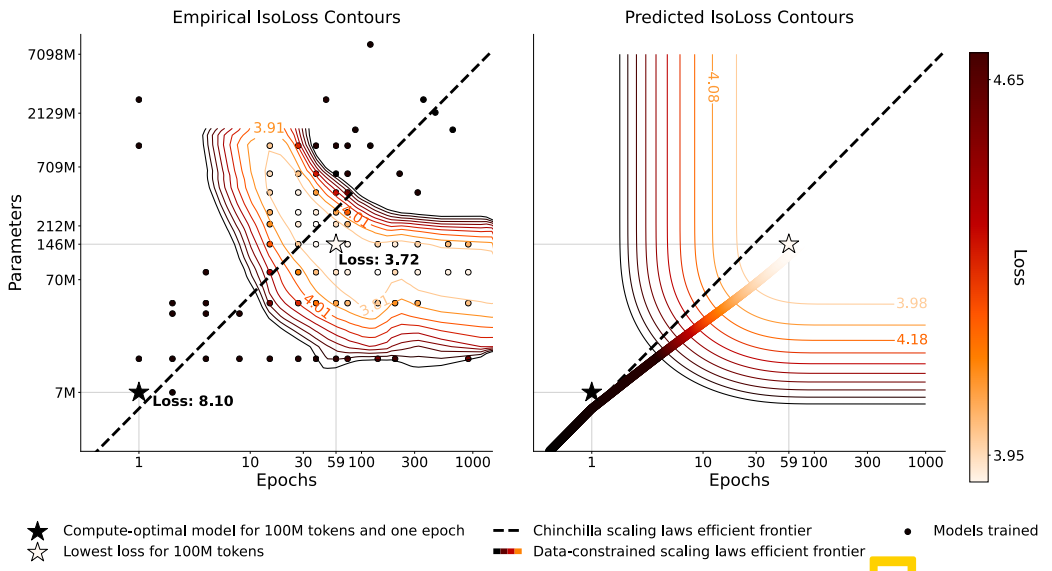


Figure 3: **IsoLoss contours for 100 million unique tokens.** *(Left):* 93 models trained with varying parameters and epochs on a fixed dataset. Contours show an interpolation of results with the same final test loss. *(Right):* Comparison with the loss predictions from our proposed scaling laws for the same budget of 100 million unique tokens and the predicted efficient frontier. The diminishing returns from training on repeated data can be seen in the increase in distance of the contour curves.

4 Experimental Setup

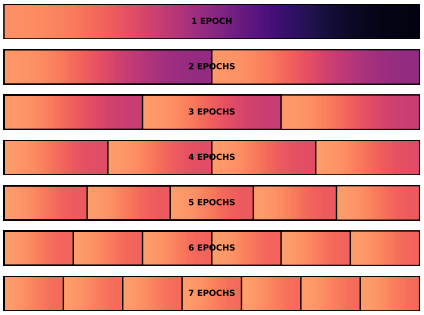


Figure 2: **Dataset setup.** We ensure that runs using less data (more epochs) always use a subset of the data used in runs with more data (fewer epochs).

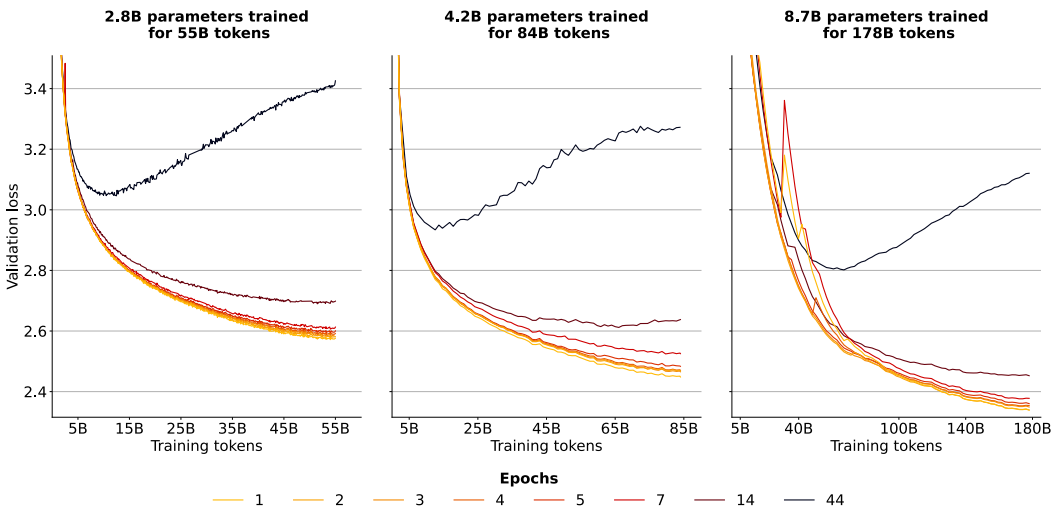
For all experiments, we train transformer language models with the GPT-2 architecture and tokenizer [88]. Models have up to 8.7 billion parameters and are trained for up to 900 billion total tokens. Following [42] we use cosine learning rate schedules that decay 10× over the course of training for each model (different schedules led to different estimates in [46]). Unlike [46], we do not use early stopping to also explore the extent of overfitting when repeating. Other hyperparameters are based on prior work [89, 42] and detailed in Appendix S. Models are trained on subsets of C4 [90]. The data constraints are carefully defined to ensure maximal overlap as shown in Figure 2. Unlike [40], we always repeat the entire available data rather than subsets of it. Data is shuffled after each epoch. As repeating data can result in extreme overfitting (see Appendix H), we report loss on a held-out test set unless otherwise specified (see Appendix K). This contrasts training loss used in [42], but should not alter our findings as the held-out data stems from the same underlying dataset.

5 Results: Resource Allocation for Data-Constrained Scaling

Our first experimental setting considers scaling in a setting where all models have the same data constraint. For these experiments, the unique training data budget D_C is fixed at either 100M, 400M or 1.5B tokens. For each data budget, we train a set of language models with increasing amounts of compute that is allocated to either more parameters or more epochs on the unique training data.

Figure 3 (left) shows the main results for scaling with 100M unique tokens[2] (see Appendix C for 400M and 1.5B tokens). For 100M tokens, the corresponding one-epoch compute-optimal model

[2]Although small, for example, this is the order of magnitude of a realistic data constraint reflecting data available after filtering the OSCAR dataset [84] for Basque, Punjabi, or Slovenian.



FLOP budget (C)	Parameters (N)	Training tokens (D)	Data budget (D_C)
9.3×10^{20}	2.8B	55B	$\{55, 28, 18, 14, 11, 9, 4, 1.25\}$B
2.1×10^{21}	4.2B	84B	$\{84, 42, 28, 21, 17, 12, 6, 1.9\}$B
9.3×10^{21}	8.7B	178B	$\{178, 88, 58, 44, 35, 25, 13, 4\}$B

Figure 4: **Validation Loss for Different Data Constraints (IsoFLOP).** Each curve represents the same number of FLOPs spent on an equal size model. Colors represent different numbers of epochs due to repeating because of data constraints. Parameters and training tokens are set to match the single-epoch compute-optimal configurations for the given FLOPs. Models trained on data that is repeated for multiple epochs have consistently worse loss and diverge if too many epochs are used.

according to scaling laws from [42] has U_N of approximately 7M parameters (see Appendix B for the scaling coefficients we use). Results show that more than a 50% reduction in loss can be attained by training for several epochs ($R_D > 0$) and increasing model size beyond what would be compute-optimal for 100M tokens ($R_N > 0$). We find the best loss to be at around 20-60$\times$ more parameters and epochs, which corresponds to spending around 7000$\times$ more FLOPs. These results suggest that one-epoch models significantly under-utilize their training data and more signal can be extracted by repeating data and adding parameters at the cost of sub-optimal compute utilization.

Figure 3 (right) shows the predicted contours created by fitting our data-constrained scaling laws on 182 training runs. In the single-epoch case ($R_D = 0$) with near compute-optimal parameters ($R_N = 0$) our scaling equation (§3.1) reduces to the Chinchilla equation. In this case, both formulas predict the optimal allocation of compute to parameters and data to be the same, resulting in overlapping efficient frontiers. As data is repeated for more than a single epoch, our fit predicts that excess parameters decay faster in value than repeated data ($R_N^* < R_D^*$). As a result, the data-constrained efficient frontier suggests allocating most additional compute to more epochs rather than more parameters. This contrasts the Chinchilla scaling laws [42], which suggest equally scaling both. However, note that they do not repeat the entire training data and their parametric fit explicitly relies on the assumption that models are trained for a single epoch only. Thus, there is no guarantee that their scaling predictions hold for repeated data.

For all three data budgets, our results suggest that *Allocation* is optimized by scaling epochs faster than additional parameters. We confirm this at scale by training the data-constrained compute-optimal model for 9.3×10^{21} FLOPs and 25 billion unique tokens as suggested by our efficient frontier. Despite having 27% less parameters, this model achieves better loss than the model suggested by the Chinchilla scaling laws (Figure 1, right). Similarly, the 120 billion parameter Galactica model trained on repeated data should have been significantly smaller according to data-constrained scaling laws (Appendix G). An additional benefit of using a smaller model is cheaper inference, though adding parameters can make it easier to parallelize training across GPUs.

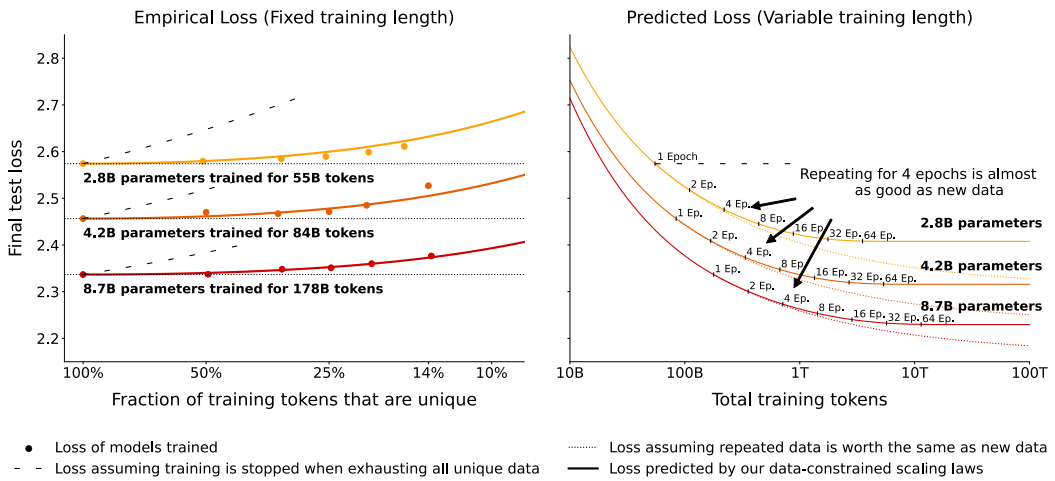


Figure 5: **Empirical and Extrapolated loss with constrained data.** *(Left):* Loss as a function of repeated tokens for three different training budgets each with fixed number of parameters. Loss curves predicted by our data-constrained scaling laws are shifted to exactly match the loss at 100% unique data. Return on FLOPs decays with repeated data in a regular pattern. *(Right):* Extrapolating from the proposed data-constrained scaling law shows that at small numbers epochs are benign, but at large number of epochs loss stops improving.

Adding parameters and epochs causes the loss to decrease and eventually increase again, suggesting that too much compute can hurt performance. Results from [46] also show that loss can increase when too many parameters are used, even with early stopping. However, we expect that appropriate regularization (such as simply removing all excess parameters as an extreme case) could prevent this behavior. Thus, our formula presented in §3 and its predicted isoLoss contours in Figure 3 do not model the possibility that excess epochs or parameters could hurt performance.

6 Results: Resource Return for Data-Constrained Scaling

Next, consider the question of *Return* on scaling. To quantify this value, we run experiments with three FLOP budgets across eight respective data budgets to compare return on FLOPs.

Figure 4 shows the configurations and validation curves for models trained on the same number of total tokens. Conforming to intuition and prior work on deduplication [55], repeated data is worth less, thus models trained on less unique data (and, correspondingly, more epochs) have consistently higher loss. However, the loss difference for a few epochs is negligible. For example, the $N = 8.7$ billion parameter model trained for four epochs ($D_C = 44$ billion unique tokens) finishes training with only 0.5% higher validation loss than the single-epoch model ($D_C = 178$ billion unique tokens).

In Figure 5 (left), we compare the final test loss of each model to predictions from our parametric fit. The data-constrained scaling laws can accurately measure the decay in the value of repeated data as seen by the proximity of empirical results (dots) and parametric fit (lines). We note however that it significantly underestimates the final test loss of failing models where loss increases midway through training, such as models trained for 44 epochs (not depicted).

In Figure 5 (right), we extrapolate the three budgets by further scaling compute while keeping the data constraints (D_C) at 55B, 84B, and 178B tokens, respectively. The parameter R_D^* introduced in §3 represents roughly the "half-life" of epochs: specifically the point where repeated tokens have lost $\frac{1}{e}$ of their value. Through our fitting in Appendix A, we found $R_D^* \approx 15$, corresponding to 15 repetitions (or 16 epochs). Graphically, this can be seen by the stark diminishing returns in the proximity of the 16-epoch marker and the flattening out soon after.

Overall, the *Return* when repeating data is relatively good. Meaningful gains from repeating data can be made up to around 16 epochs (R_D^*) beyond which returns diminish extremely fast.

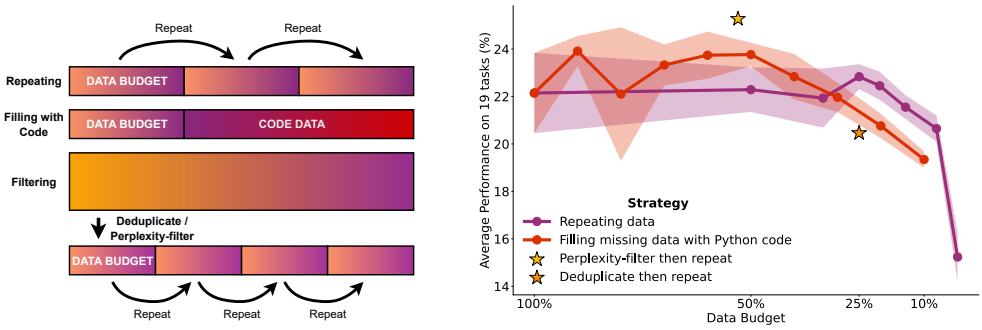


Figure 6: **Strategies for data-constrained settings and their downstream performance.** *(Left):* Schematic showing alternative data use strategies of code filling and filtering. *(Right):* $N = 4.2$ billion parameter models trained for a total of $D = 84$ billion tokens with varying budgets D_C. For repeating and filling with code, five models with different seeds are trained for each dot and the standard deviation is visualized as the shaded area.

7 Results: Complementary Strategies for Obtaining Additional Data

While repeating data is effective, it has diminishing returns. We therefore consider strategies for scaling D targeting improved downstream performance as opposed to directly minimizing loss.

Figure 6 (left) illustrates the strategies: **(a) Code augmentation:** We use Python code from The Stack [49] to make up for missing natural language data. The combined dataset consisting of code and natural language samples is shuffled randomly. **(b) Adapting filtering:** We investigate the performance impact of deduplication and perplexity filtering, two common filtering steps that can severely limit available data. Removing such filtering steps can free up additional training data.

For these experiments, we set a maximum data budget (D_C) of 84 billion tokens. For repetition and code filling, only a subset of D_C is available and the rest needs to be compensated for via repeating or adding code. For both filtering methods, we start out with approximately twice the budget (178 billion tokens), as it is easier to gather noisy data and filter it than it is to gather clean data for training. For perplexity filtering, we select the top 25% samples with the lowest perplexity according to a language model trained on Wikipedia. This results in 44 billion tokens that are repeated for close to two epochs to reach the full data budget. For deduplication filtering, all samples with a 100-char overlap are removed resulting in 21 billion tokens that are repeated for four epochs during training. See Appendix N for more details on the filtering procedures.

When comparing across data strategies, loss ceases to be a good evaluation metric as the models are trained on different data distributions. We thus evaluate models on 19 natural language tasks with zero to five in-context few-shot exemplars [15] producing 114 scores per model. As our evaluation tasks cover different metrics and random baselines, we re-scale all scores to be in the same range to better reflect performance ranges before averaging. Details on the evaluation datasets are in Appendix K.

In Figure 6 (right) we compare the downstream performance of all strategies. For repeating data, differences in downstream performance are insignificant for up to around 4 epochs (25% budget) and then start dropping, which aligns with our results on test loss in §6. Filling up to 50% of data with code (42 billion tokens) also shows no deterioration. Beyond that, performance decreases quickly on natural language tasks. However, adding more code data may benefit non-natural language tasks, which are not considered in the benchmarking. Two of the tasks benchmarked, WebNLG [17, 34], a generation task, and bAbI [122, 57], a reasoning task, see jumps in performance as soon as code is added, possibly due to code enabling models to learn long-range state-tracking capabilities beneficial for these tasks.

Of the filtering approaches, we find perplexity-filtering to be effective, while deduplication does not help. Prior work found deduplication was able to improve perplexity [55]; however, it did not evaluate on downstream tasks. Deduplication may have value not captured in our benchmark, such as reducing memorization [45, 40, 16, 10]. We also investigate filtering on a different noisier dataset in Appendix O, where we find it to be more effective. Overall, in a data-constrained regime, we recommend reserving filtering for noisy datasets and using both code augmentation and repeating to

increase data tokens. For example, first doubling the available data by adding code and then repeating the new dataset for four epochs results in 8× more training tokens that are expected to be just as good as having had 8× more unique data from the start.

8 Related Work

Large language models Scaling up transformer language models [111] across parameter count and training data has been shown to result in continuous performance gains [19]. Starting with the 1.4 billion parameter GPT-2 model [88], a variety of scaled-up language models have been trained, commonly referred to as large language models (LLMs). They can be grouped into dense models [15, 47, 58, 89, 20, 13, 132, 109, 103, 108, 130, 95, 56, 65] and sparse models [30, 131, 28, 135] depending on whether each forward pass makes use of all parameters. These models are generally pre-trained to predict the next token in a sequence, which makes them applicable to various language tasks directly after pre-training [15, 118, 50, 71, 102] by reformulating said NLP tasks as context continuation tasks (see [67] for an earlier proposal on this topic). We focus on the most common scenario, where a dense transformer model is trained to do next-token prediction on a large corpus and evaluated directly after pre-training using held-out loss or zero- to few-shot prompting.

Scaling laws Prior work has estimated an optimal allocation of compute for the training of LLMs. Kaplan et al. [46] suggested a 10× increase in compute should be allocated to a 5.5× increase in model size and a 1.8× increase in training tokens. This first scaling law has led to the creation of very large models trained on relatively little data, such as the 530 billion parameter MT-NLG model trained on 270 billion tokens [99]. More recent work [42], however, showed that model size and training data should rather be scaled in equal proportions. These findings called for a renewed focus on the scaling of pre-training data rather than scaling model size via complex parallelization strategies [98, 91, 9, 78]. Up-sampling is often employed when pre-training data is partly limited, such as data from a high-quality domain like Wikipedia or text in a rare language for training multilingual LLMs [60, 82]. Hernandez et al. [40] study up-sampling of data subsets and find that repeating only 0.1% of training data 100 times significantly degrades performance. In contrast, our work focuses on repeating the entire pre-training corpus for multiple epochs rather than up-sampling parts of it.

Alternative data strategies Large pre-training datasets are commonly filtered to remove undesired samples or reduce noise [101]. Perplexity-based filtering, whereby a trained model is used to filter out samples with high perplexity, has been found beneficial to reduce noise in web-crawled datasets [121]. Mixing of data is employed for the pre-training data of multilingual LLMs, where text data from different languages is combined [23, 126, 100, 74]. However, both for code and natural language models, mixing different (programming) languages has been reported to under-perform monolingual models [80, 113]. Some work has investigated mixing code and natural language data for prediction tasks, such as summarizing code snippets [44] or predicting function names [4]. Several pre-training datasets for LLMs include low amounts of code data [31, 89, 95]. However, these past works generally do not provide any ablation on the drawbacks of including code or the benefits for natural language task performance. We perform a detailed benchmarking of mixing Python and natural language in LLM pre-training at 10 different mixing rates.

9 Conclusion

This work studies data-constrained scaling, focusing on the optimal use of computational resources when unique data is limited. We propose an extension to the Chinchilla scaling laws that takes into account the decay in value of repeated data, and we fit this function using a large set of controlled experiments. We find that despite recommendations of earlier work, training large language models for multiple epochs by repeating data is beneficial and that scaling laws continue to hold in the multi-epoch regime, albeit with diminishing returns. We also consider complementary approaches to continue scaling models, and find that code gives the ability to scale an additional 2×. We believe that our findings will enable further scaling of language models to unlock new capabilities with current data. However, our work also indicates that there are limits on the scaling horizon. In addition to collecting additional data, researchers should explore using current data in a more effective manner.

Acknowledgments and Disclosure of Funding

This work was co-funded by the European Union under grant agreement No 101070350. The authors wish to acknowledge CSC – IT Center for Science, Finland, for generous computational resources on the LUMI supercomputer.[3] We are thankful for the immense support from teams at LUMI and AMD, especially Samuel Antao. Hugging Face provided storage and additional compute instances. This work was supported by a Simons Investigator Fellowship, NSF grant DMS-2134157, DARPA grant W911NF2010021, and DOE grant DE-SC0022199. We are grateful to Harm de Vries, Woojeong Kim, Mengzhou Xia and the EleutherAI community for exceptional feedback. We thank Loubna Ben Allal for help with the Python data and Big Code members for insightful discussions on scaling laws. We thank Thomas Wang, Helen Ngo and TurkuNLP members for support on early experiments.

References

[1] Armen Aghajanyan, Lili Yu, Alexis Conneau, Wei-Ning Hsu, Karen Hambardzumyan, Susan Zhang, Stephen Roller, Naman Goyal, Omer Levy, and Luke Zettlemoyer. 2023. Scaling Laws for Generative Mixed-Modal Language Models. *arXiv preprint arXiv:2301.03728*.

[2] Ibrahim M Alabdulmohsin, Behnam Neyshabur, and Xiaohua Zhai. 2022. Revisiting neural scaling laws in language and vision. *Advances in Neural Information Processing Systems*, 35:22300–22312.

[3] Loubna Ben Allal, Raymond Li, Denis Kocetkov, Chenghao Mou, Christopher Akiki, Carlos Munoz Ferrandis, Niklas Muennighoff, Mayank Mishra, Alex Gu, Manan Dey, et al. 2023. SantaCoder: don't reach for the stars! *arXiv preprint arXiv:2301.03988*.

[4] Miltiadis Allamanis, Earl T Barr, Christian Bird, and Charles Sutton. 2015. Suggesting accurate method and class names. In *Proceedings of the 2015 10th joint meeting on foundations of software engineering*, pages 38–49.

[5] Stephen H. Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged S. Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Xiangru Tang, Mike Tian-Jian Jiang, and Alexander M. Rush. 2022. PromptSource: An Integrated Development Environment and Repository for Natural Language Prompts.

[6] Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. 2021. Explaining neural scaling laws. *arXiv preprint arXiv:2102.06701*.

[7] Yamini Bansal, Behrooz Ghorbani, Ankush Garg, Biao Zhang, Colin Cherry, Behnam Neyshabur, and Orhan Firat. 2022. Data scaling laws in NMT: The effect of noise and architecture. In *International Conference on Machine Learning*, pages 1466–1482. PMLR.

[8] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623.

[9] Zhengda Bian, Hongxin Liu, Boxiang Wang, Haichen Huang, Yongbin Li, Chuanrui Wang, Fan Cui, and Yang You. 2021. Colossal-AI: A unified deep learning system for large-scale parallel training. *arXiv preprint arXiv:2110.14883*.

[10] Stella Biderman, USVSN Sai Prashanth, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, and Edward Raf. 2023. Emergent and Predictable Memorization in Large Language Models. *arXiv preprint arXiv:2304.11158*.

[11] Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. *arXiv preprint arXiv:2304.01373*.

[3] https://www.lumi-supercomputer.eu/

[12] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about Physical Commonsense in Natural Language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

[13] Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. GPT-NeoX-20B: An Open-Source Autoregressive Language Model. *arXiv preprint arXiv:2204.06745*.

[14] Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large scale autoregressive language modeling with mesh-tensorflow. *If you use this software, please cite it using these metadata*, 58.

[15] 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.

[16] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*.

[17] Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. 2020. The 2020 Bilingual, Bi-Directional WebNLG+ Shared Task Overview and Evaluation Results (WebNLG+ 2020). In *Proceedings of the 3rd WebNLG Workshop on Natural Language Generation from the Semantic Web (WebNLG+ 2020)*, pages 55–76, Dublin, Ireland (Virtual). Association for Computational Linguistics.

[18] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. 2020. Generative pretraining from pixels. In *International conference on machine learning*, pages 1691–1703. PMLR.

[19] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

[20] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling Instruction-Finetuned Language Models. *arXiv preprint arXiv:2210.11416*.

[21] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. In *NAACL*.

[22] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv:1803.05457v1*.

[23] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised Cross-lingual Representation Learning at Scale. *arXiv preprint arXiv:1911.02116*.

[24] Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment: First PASCAL Machine Learning Challenges Workshop, MLCW 2005, Southampton, UK, April 11-13, 2005, Revised Selected Papers*, pages 177–190. Springer.

[25] Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pages 107–124.

[26] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. 2023. Scaling vision transformers to 22 billion parameters. *arXiv preprint arXiv:2302.05442*.

[27] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819*.

[28] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR.

[29] Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the State-of-the-Art of End-to-End Natural Language Generation: The E2E NLG Challenge. *Computer Speech & Language*, 59:123–156.

[30] William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res*, 23:1–40.

[31] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. *arXiv preprint arXiv:2101.00027*.

[32] Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2021. A framework for few-shot language model evaluation.

[33] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. *arXiv preprint arXiv:2009.11462*.

[34] Sebastian Gehrmann, Tosin Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Aremu Anuoluwapo, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna Clinciu, Dipanjan Das, Kaustubh D Dhole, et al. 2021. The gem benchmark: Natural language generation, its evaluation and metrics. *arXiv preprint arXiv:2102.01672*.

[35] Behrooz Ghorbani, Orhan Firat, Markus Freitag, Ankur Bapna, Maxim Krikun, Xavier Garcia, Ciprian Chelba, and Colin Cherry. 2021. Scaling laws for neural machine translation. *arXiv preprint arXiv:2109.07740*.

[36] Himanshu Gupta, Saurabh Arjun Sawant, Swaroop Mishra, Mutsumi Nakamura, Arindam Mitra, Santosh Mashetty, and Chitta Baral. 2023. Instruction Tuned Models are Quick Learners. *arXiv preprint arXiv:2306.05539*.

[37] Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland. Association for Computational Linguistics.

[38] Peter Henderson, Mark Krass, Lucia Zheng, Neel Guha, Christopher D Manning, Dan Jurafsky, and Daniel Ho. 2022. Pile of law: Learning responsible data filtering from the law and a 256gb open-source legal dataset. *Advances in Neural Information Processing Systems*, 35:29217–29234.

[39] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. 2020. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*.

[40] Danny Hernandez, Tom Brown, Tom Conerly, Nova DasSarma, Dawn Drain, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Tom Henighan, Tristan Hume, et al. 2022. Scaling Laws and Interpretability of Learning from Repeated Data. *arXiv preprint arXiv:2205.10487*.

[41] Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. 2021. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*.

[42] 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. 2022. Training Compute-Optimal Large Language Models. *arXiv preprint arXiv:2203.15556*.

[43] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. 2018. Music transformer. *arXiv preprint arXiv:1809.04281*.

[44] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2073–2083.

[45] Nikhil Kandpal, Eric Wallace, and Colin Raffel. 2022. Deduplicating Training Data Mitigates Privacy Risks in Language Models.

[46] 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. *arXiv preprint arXiv:2001.08361*.

[47] Mikhail Khrushchev, Ruslan Vasilev, Alexey Petrov, and Nikolay Zinov. 2022. YaLM 100B.

[48] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[49] Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, et al. 2022. The Stack: 3 TB of permissively licensed source code. *arXiv preprint arXiv:2211.15533*.

[50] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.

[51] Aran Komatsuzaki. 2019. One epoch is all you need. *arXiv preprint arXiv:1906.06669*.

[52] Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

[53] Faisal Ladhak, Esin Durmus, Claire Cardie, and Kathleen McKeown. 2020. WikiLingua: A new benchmark dataset for cross-lingual abstractive summarization. *arXiv preprint arXiv:2010.03093*.

[54] Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro Von Werra, Chenghao Mou, Eduardo González Ponferrada, Huu Nguyen, et al. 2022. The BigScience ROOTS Corpus: A 1.6 TB Composite Multilingual Dataset. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[55] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2021. Deduplicating Training Data Makes Language Models Better. *arXiv preprint arXiv:2107.06499*.

[56] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav

Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2023. StarCoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

[57] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

[58] Opher Lieber, Or Sharir, Barak Lenz, and Yoav Shoham. 2021. Jurassic-1: Technical details and evaluation. *White Paper. AI21 Labs*, 1.

[59] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

[60] Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, et al. 2021. Few-shot learning with multilingual language models. *arXiv preprint arXiv:2112.10668*.

[61] Yu-Hsiang Lin, Chian-Yu Chen, Jean Lee, Zirui Li, Yuyan Zhang, Mengzhou Xia, Shruti Rijhwani, Junxian He, Zhisong Zhang, Xuezhe Ma, et al. 2019. Choosing transfer languages for cross-lingual learning. *arXiv preprint arXiv:1905.12688*.

[62] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The Flan Collection: Designing Data and Methods for Effective Instruction Tuning. *arXiv preprint arXiv:2301.13688*.

[63] Shayne Longpre, Robert Mahari, Anthony Chen, Naana Obeng-Marnu, Damien Sileo, William Brannon, Niklas Muennighoff, Nathan Khazam, Jad Kabbara, Kartik Perisetla, Xinyi Wu, Enrico Shippole, Kurt Bollacker, Tongshuang Wu, Luis Villa, Sandy Pentland, Deb Roy, and Sara Hooker. 2023. The Data Provenance Initiative: A Large Scale Audit of Dataset Licensing & Attribution in AI.

[64] Shayne Longpre, Gregory Yauney, Emily Reif, Katherine Lee, Adam Roberts, Barret Zoph, Denny Zhou, Jason Wei, Kevin Robinson, David Mimno, and Daphne Ippolito. 2023. A Pretrainer's Guide to Training Data: Measuring the Effects of Data Age, Domain Coverage, Quality, & Toxicity.

[65] Risto Luukkonen, Ville Komulainen, Jouni Luoma, Anni Eskelinen, Jenna Kanerva, Hanna-Mari Kupari, Filip Ginter, Veronika Laippala, Niklas Muennighoff, Aleksandra Piktus, Thomas Wang, Nouamane Tazi, Teven Le Scao, Thomas Wolf, Osma Suominen, Samuli Sairanen, Mikko Merioksa, Jyrki Heinonen, Aija Vahtola, Samuel Antao, and Sampo Pyysalo. 2023. FinGPT: Large Generative Models for a Small Language. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

[66] Ali Madani, Bryan McCann, Nikhil Naik, Nitish Shirish Keskar, Namrata Anand, Raphael R Eguchi, Po-Ssu Huang, and Richard Socher. 2020. Progen: Language modeling for protein generation. *arXiv preprint arXiv:2004.03497*.

[67] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The Natural Language Decathlon: Multitask Learning as Question Answering. *CoRR*, abs/1806.08730.

[68] Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2021. Metaicl: Learning to learn in context. *arXiv preprint arXiv:2110.15943*.

[69] Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James Allen. 2017. Lsdsem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 46–51.

[70] Niklas Muennighoff. 2020. Vilio: State-of-the-art visio-linguistic models applied to hateful memes. *arXiv preprint arXiv:2012.07788*.

[71] Niklas Muennighoff. 2022. SGPT: GPT Sentence Embeddings for Semantic Search. *arXiv preprint arXiv:2202.08904*.

[72] Niklas Muennighoff, Qian Liu, Armel Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro von Werra, and Shayne Longpre. 2023. OctoPack: Instruction Tuning Code Large Language Models. *arXiv preprint arXiv:2308.07124*.

[73] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. MTEB: Massive Text Embedding Benchmark. *arXiv preprint arXiv:2210.07316*.

[74] Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. 2022. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*.

[75] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. 2021. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003.

[76] Preetum Nakkiran, Behnam Neyshabur, and Hanie Sedghi. 2021. The Deep Bootstrap Framework: Good Online Learners are Good Offline Generalizers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

[77] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium.

[78] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. 2021. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15.

[79] Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A New Benchmark for Natural Language Understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

[80] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*.

[81] nostalgebraist. 2022. chinchilla's wild implications. *lesswrong*.

[82] Gabriel Orlanski, Kefan Xiao, Xavier Garcia, Jeffrey Hui, Joshua Howland, Jonathan Malmaud, Jacob Austin, Rishah Singh, and Michele Catasta. 2023. Measuring The Impact Of Programming Language Distribution. *arXiv preprint arXiv:2302.01973*.

[83] Pedro Javier Ortiz Su'arez, Laurent Romary, and Benoit Sagot. 2020. A Monolingual Approach to Contextualized Word Embeddings for Mid-Resource Languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1703–1714, Online. Association for Computational Linguistics.

[84] Pedro Javier Ortiz Su'arez, Benoit Sagot, and Laurent Romary. 2019. Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures. Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-7) 2019. Cardiff, 22nd July 2019, pages 9 – 16, Mannheim. Leibniz-Institut f"ur Deutsche Sprache.

[85] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

[86] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data, and Web Data Only. *arXiv preprint arXiv:2306.01116*.

[87] Shrimai Prabhumoye, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2023. Adding Instructions during Pretraining: Effective Way of Controlling Toxicity in Language Models. *arXiv preprint arXiv:2302.07388*.

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

[89] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

[90] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

[91] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

[92] Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of Plausible Alternatives: An Evaluation of Commonsense Causal Reasoning. In *AAAI spring symposium: logical formalizations of commonsense reasoning*, pages 90–95.

[93] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

[94] Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2022. Multitask Prompted Training Enables Zero-Shot Task Generalization. In *The Tenth International Conference on Learning Representations*.

[95] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

[96] Teven Le Scao, Thomas Wang, Daniel Hesslow, Lucile Saulnier, Stas Bekman, M Saiful Bari, Stella Bideman, Hady Elsahar, Niklas Muennighoff, Jason Phang, et al. 2022. What Language Model to Train if You Have One Million GPU Hours? *arXiv preprint arXiv:2210.15424*.

[97] Seongjin Shin, Sang-Woo Lee, Hwijeen Ahn, Sungdong Kim, HyoungSeok Kim, Boseop Kim, Kyunghyun Cho, Gichang Lee, Woomyoung Park, Jung-Woo Ha, et al. 2022. On the effect of pretraining corpora on in-context learning by a large-scale language model. *arXiv preprint arXiv:2204.13509*.

[98] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

[99] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*.

[100] Saleh Soltan, Shankar Ananthakrishnan, Jack FitzGerald, Rahul Gupta, Wael Hamza, Haidar Khan, Charith Peris, Stephen Rawls, Andy Rosenbaum, Anna Rumshisky, et al. 2022. Alexatm 20b: Few-shot learning using a large-scale multilingual seq2seq model. *arXiv preprint arXiv:2208.01448*.

[101] Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari S Morcos. 2022. Beyond neural scaling laws: beating power law scaling via data pruning. *arXiv preprint arXiv:2206.14486*.

[102] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

[103] Hui Su, Xiao Zhou, Houjing Yu, Yuwen Chen, Zilin Zhu, Yang Yu, and Jie Zhou. 2022. WeLM: A Well-Read Pre-trained Language Model for Chinese. *arXiv preprint arXiv:2209.10372*.

[104] Yi Tay, Mostafa Dehghani, Samira Abnar, Hyung Won Chung, William Fedus, Jinfeng Rao, Sharan Narang, Vinh Q Tran, Dani Yogatama, and Donald Metzler. 2022. Scaling Laws vs Model Architectures: How does Inductive Bias Influence Scaling? *arXiv preprint arXiv:2207.10551*.

[105] Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. 2021. Scale efficiently: Insights from pre-training and fine-tuning transformers. *arXiv preprint arXiv:2109.10686*.

[106] Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Neil Houlsby, and Donald Metzler. 2022. Unifying Language Learning Paradigms. *arXiv preprint arXiv:2205.05131*.

[107] Yi Tay, Jason Wei, Hyung Won Chung, Vinh Q Tran, David R So, Siamak Shakeri, Xavier Garcia, Huaixiu Steven Zheng, Jinfeng Rao, Aakanksha Chowdhery, et al. 2022. Transcending scaling laws with 0.1% extra compute. *arXiv preprint arXiv:2210.11399*.

[108] Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*.

[109] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

[110] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

[111] 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.

[112] Pablo Villalobos, Jaime Sevilla, Lennart Heim, Tamay Besiroglu, Marius Hobbhahn, and Anson Ho. 2022. Will we run out of data? An analysis of the limits of scaling datasets in Machine Learning. *arXiv preprint arXiv:2211.04325*.

[113] Antti Virtanen, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter, and Sampo Pyysalo. 2019. Multilingual is not enough: BERT for Finnish. *arXiv preprint arXiv:1912.07076*.

[114] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. *CoRR*, abs/1905.00537.

[115] Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, et al. 2023. How Far Can Camels Go? Exploring the State of Instruction Tuning on Open Resources. *arXiv preprint arXiv:2306.04751*.

[116] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-Instruct: Aligning Language Model with Self Generated Instructions. *arXiv preprint arXiv:2212.10560*.

[117] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

[118] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

[119] Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, et al. 2021. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*.

[120] Johannes Welbl, Nelson F Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*.

[121] Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2019. CCNet: Extracting high quality monolingual datasets from web crawl data. *arXiv preprint arXiv:1911.00359*.

[122] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart Van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

[123] Mengzhou Xia, Mikel Artetxe, Chunting Zhou, Xi Victoria Lin, Ramakanth Pasunuru, Danqi Chen, Luke Zettlemoyer, and Ves Stoyanov. 2022. Training Trajectories of Language Models Across Scales. *arXiv preprint arXiv:2212.09803*.

[124] Mengzhou Xia, Guoqing Zheng, Subhabrata Mukherjee, Milad Shokouhi, Graham Neubig, and Ahmed Hassan Awadallah. 2021. MetaXL: Meta representation transformation for low-resource cross-lingual learning. *arXiv preprint arXiv:2104.07908*.

[125] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

[126] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mT5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

[127] Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. 2021. Tuning large neural networks via zero-shot hyperparameter transfer. *Advances in Neural Information Processing Systems*, 34:17084–17097.

[128] Zheng-Xin Yong, Hailey Schoelkopf, Niklas Muennighoff, Alham Fikri Aji, David Ifeoluwa Adelani, Khalid Almubarak, M Saiful Bari, Lintang Sutawika, Jungo Kasai, Ahmed Baruwa, et al. 2022. BLOOM+ 1: Adding Language Support to BLOOM for Zero-Shot Prompting. *arXiv preprint arXiv:2212.09535*.

[129] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

[130] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. GLM-130B: An Open Bilingual Pre-trained Model. *arXiv preprint arXiv:2210.02414*.

[131] Wei Zeng, Xiaozhe Ren, Teng Su, Hui Wang, Yi Liao, Zhiwei Wang, Xin Jiang, ZhenZhang Yang, Kaisheng Wang, Xiaoda Zhang, et al. 2021. PanGu-alpha: Large-scale Autoregressive Pretrained Chinese Language Models with Auto-parallel Computation. *arXiv preprint arXiv:2104.12369*.

[132] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

[133] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

[134] Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*.

[135] Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. Designing effective sparse expert models. *arXiv preprint arXiv:2202.08906*.