What Language Model Architecture and Pretraining Objective Work Best for Zero-Shot Generalization?

The BigScience Architecture & Scaling Group

Thomas Wang[1*] **Adam Roberts**[2*]

Daniel Hesslow[3] **Teven Le Scao**[1] **Hyung Won Chung**[2]

Iz Beltagy[4] **Julien Launay**[3,5†] **Colin Raffel**[1†]

[1] Hugging Face [2]Google [3]LightOn

[4]Allen Institute for AI [5]LPENS, École Normale Supérieure

Abstract

Large pretrained Transformer language models have been shown to exhibit *zero-shot generalization*, i.e. they can perform a wide variety of tasks that they were not explicitly trained on. However, the architectures and pretraining objectives used across state-of-the-art models differ significantly, and there has been limited systematic comparison of these factors. In this work, we present a large-scale evaluation of modeling choices and their impact on zero-shot generalization. In particular, we focus on text-to-text models and experiment with three model architectures (causal/non-causal decoder-only and encoder-decoder), trained with two different pretraining objectives (autoregressive and masked language modeling), and evaluated with and without multitask prompted finetuning. We train models with over 5 billion parameters for more than 170 billion tokens, thereby increasing the likelihood that our conclusions will transfer to even larger scales. Our experiments show that causal decoder-only models trained on an autoregressive language modeling objective exhibit the strongest zero-shot generalization after purely unsupervised pretraining. However, models with non-causal visibility on their input trained with a masked language modeling objective followed by multitask finetuning perform the best among our experiments. We therefore consider the adaptation of pretrained models across architectures and objectives. We find that pretrained non-causal decoder models can be adapted into performant generative causal decoder models, using autoregressive language modeling as a downstream task. Furthermore, we find that pretrained causal decoder models can be efficiently adapted into non-causal decoder models, ultimately achieving competitive performance after multitask finetuning. Code and checkpoints are available at https://github.com/bigscience-workshop/architecture-objective.

[*]Equal contribution.

[†]Equal supervision.

Individual contributions outlined in Appendix A

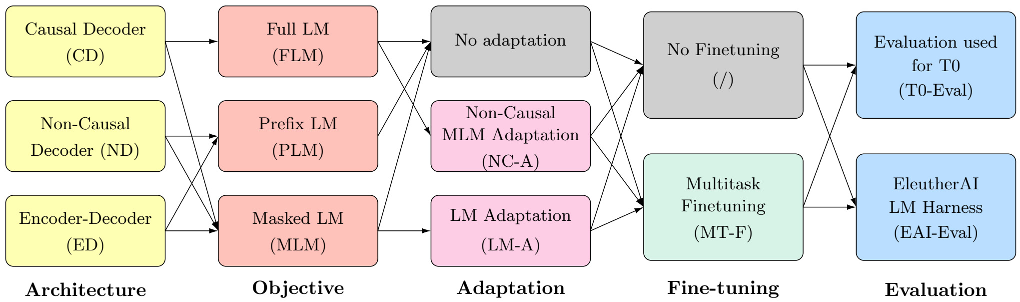

Figure 1: **We perform an extensive study of architecture, objective, adaptation, and finetuning impact on zero-shot generalization.** With 6 pre-trained models of over 5 billion parameters each trained on 168 billion tokens, adaptations on up to 100 billion tokens, multi-task finetuning on 13 billion tokens, and evaluation on 30 tasks from 2 different benchmarks with varied prompts, our study is the largest to date on the influence of modeling decisions on zero-shot generalization.

1 Introduction

Large language models (LLMs) pretrained on unstructured text data have been shown to be capable of performing a wide variety of text processing tasks without additional training. This ability has been referred to as **zero-shot generalization** since these models are typically pretrained with a self-supervised objective that is not specific to a downstream task. Zero-shot generalization is particularly useful because it does not require any additional data or training in order to enable the model to perform a given task. As such, there has been an explosion of work on developing LLMs and training techniques that produce strong zero-shot generalization [Brown et al., 2020, Wang and Komatsuzaki, 2021, Du et al., 2021, Lin et al., 2021, Rae et al., 2021, Hoffmann et al., 2022, Chowdhery et al., 2022]. One recent line of work [Sanh et al., 2021, Wei et al., 2021, Xu et al., 2022] has demonstrated that adding an explicit **multitask finetuning stage** on an ensemble of prompted tasks after pretraining can significantly boost the zero-shot capabilities of LLMs.

Modern LLMs are based on the **Transformer architecture** [Vaswani et al., 2017]. While the original Transformer included a separate encoder that processes input text and a decoder that generates target text, most recent LLMs are causal decoder-only (CD) models trained to autoregressively predict a text sequence [Liu et al., 2018, Radford et al., 2018, Al-Rfou et al., 2019]. In contrast with this trend, Raffel et al. [2020] has shown that encoder-decoder (ED) models outperform decoder-only LLMs for transfer learning (i.e. where a pretrained model is finetuned on a single downstream task). Non-causal decoders (ND) [Liu et al., 2018, Dong et al., 2019] use a modified attention mask to bridge the gap between decoder-only and encoder-decoder models. However, they have seen limited adoption. Recently, Sanh et al. [2021] proposed a multitask finetuned encoder-decoder LLM that outperforms decoder-only models on zero-shot generalization, despite being an order of magnitude smaller. Concurrent work also demonstrated this approach with a decoder-only model [Wei et al., 2021]. This begs the question as to whether an encoder-decoder or a decoder-only would be a better choice for zero-shot generalization, especially if used in conjunction with multitask finetuning.

Transformer models can be trained with a variety of **unsupervised training objectives**. Typically, decoder-only LLMs are pretrained using a *full* language modeling (FLM) objective with a loss computed on all tokens [Dai and Le, 2015, Radford et al., 2018], and encoder-decoder models with a *masked* language modeling (MLM) objective [Taylor, 1953, Devlin et al., 2018], such as span corruption [Raffel et al., 2019, Joshi et al., 2020]. It has repeatedly been shown that an MLM objective produces a better pretrained model for subsequent supervised finetuning in transfer learning settings [Devlin et al., 2018, Lample and Conneau, 2019, Voita et al., 2019, Raffel et al., 2020]. The frequent use of the standard full language modeling objective nonetheless could be attributed to the fact that it lends itself to straightforward application of the model to many downstream tasks [Radford et al., 2019]. Still, the effectiveness of MLM in the transfer learning setting suggests it could create LLMs that are better suited to multitask finetuning. Notably, the T0 model of Sanh et al. [2021] used an MLM pretraining objective, which may have contributed to its strong performance relative to larger models trained with only an FLM objective. Recently, Lester et al. [2021] also proposed introducing an **adaptation stage** (i.e. extending pretraining but with a different objective) to enable an MLM model to perform prompted text generation tasks, bridging the gap across objectives.

These results indicate a need for a more systematic analysis of which architecture and pretraining objective pair produces LLMs with the strongest zero-shot generalization capabilities. Past studies on architectures and objectives for language models [e.g. Narang et al., 2021, Raffel et al., 2020] have focused mainly on the transfer learning setting, with models that were orders of magnitude smaller than the current state-of-the-art. Furthermore, recent results demonstrating the effectiveness of multitask finetuning raise the question of which architecture and pretraining objective is best suited to that promising setting. Finally, novel adaptation practices also question the rigidity of these architecture and objective choices, and whether it is possible to efficiently convert pretrained models from one architecture to another. We propose to fill this gap with the following contributions.

Large-scale systematic study. We undertake a study of **architecture** and **pretraining objective** combinations for LLMs with a focus on **zero-shot generalization**. We consider decoder-only and encoder-decoder models using standard, prefix, and masked language modeling, spanning six ⟨architecture, objective⟩ pairs. We also evaluate performance with and without **multitask finetuning**. In hopes of producing insights that transfer to very large models, we undertake our experiments at large scale: we train models with 5 billion parameters (11 for encoder-decoder) on 168 billion tokens, and perform multitask finetuning on 13 billion tokens. We base our zero-shot evaluation on the set of tasks used by Sanh et al. [2021] (T0-Eval) and the EleutherAI language model evaluation harness (EAI-Eval, Gao et al. [2021]), totalling 30 different datasets with varied prompt styles. Figure 1 provides an overview of our study, Section 2 introduces background on the LLM architectures, objectives, training strategies, and evaluations considered, and Section 3 details our methods.

Multitask finetuning impacts architecture and objective choice. We find that the popular recipe of a decoder-only model trained with a standard FLM objective performs best when zero-shot capabilities are measured immediately after pretraining, without any finetuning or adaptation. However, after multitask finetuning, the results are the opposite: models pretrained with MLM perform significantly better and decoder-only models perform worse. These experimental results are discussed in Section 4.

Bridging across architectures and objectives with adaptation. This discrepancy motivates us to explore the practice of adaptation (i.e. extending the pretraining of a model with a different architecture/objective) as a way to efficiently obtain both a model suited to generative use cases and to mutitask finetuning. We first consider **language modeling adaptation**: adapting an MLM-trained non-causal decoder model by converting it to a causal decoder and extending its pretraining with an FLM objective. We find that using a pretrained model in this way speeds up convergence on the language modeling task by a factor 1.6x. We then explore **non-causal MLM adaptation**, starting from a causal decoder trained with an FLM objective, converting it to a non-causal decoder, and expanding its pretraining with an MLM objective. This form of adaptation produces a new version of the model suited for multitask finetuning, achieving second-best performance across our benchmarks. Convergence on the MLM task is sped-up by 3.3x, making this the most efficient approach to obtain two distinct models for generative tasks and multitask finetuning. We detail these results in Section 5.

Accordingly, our results both confirm the validity of current standard practices, and help better understand the interplay between architecture, objective, multi-task fine-tuning, and zero-shot generalization. They also identify novel paths forward for efficiently obtaining better LLMs suited to either purely generative prompted usecases, or for multitask finetuning, as discussed in Section 6.

2 Background

2.1 Architectures

Transformer. Virtually all state-of-the-art LLMs are based on the Transformer architecture [Vaswani et al., 2017]. Due to its ubiquity, we only highlight a few relevant high-level characteristics. The main architectural unit of the Transformer is a Transformer *block*, which consists of (at minimum) multi-headed self attention [Cheng et al., 2016], layer normalization [Ba et al., 2016], a dense two-layer feedforward network, and residual connections [He et al., 2016]. A Transformer *stack* is a sequence of such blocks. In NLP applications, the Transformer ingests and outputs *tokens*. Since being introduced by Vaswani et al. [2017], various **architectural variants** of the Transformer have been proposed. A major difference between these architectures is the masking pattern applied to the provided *inputs*, which act as contextual information for the model to make a prediction. Figure 2 showcases the attention masking patterns in the three architectural variants we consider.

Encoder-decoder. As originally proposed, the Transformer consisted of two stacks: an encoder and a decoder. The encoder is fed the sequence of input tokens and outputs a sequence of vectors of the same length as the input. Then, the decoder autoregressively predicts the target sequence, token by token, conditioned on the output of the encoder. To achieve this conditioning, the decoder includes cross-attention layers in each of its blocks, allowing the decoder to also attend to the output of the encoder. The self-attention layers in the decoder utilize a *causal* masking pattern that prevents the model from attending to future tokens when predicting the output sequence (see Figure 2, on the right). We hereafter refer to this architecture as the **encoder-decoder (ED)**. Notable pretrained language models using an encoder-decoder architecture include BART [Lewis et al., 2019] and T5 [Raffel et al., 2020]. T5 in particular was recently used as the foundation for the T0 model [Sanh et al., 2021], which leveraged large-scale multitask finetuning to achieve strong zero-shot generalization, outperforming decoder-only models an order of magnitude larger.

Causal decoder-only. Although the encoder-decoder is the original Transformer variant, most recent LLMs use a decoder-only architecture. These models can be trained as a traditional language model (i.e. to predict the next token in a sequence). Decoder-only models have no independent means of processing or representing the input sequence and target sequence differently–all tokens are processed in an equivalent fashion, and, because of the causal masking pattern, conditioning is simply based on past tokens (see Figure 2, on the left). On the one hand, this means that the representation for any conditioning text is inherently weaker; on the other hand, it yields a simpler architecture that is naturally suited to a standard autoregressive next-step-prediction pretraining objective. We refer to this architecture as **causal decoder-only (CD)**. Most notably, the CD architecture makes up the backbone of the GPT series of models [Radford et al., 2018, 2019, Brown et al., 2020] as well as many other recent record-breaking LLMs [Zeng et al., 2021, Kim et al., 2021, Smith et al., 2022, Thoppilan et al., 2022, Rae et al., 2021, Hoffmann et al., 2022, Chowdhery et al., 2022].

Non-causal decoder-only. To allow decoder-only models to build richer non-causal representations of the input/conditioning text, it has been proposed to simply modify the attention mask used. Specifically, the self-attention masking pattern can be changed so that the region of the input sequence corresponding to conditioning information has a non-causal mask (i.e. attention in this region is not restricted to past tokens, see middle of Figure 2), as in the encoder of an encoder-decoder architecture. We refer to this architecture as **non-causal decoder-only (ND)**. Sometimes called a *prefix* language model, this approach was introduced by [Liu et al., 2018] and was later explored as an architectural variant by [Raffel et al., 2020, Wu et al., 2021]. Despite single-task finetuning performance nearly on par with encoder-decoder models [Raffel et al., 2020], it has seen limited adoption in the literature.

Encoder-only. As an aside, we note that another popular architectural variant is to only use a Transformer encoder layer stack. This model architecture underlies the ubiquitous BERT [Devlin et al., 2018] and its derivatives. However, this architecture is limited to producing the same number of tokens as it was fed as input, considerably limiting its applicability and making it only rarely used in the zero-shot setting [Tamborrino et al., 2020]. We therefore omit it from consideration.

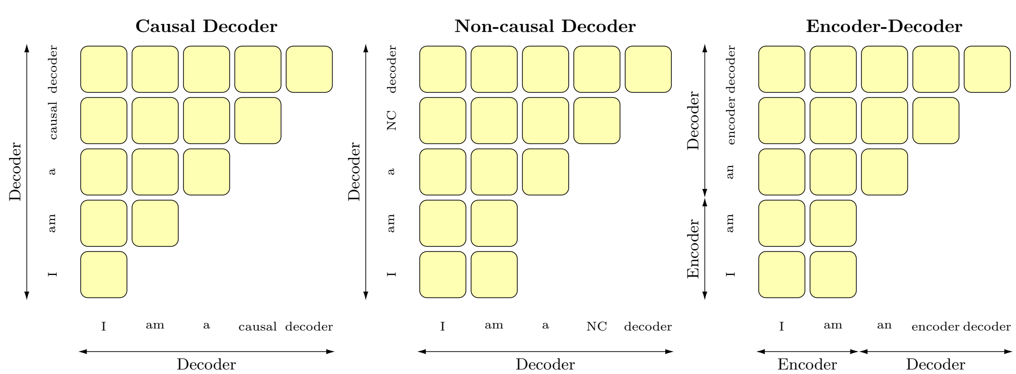

Figure 2: **Attention patterns in a causal decoder, non-causal decoder, and encoder-decoder architecture**. In a causal decoder, each token attends to the previous tokens only. In both non-causal decoder and encoder-decoder, attention is allowed to be bidirectional on any conditioning information. For the encoder-decoder, that conditioning is fed into the encoder part of the model.

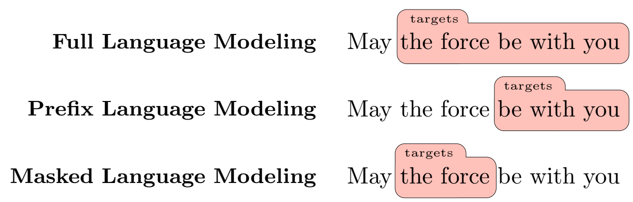

Figure 3: **Input and targets tokens in full, prefix, and masked language modeling training objectives.** For full language modeling, all tokens in a sequence are used during training. For prefix language modeling, we randomly select a prefix size, and hence only half of the tokens are used on average to derive the loss. At inference time, the prefix would be over the input/conditioning information. Finally, for masked language modeling, we mask 15% of the tokens, in spans of 3 tokens on average. We use sentinel tokens to replace spans (not represented here), and the model outputs subsequently each sentinel followed by its prediction of the content masked by the sentinel.

Comparisons across architectures. Decoder-only models process a single sequence consisting of the concatenation of the input and target text. On the other hand, in an encoder-decoder, the encoder processes only the input and the decoder processes only the target. The total *amount of computation* performed by an encoder-decoder will therefore be approximately equivalent to a decoder-only model when the encoder and decoder each have as many parameters as the entire decoder-only model (ignoring the cross-attention layers in the encoder-decoder). However, such an encoder-decoder will have twice the parameters of the decoder-only model, and hence twice the memory footprint.

2.2 Pretraining objectives

An important step in building LLMs is pretraining, where the model is trained on a large, unlabeled dataset via self-supervision. The choice of **pretraining objective** can have significant impact on the downstream usability of the LLM, and we therefore include objective choice as a factor in our empirical study. Figure 3 outlines the input and target tokens for the pretraining objectives considered.

Language modeling. Since the advent of GPT-2 [Radford et al., 2019], large decoder-only models have generally been pretrained with an autoregressive language modeling objective [Brown et al., 2020, Wu et al., 2021, Rae et al., 2021]. Given previous tokens, the model is tasked with predicting the following one. We refer to this as **full language modeling (FLM)**. This objective is particularly efficient during pretraining: all tokens in a sequence can generate a loss signal in parallel. At inference time, the model is iteratively asked to predict the next token.

Prefix language modeling. For encoder-decoder and non-causal decoder-only models to perform language modeling, one can define a prefix where the attention mask is allowed to be non-causal. Similar to standard language modeling, the model is tasked to predict each token outside the prefix given all previous tokens. We hereafter refer to this objective as **prefix language modeling (PLM)**. Loss on the prefix is ignored as tokens in the prefix can attend to their targets. For inference, the prefix is naturally the input text; during pretraining, it is usually chosen at random for each sample.

Masked language modeling. Encoder-only models, such as BERT [Devlin et al., 2018], have typically been pretrained with a masked language modeling objective. Tokens or spans of tokens in the input text are replaced with a special mask token and the model is trained to predict the missing tokens. Raffel et al. [2020] introduced a version of this objective adapted to text-to-text models in the form of *span corruption*: sentinel tokens are used to flag masked spans of short random lengths, and, after processing the masked input, the model outputs the sentinels followed by their respective predicted content. We refer to this approach as **masked language modeling (MLM)**.

2.3 Model adaptation

Adaptation extends pretraining with a different objective and/or architecture. In contrast with finetuning, no new *downstream* data is used, only additional pretraining data. **Language modeling adaptation (LM-A)** takes a model pretrained with MLM and extend its training with PLM or FLM. It has been used to convert encoder-decoder models pretrained with MLM, such as T5, into better generative models. Notably, it is used as a first step before prompt tuning [Lester et al., 2021] and also to prepare the model before multitask finetuning in T0 [Sanh et al., 2021]. When we perform language modeling adaptation on a non-causal decoder-only model, we convert it into a causal decoder-only by simply switching the attention mask. Furthermore, we propose to study the opposite adaptation: starting from a causal decoder pretrained with FLM, we cast the model into a non-causal decoder (again by switching the attention mask) and we extend pretraining with MLM. We call this approach **non-causal MLM adaptation (NC-A)**; to our knowledge, this is an entirely novel practice.

2.4 Multitask finetuning

Modern pretraining corpora are typically massive preprocessed generalist webcrawls [Ortiz Suárez et al., 2019, Raffel et al., 2020], collected with no explicit regard for downstream tasks–although adding curated high-quality cross-domain data has been proposed as a path towards better zero-shot generalization [Gao et al., 2020, Scao et al., 2022]. Recently, Sanh et al. [2021] (on an encoder-decoder model trained with MLM) and Wei et al. [2021] (on a causal decoder-only model trained with FLM) explored the potential of explicitly finetuning the model to solve multiple tasks in order to bolster zero-shot generalization. This is done by finetuning the model on a dataset of prompted tasks (i.e. in a natural language format, leveraging prompt templates applied over many datasets), which ultimately improves zero-shot performance over purely unsupervised pretraining. We refer to this as **multitask finetuning (MT-F)**, and use the openly available datasets and prompts developed for T0.

2.5 Zero-shot evaluation

Radford et al. [2019] first demonstrated that LLMs display zero-shot capabilities: given sufficient scale, language models are able to perform many tasks without having explicitly accessed any supervised samples. Zero-shot use of language models relies on a technique called *prompting*, where tasks are formulated in a natural language format (in accordance with the pretraining objective). The template applied to each example to convert it to this format is called the prompt. Unfortunately, models can exhibit significant sensitivity to the wording of the prompt [Sanh et al., 2021], and it can be difficult to diagnose whether poor performance is a prompt- or model-related problem.

Zero-shot capabilities are of increasing interest in the community, as evidenced by most record-breaking LLMs only reporting zero/few-shot results [Brown et al., 2020, Smith et al., 2022, Rae et al., 2021, Chowdhery et al., 2022]. There are many reasons why zero-shot use is gaining such traction: it does not require any labeled examples, it removes the complexity of model finetuning and deployment, and it also tests generalization to unseen tasks.

We rely on two evaluation benchmarks aggregating prompts across NLP tasks, totalling 30 tasks: the **EleutherAI LM Evaluation Harness (EAI-Eval)** [Gao et al., 2021], which reimplements the prompts from Brown et al. [2020] and is aimed at evaluation of FLM-trained causal decoder-only models, and the **evaluation set from T0 (T0-Eval)** [Sanh et al., 2021]. Note that EAI-Eval only includes one prompt per task, whereas performance on T0-Eval is averaged over many prompts. Hence, when reporting performance on T0-Eval, we report a spread across prompts, giving an indication of the impact of the choice of prompt on performance.

3 Methods

To better understand how architecture, pretraining objective, multitask finetuning, and possible adaptations influence zero-shot performance, we undertake a systematic large-scale study. We pretrain all possible $\langle$architecture, objective$\rangle$ pairs on 168 billion tokens from C4, consider intermediate multitask finetuning, and finally evaluate zero-shot performance. We also study the possibility of using adaptation to efficiently transfer the benefits from one architecture/objective to another.

Compute budget guideline. Different architectures and objectives come with different compute trade-offs. We aim to make the training budget similar across all models, using ~ 15 petaflops-days for pretraining (for a total of 830,000 TPUv4-hours over the study, see Appendix B.2 for details). We do not take into account memory use: typical use cases are compute-bound by the available GPU/TPU-hours. We note that the encoder-decoder ends up with twice the memory footprint.

Resources and implementation. We run all computation on Google Cloud TPUv4s, using T5X [Roberts et al., 2022], leveraging JAX [Bradbury et al., 2018] and Flax [Heek et al., 2020].

3.1 Architecture

We consider **causal decoder (CD)**, **encoder-decoder (ED)**, and **non-causal decoder (ND)** architectures. All models share the basic configuration outlined in Table 1. For fair comparison across architectures, we aim to approximately match pretraining compute budget; accordingly, our encoder-decoder models have twice as many layers as the decoder-only models. This results in encoder-decoder models with 11B parameters and decoder-only models with 4.8B parameters. We note that due to the cross-attention layers, encoder-decoder models are approximately $\sim 10\%$ more computationally expensive to run than the decoder-only models we consider.

3.2 Pretraining

We consider **full language modeling (FLM)**, **prefix language modeling (PLM)**, and **masked language modeling (MLM)** (specifically, the span corruption objective of Raffel et al. [2020]). The choice of language modeling objective depends on the architecture: the causal decoder uses either FLM or MLM, while the non-causal decoder and the encoder-decoder use either PLM or MLM.

All of our models are pretrained on 168 billion tokens of the C4 dataset from Raffel et al. [2020]. We use the Adafactor [Shazeer and Stern, 2018] optimizer with an inverse square root learning rate schedule, training on batches of 2,048 sequences of length 626 tokens (for a total of 131,072 training steps). Detailed pretraining hyperparameters can be found in Table 2: we based elements of our pretraining setup (such as Adafactor, GEGLU, and the use of an auxiliary Z loss $\mathcal{L}(Z) = 10^{-4} * \log^2(Z)$ to stabilize training [Chowdhery et al., 2022]) on the popular T5.1.1 recipe.

To operate with a fixed compute budget, we match the amount of tokens seen during pretraining (which corresponds to the total computational cost), not the number of tokens trained on (i.e. on which a loss is calculated). Full language modeling computes a loss on all the tokens it sees, whereas prefix language modeling cannot train on the tokens in its prefix: on average, it will train on half as many tokens as full language modeling. We consider these to be inherent trade-offs in efficiency between training objectives. We concatenated and sampled text from documents in such a way that there was virtually no padding during pretraining. More specifically to each objective:

- For full language modeling, the loss is computed for all 626 token in each sequence in parallel, making for the most efficient configuration (100% of tokens are trained on).
- For prefix language modeling, we select a random split point in [1, 626], which we use as the prefix length of one example and the suffix length for another, packing them together to avoid padding (using appropriately masked attention), and computing the loss only on the suffixes (50% of tokens on average). See Appendix C for implementation details on TPUs.
- For masked language modeling, 15% of input tokens are masked with an average span length of 3 (as used by Raffel et al. [2020]), such that there are approximately 512 input and 114 target tokens, with the loss computed only on the targets (18% of tokens on average).

Table 1: **Shared architecture for all models trained.** Encoder-decoder architectures are doubled in size to obtain a pretraining compute budget similar to the decoder-only architecture.

	MODELS ARCHITECTURE	
	Decoder-only	Encoder-decoder
Parameters	4.8B	11.0B
Vocabulary	32,128	
Positional embed.	T5 relative	
Embedding dim.	4,096	
Attention heads	64	
Feedforward dim.	10,240	
Activation	GEGLU [Shazeer, 2020]	
Layers	24	48
Tied embeddings	True	
Precision	`bfloat16`	

3.3 Multitask finetuning

Drawing from recent work demonstrating that multitask finetuning improves zero-shot performance, we also evaluate our models after **multitask finetuning (MT-F)**, following the procedure used for the T0 model of Sanh et al. [2021]. Our goal is to better disambiguate the influence of architecture and objective in this relatively nascent practice. For example, we note that T0 and FLAN are significantly different in the architecture and objective used (encoder-decoder with MLM and causal decoder with FLM, respectively). We hope our experiments can help lend insight into which of these design choices is more effective for enabling zero-shot generalization after multitask finetuning.

After pretraining, we create multitask versions of our models by finetuning on the T0 training dataset mixture from Sanh et al. [2021] (not T0+ or T0++) for 13 billion tokens. Our finetuning configurations follow those used for T0 (see Table 2 for details), and note that we found dropout to significantly impact zero-shot generalization (see Appendix E.3 for a comparison with and without dropout). We refer readers to Sanh et al. [2021] for further information about this multitask finetuning procedure.

One significant departure to note from the approach of Sanh et al. [2021] is that we do not perform language modeling adaptation first before multitask finetuning. Preliminary results (see Appendix E.1 in the Appendix) did not show any systematic improvement from performing language modeling adaptation, so we omitted this step. This is consistent with the finding from Lester et al. [2021] that language modeling adaptation is not necessary before prompt tuning for large models.

3.4 Evaluation

We use two zero-shot evaluation benchmarks to assess our models. First, we use the **same set of tasks, datasets, and prompts as was used to evaluate T0 (T0-Eval)** [Sanh et al., 2021], and second, the **EleutherAI LM evaluation harness (EAI-Eval)** [Gao et al., 2021]. The EAI prompts attempt to replicate the evaluation set of Brown et al. [2020]. The prompts of T0 were built to be "human understandable" and were originally used in conjunction with an encoder-decoder model. See Appendix D for a detailed list of tasks, and the overlap between T0-Eval, EAI-Eval, and T0-Train.

T0-Eval provides multiple prompts per task, whereas EAI-Eval provides only one prompt per task. Accordingly, for T0-Eval, we take the median accuracy over all prompts for each task and then average across all 11 datasets. For EAI-Eval we simply average the accuracy obtained on each of the 31 datasets. Note that because these are aggregated zero-shot benchmarks, variations of even a percent can hide significant differences on a single task.

All but one task in T0-Eval (StoryCloze) are also in EAI-Eval. Some of the datasets in EAI-Eval are also in the T0 training datasets: GLUE-MRPC, GLUE-QQP, and SciQ. We also did not check for contamination from C4, but given the fact that all models would have the opportunity to memorize the tasks leaked in C4, we believe it does not impact our evaluation. For additional discussion of the overlap between T0 and C4, we refer readers to the original T0 paper Sanh et al. [2021].

We perform evaluation of model checkpoints at 42B, 84B, and 168B tokens. We note that the random baselines are $\sim 33\%$ for EAI-Eval and $\sim 42\%$ for T0-Eval. The complete set of results across all checkpoints obtained through this study is made available in Appendix E.2.

Table 2: **Pretraining and multitask finetuning configurations for all models trained.** Pretraining lasts for 168 billion tokens, while multitask finetuning is done for 13 billion tokens.

	Pretraining	Multitask finetuning
Dataset	C4	T0-Train
Steps	131,072	10,000
Batch size in tokens	1,282,048	1,310,720
Optimizer	Adafactor(decay_rate=0.8)	
LR schedule	$\frac{1}{\sqrt{\max(n, 10^4)}}$	fixed, 0.001
Dropout	0.0	0.1
z loss	0.0001	
Precision	`bfloat16`	

4 Experiments

4.1 After unsupervised pretraining only

We are first interested in the architecture and objective achieving the best zero-shot performance after unsupervised pretraining only. For this, we only consider the full/prefix language modeling objectives since masked language modeling does not yield a model appropriate for zero-shot prompted evaluation on its own. This is validated with early checkpoints in Appendix E.1.

We present our main full/prefix language modeling pretraining results in Table 3. On both our evaluation benchmarks, the causal decoder architecture systematically outperforms the other architectures when using language modeling pretraining alone. The non-causal decoder remains within a percent of the causal decoder performance, but the encoder-decoder performance lags far behind. Finally, we note that the performances on T0-Eval are close to the random baseline, while performance differences on EAI-Eval are significant enough to make comparison across experiments.

Finding 1. **Causal decoder-only** models pretrained with a **full language modeling** objective achieve best zero-shot generalization when evaluated immediately after **unsupervised pretraining**, in line with current common practices for large language models.

4.2 After multitask finetuning

We now focus on the relatively new practice of multitask finetuning, where there has not yet been any systematic study of the influence of the architecture and training objective. Notably, the two main papers advocating this practice use completely different approaches: Sanh et al. [2021] finetunes an encoder-decoder model pretrained with span corruption, whereas Wei et al. [2021] finetunes a decoder-only pretrained with full language modeling. It is not immediately clear which approach is more natural: while decoder-only models trained with full language modeling are better at zero-shot generalization (as evidenced in Section 4.1), encoder-decoder models and masked language modeling pretraining have been shown to perform significantly better after finetuning [Raffel et al., 2020]. We therefore evaluate every architecture and objective combination after multitask finetuning.

Our results are outlined in Figure 4. The encoder-decoder pretrained with span corruption offers the best performance after multitask finetuning. Specifically, on EAI-Eval, the best performance is achieved by the encoder-decoder with MLM, and the non-causal decoder with MLM comes in a close second. However, the difference is more significant on T0-Eval, where the encoder-decoder with MLM pretraining outperforms other models by a large margin. Finally, encoder-decoder pretrained with PLM and causal decoder with MLM achieve significantly worse performance than other models. These results are consistent across all levels of pretraining (see early checkpoints in Appendix D).

Finding 2. **Encoder-decoder** models trained with **masked language modeling** achieve the best zero-shot performance after **multitask finetuning**. More broadly, approaches that perform well in the single-task finetuning setting perform well on multitask finetuning.

Table 3: **After full or prefix language modeling pretraining, the causal decoder (FLM) exhibits the best zero-shot generalization abilities, followed closely by the non-causal decoder (PLM).** Average accuracy on EAI-Eval and T0-Eval after pretraining for 168B tokens. MLM pretraining is not considered here, as the models produced are not suitable for direct use in a zero-shot setting. Note that performance on T0-Eval remains close to random baseline. **Best for each benchmark.**

	EAI-EVAL	T0-EVAL
Causal decoder	**44.2**	**42.4**
Non-causal decoder	43.5	41.8
Encoder-decoder	39.9	41.7
Random baseline	32.9	41.7

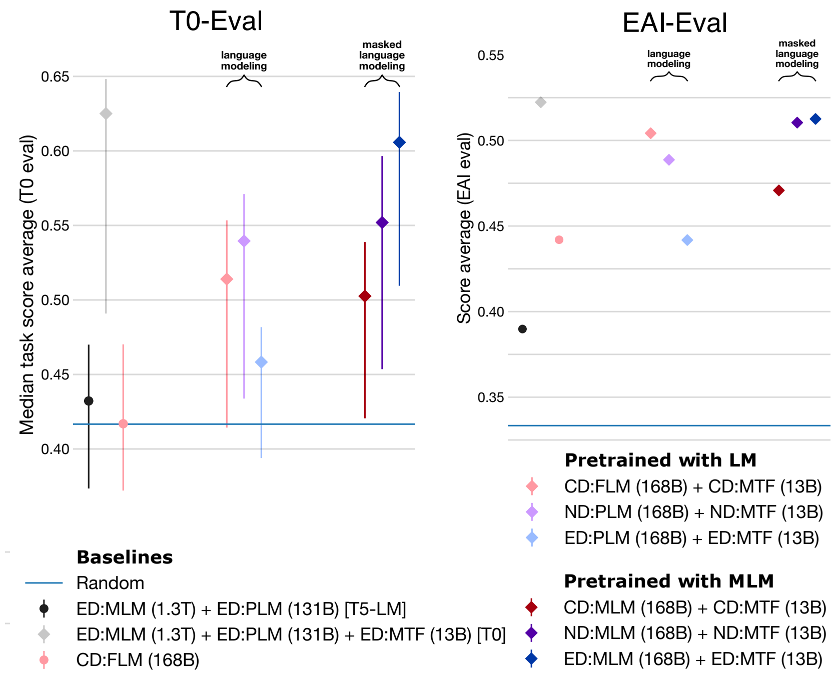

Figure 4: **When considering multitask finetuning, the encoder-decoder pretrained with MLM significantly outperforms other models, with the non-causal decoder pretrained with MLM a close second on EAI-Eval.** Detailed mean performance and spread compared to baselines on T0-Eval (left) and EAI-Eval (right) after multitask finetuning on T0 training set. The T5-LM and T0 results are taken from Sanh et al. [2021] and utilize the original T5.1.1-XXL model, which was pretrained for 7.6× the number of tokens used in this study. Other models are evaluated after pretraining with 168B tokens and multitask finetuning on T0 training set for 13B tokens.

4.3 Influence of the tasks and prompts used for zero-shot evaluation

Although the datasets considered in EAI-Eval and T0-Eval have significant overlap (10 out of 11 T0 tasks are in EAI-Eval), the prompts are always different between the two benchmarks. The EAI prompts for these datasets were taken from Brown et al. [2020], who hand-tuned them to maximize performance of the GPT-3 models. In contrast, the T0-Eval prompts were sourced through a community effort with prompt diversity and naturalness as primary goals without any regard for model performance [Sanh et al., 2021]. Consequently, on each task, the EAI prompt has higher performance than the average T0 prompt for all models and tends to be on par with the best T0 prompt. The difference is most pronounced for causal decoder language models without multitask finetuning, likely because this is the setting GPT-3 prompts were optimized for. This is reflected in the structure of the prompts, which tend not to explain the task to the reader like T0 evaluation prompts do. Instead, they attempt to reformulate it as close to language modeling as possible.

In addition to this base performance discrepancy, EAI-Eval has less discrepancy between encoder-decoder models and the rest, and better performance for autoregressive decoder models. We untangle the effect of the difference in prompts and the different task sets by separately comparing performance on tasks thare in T0-Eval and those that are not, while always using EAI-Eval prompts, as shown in Figure 5. The set of EAI-Eval tasks considered in T0-Eval seems to lend itself better to encoder-decoder models than the rest. On non-T0-Eval tasks, in contrast, causal decoder performance shoots up dramatically, although a lot of the difference is driven by LAMBADA [Paperno et al., 2016a], a language modeling task. Nevertheless, we note that when considering wide and varied task aggregates, our high-level findings are mostly consistent across evaluation settings–although specific tasks, such as LAMBADA, may indeed favor a specific architecture and objective combination.

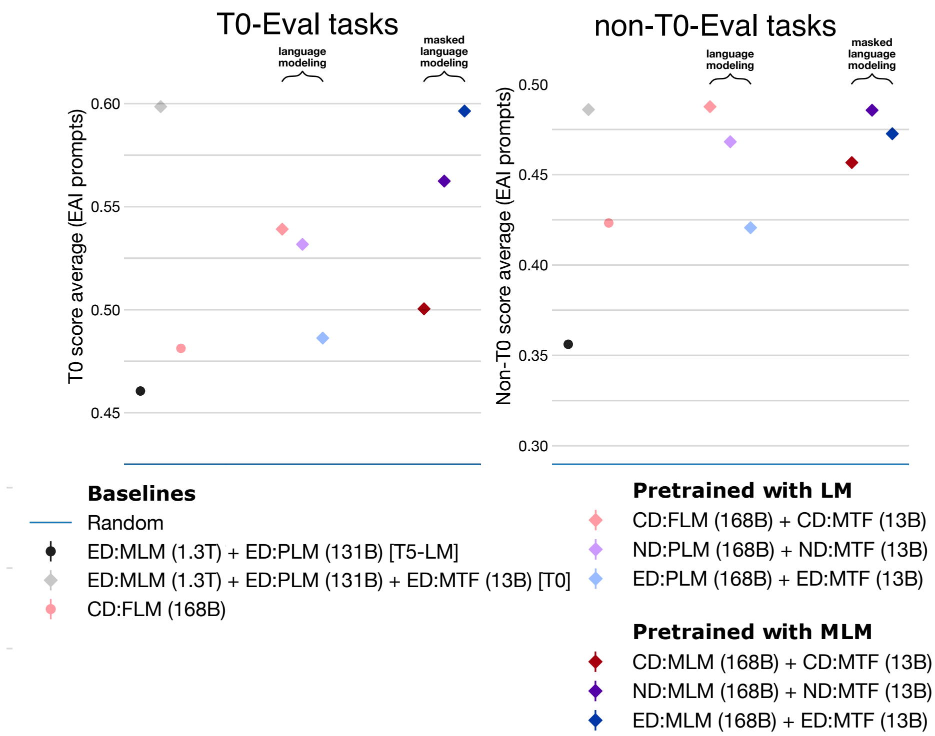

Figure 5: **Decoder-only models perform better on EAI-Eval because of specific tasks, not because of differences in prompts used.** Zero-shot performance on T0-Eval tasks (left) and non-T0-Eval tasks (right), both using EAI-Eval prompts to control for the influence of T0-Eval prompts.

5 Can models be adapted from one architecture/objective to another?

Our experimental study has led us to conclude the optimal architecture and objective choice for zero-shot performance depends on whether or not the model will ultimately undergo multitask finetuning: while a decoder-only model trained with full language modeling achieves the best zero-shot performance after unsupervised pretraining only, an encoder-decoder with masked language modeling is best once multitask finetuning is applied. This is inconvenient, as the multitask finetuned encoder-decoder model may not be suitable for many open-ended generative tasks that the decoder-only model excels at, while the decoder-only model will not be the best at many zero-shot tasks.

In this section, we attempt a compromise between the two options above. We study the practice of adaptation: extending pretraining with a different architecture and/or objective. Our end-goal is to efficiently obtain two distinct models: one that leverages multitask finetuning to maximize zero-shot performance, and another that can be used as a high-quality language model.

Language modeling adaptation (LM-A). First, we propose to pretrain a non-causal decoder model with an MLM objective and then further train the model as a causal decoder with a FLM objective (*language modeling adaptation*). This conversion is simple, as the parameters and overall architecture can be kept the same, and only the attention mask needs to be switched. We note that we also attempted this adaptation from the decoder portion of an encoder-decoder model, but it performed significantly worse than training from scratch, as discussed in Appendix E.4.

Validations losses are plotted in Figure 6, on the left. Starting from an MLM-pretrained non-causal decoder model speeds up convergence significantly compared to training a causal-decoder model with an FLM objective from scratch. To achieve a loss comparable to the one achieved after 168B tokens of FLM pretraining, language modeling adaptation requires only 105B additional tokens (a $1.6\times$ speed-up). This makes it possible to obtain both a high-quality zero-shot model and a good generative model, for only $1.6\times$ the cost of training a single model.

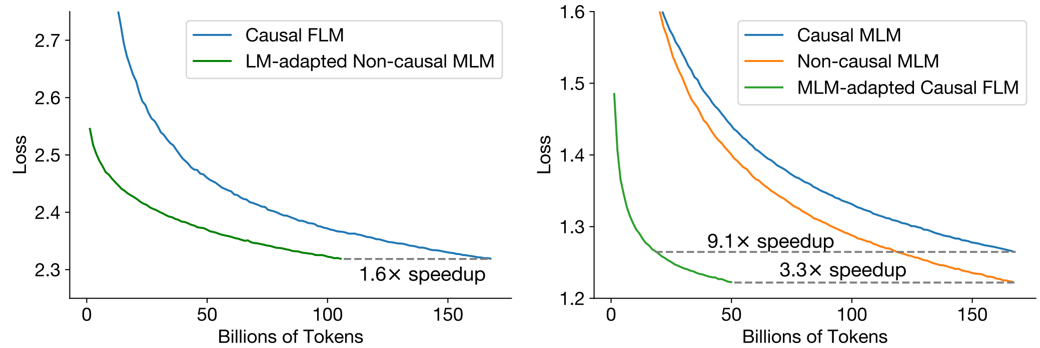

Figure 6: **Adaptation can efficiently convert non-causal decoder-only models pretrained with MLM into causal decoder-only models with FLM (left), and vice-versa (right).** Validation loss when adapting decoder-only models to different architecture/objectives following pretraining. Left: A causal decoder-only pretrained with FLM from scratch (Causal FLM) compared to a model being adapted with FLM *following* 168B pretraining tokens as a non-causal masked language model (LM-adapted Non-causal MLM). The adaptation requires 63% of the tokens (1.6× speedup) versus training from scratch. Right: Causal and non-causal decoder-only masked language models (Causal MLM, Non-causal MLM) trained from scratch compared to a model being adapted to a non-causal MLM *following* 168B pretraining tokens as a causal FLM (MLM-adapted Causal FLM). The adaptation requires 30% of the tokens (3.3× speedup) versus training the non-causal MLM from scratch.

Non-causal masked language modeling adaptation (NC-A). To investigate alternative avenues for adaptation, we now introduce *non-causal masked language modeling adaptation*: starting from a causal decoder model pretrained with FLM as the objective, we then continue training the model as a non-causal decoder using an MLM objective. This is essentially the reverse of the language modeling adaptation setup, and the conversion is as easily undertaken by switching the attention mask.

Validation losses are plotted in Figure 6, on the right. Convergence on the MLM pretraining objective is significantly accelerated: by a factor of 3.3× compared to training a non-causal decoder from scratch, and up to a factor 9.1× compared to training a causal decoder from scratch (both with a masked language modeling objective). This is a significant improvement over even the previously considered language modeling adaptation, enabling one to obtain both a zero-shot model and an excellent generative model for only 1.3× the cost of training a single model.

Finally, we confirm that the improvement in validation loss also transfer to an improvement in zero-shot generalization. We evaluate the non-causal MLM adapted model, and check that it is better than the original causal decoder model pretrained with full language modeling, and control for the total number of training tokens. Specifically, we evaluate zero-shot performance after multitask finetuning in three settings: first, a causal decoder model pretrained with FLM for 219 billion tokens before being multitask finetuned; second, a causal decoder model pretrained with FLM for 219 billion tokens and then multitask finetuned as a non-causal decoder model; and, third, a causal decoder model first trained with FLM for 168 billion tokens, then MLM-adapted as an non-causal model for 51 billion tokens, and finally multitask finetuned. All three variants are multitask finetuned for 13 billion tokens. Results are presented Figure 7. We find that the MLM-adapted model performs best by a significant margin and outperforms every other model we considered on EAI-Eval. Furthermore, the measured zero-shot generalization is in line with the MLM-pretrained non-causal decoder reported in Figure 4, though it still lags behind the MLM-pretrained encoder-decoder, despite the adapted models having seen 51 billion additional tokens. Finally, we note that performing non-causal multitask finetuning of the causal model produces no meaningful change in performance.

Finding 3. Decoder-only models can be efficiently adapted from one architecture/objective prior to the other. Specifically, to obtain both a generative and a multitask model with the smallest total compute budget possible, we recommend starting with a **causal decoder-only** model, pretraining it with a **full language modeling** objective, and then using **non-causal masked language modeling adaptation** before taking it through **multitask finetuning**.

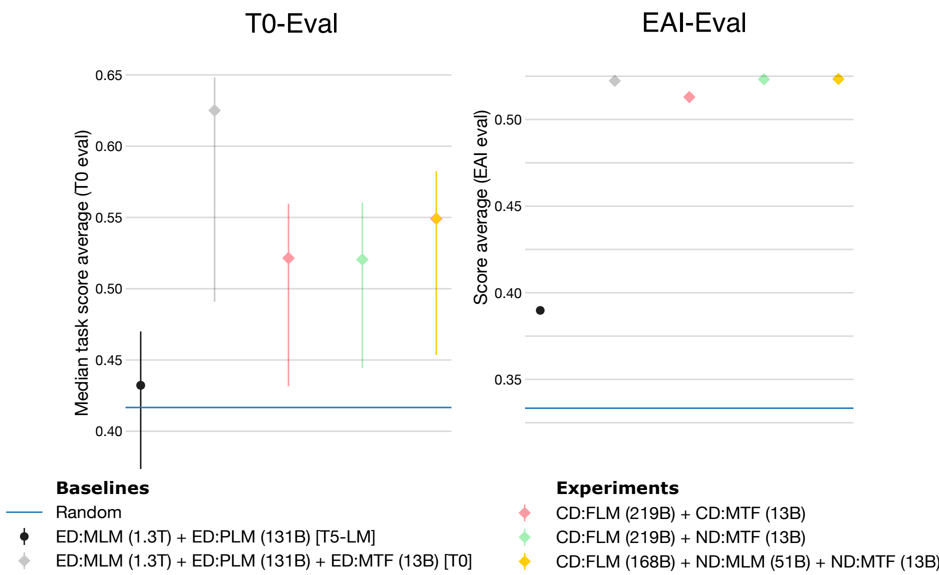

Figure 7: **Applying non-causal MLM adaptation to a causal decoder-only FLM before multitask finetuning improves zero-shot performance, even when controlling for additional LM pretraining for the same number of tokens.** Zero-shot generalization on T0-Eval (left) and EAI-Eval (right), for the T5-LM and T0 baselines (grey), and for models from our study. Converting the model into a non-causal decoder for multitask finetuning only does not improve performance on T0-Eval. Results after adaptation are in line with non-causal decoder-only pretrained with MLM in Figure 4.

6 Conclusion

In this paper, we systematically studied the effects of pretraining objective and architecture choices on the zero-shot generalization abilities of large language models. Specifically, we compared language modeling and masked language modeling objectives applied to causal/non-causal decoder-only and encoder-decoder architectures. We also evaluated zero-shot performance with and without multitask finetuning. Notably, we found that the best objective and architecture is the *opposite* in these two settings: a causal decoder-only pretrained with full language modeling performs best if evaluated immediately after pretraining, whereas when adding a multitask finetuning step, an encoder-decoder pretrained with masked language modeling performs best. We therefore evaluate the practice of adaptation, to convert models across architectures and objectives. We found a simple efficient compromise, where a causal decoder-only model pretrained with full language modeling underwent additional masked language model training as a non-causal decoder-only model, yielding significant speedup in convergence over starting from scratch. This enables practitioners to get both an excellent generative model and a model that delivers good performance after multitask finetuning. Our results provide significant new insights into the design of LLMs. In the future, we are interested in work investigating architectures and objectives that perform well regardless of whether multitask finetuning is performed. To facilitate future work, we release all models, code, and data used in our study.

Acknowledgements

BigScience. This work was pursued as part of the Big Science research workshop, a one-year long initiative on large multilingual models and datasets. Specifically, this work was conducted by a task force within the architecture & scaling group, seeking to establish the optimal architecture and pretraining objective for the final 176B parameter model produced by BigScience. We would also like to thank Stella Biderman for valuable comments.

Compute. We thank the TPU Research Cloud team for providing us with generous access to TPUv4. We thank the TPUv4 Alpha team for providing technical support for this work, and notably James Bradbury. All pretraining, adaptations, finetuning, and evaluations featured in this paper used TPUv4.

This work was granted access to the HPC resources of Institut du développement et des ressources en informatique scientifique (IDRIS) du Centre national de la recherche scientifique (CNRS) under the allocation 2021-A0101012475 made by Grand équipement national de calcul intensif (GENCI). Specifically, early experiments on non-causal decoder models, which sparked this exhaustive study, were performed on the Jean Zay supercomputer of IDRIS.

Author-specific funding.

Daniel Hesslow has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 860360.

References

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. URL `https://arxiv.org/abs/2005.14165`.

Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`, May 2021.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. *arXiv preprint arXiv:2112.06905*, 2021. URL `https://arxiv.org/abs/2112.06905`.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, et al. Few-shot learning with multilingual language models. *arXiv preprint arXiv:2112.10668*, 2021. URL `https://arxiv.org/abs/2112.10668`.

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

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022. URL `https://arxiv.org/abs/2203.15556`.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022. URL https://arxiv.org/abs/2204.02311.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021. URL https://arxiv.org/abs/2110.08207.

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

Hanwei Xu, Yujun Chen, Yulun Du, Nan Shao, Yanggang Wang, Haiyu Li, and Zhilin Yang. Zeroprompt: Scaling prompt-based pretraining to 1,000 tasks improves zero-shot generalization. *arXiv preprint arXiv:2201.06910*, 2022. URL https://arxiv.org/abs/2201.06910.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*, 2018.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Blog*, 2018.

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3159–3166, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 13063–13075, 2019.

Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. *Advances in neural information processing systems*, 28:3079–3087, 2015.

Wilson L Taylor. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30 (4):415–433, 1953.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. URL https://arxiv.org/abs/1810.04805.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, abs/1910.10683, 2019. URL http://arxiv.org/abs/1910.10683.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*, 2019. URL https://arxiv.org/abs/1901.07291.

Elena Voita, Rico Sennrich, and Ivan Titov. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. *arXiv preprint arXiv:1909.01380*, 2019. URL https://arxiv.org/abs/1909.01380.

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

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021. URL https://arxiv.org/abs/2104.08691.

Sharan Narang, Hyung Won Chung, Yi Tay, William Fedus, Thibault Févry, Michael Matena, Karishma Malkan, Noah Fiedel, Noam Shazeer, Zhenzhong Lan, Yanqi Zhou, Wei Li, Nan Ding, Jake Marcus, Adam Roberts, and Colin Raffel. Do transformer modifications transfer across implementations and applications? *arXiv preprint arXiv:2102.11972*, abs/2102.11972, 2021. URL https://arxiv.org/abs/2102.11972.

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. A framework for few-shot language model evaluation, September 2021. URL https://doi.org/10.5281/zenodo.5371628.

Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016. URL https://arxiv.org/abs/1601.06733.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. URL https://arxiv.org/abs/1607.06450.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019. URL https://arxiv.org/abs/1910.13461.

Wei Zeng, Xiaozhe Ren, Teng Su, Hui Wang, Yi Liao, Zhiwei Wang, Xin Jiang, ZhenZhang Yang, Kaisheng Wang, Xiaoda Zhang, et al. Pangu-alpha: Large-scale autoregressive pretrained chinese language models with auto-parallel computation. *arXiv preprint arXiv:2104.12369*, 2021. URL https://arxiv.org/abs/2104.12369.

Boseop Kim, HyoungSeok Kim, Sang-Woo Lee, Gichang Lee, Donghyun Kwak, Dong Hyeon Jeon, Sunghyun Park, Sungju Kim, Seonhoon Kim, Dongpil Seo, et al. What changes can large-scale language models bring? intensive study on hyperclova: Billions-scale korean generative pretrained transformers. *arXiv preprint arXiv:2109.04650*, 2021. URL https://arxiv.org/abs/2109.04650.

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

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

Shaohua Wu, Xudong Zhao, Tong Yu, Rongguo Zhang, Chong Shen, Hongli Liu, Feng Li, Hong Zhu, Jiangang Luo, Liang Xu, et al. Yuan 1.0: Large-scale pre-trained language model in zero-shot and few-shot learning. *arXiv preprint arXiv:2110.04725*, 2021. URL https://arxiv.org/abs/2110.04725.

Alexandre Tamborrino, Nicola Pellicano, Baptiste Pannier, Pascal Voitot, and Louise Naudin. Pre-training is (almost) all you need: An application to commonsense reasoning. *arXiv preprint arXiv:2004.14074*, 2020. URL https://arxiv.org/abs/2004.14074.

Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures. In *Proceedings of the Workshop on Challenges in the Management of Large Corpora*, 2019.

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. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020. URL https://arxiv.org/abs/2101.00027.

Teven Le Scao, Thomas Wang, Daniel Hesslow, Lucile Saulnier, Stas Bekman, M Saiful Bari, Stella Biderman, Hady Elsahar, Jason Phang, Ofir Press, Colin Raffel, Victor Sanh, Sheng Shen, Lintang Sutawika, Jaesung Tae, Zheng Xin Yong, Julien Launay, and Iz Beltagy. What language model to train if you have one million GPU hours? In *Challenges & Perspectives in Creating Large Language Models*, 2022. URL https://openreview.net/forum?id=rI7BL3fHIZq.

Adam Roberts, Hyung Won Chung, Anselm Levskaya, Gaurav Mishra, James Bradbury, Daniel Andor, Sharan Narang, Brian Lester, Colin Gaffney, Afroz Mohiuddin, Curtis Hawthorne, Aitor Lewkowycz, Alex Salcianu, Marc van Zee, Jacob Austin, Sebastian Goodman, Livio Baldini Soares, Haitang Hu, Sasha Tsvyashchenko, Aakanksha Chowdhery, Jasmijn Bastings, Jannis Bulian, Xavier Garcia, Jianmo Ni, Andrew Chen, Kathleen Kenealy, Jonathan H. Clark, Stephan Lee, Dan Garrette, James Lee-Thorp, Colin Raffel, Noam Shazeer, Marvin Ritter, Maarten Bosma, Alexandre Passos, Jeremy Maitin-Shepard, Noah Fiedel, Mark Omernick, Brennan Saeta, Ryan Sepassi, Alexander Spiridonov, Joshua Newlan, and Andrea Gesmundo. Scaling up models and data with `t5x` and `seqio`. *arXiv preprint arXiv:2203.17189*, 2022. URL https://arxiv.org/abs/2203.17189.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2020. URL http://github.com/google/flax.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. URL https://arxiv.org/abs/2002.05202.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR, 2018.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, abs/1606.06031, 2016a. URL http://arxiv.org/abs/1606.06031.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, et al. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*, 2019. URL https://arxiv.org/abs/1908.09203.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. URL https://arxiv.org/abs/2108.07258.

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Gregory Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Martin Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. GPT-neox-20b: An open-source autoregressive language model. In *Challenges & Perspectives in Creating Large Language Models*, 2022. URL https://openreview.net/forum?id=HL7IhzS8W5.

Sara Hooker. Moving beyond "algorithmic bias is a data problem". *Patterns*, 2(4):100241, 2021.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021. URL https://arxiv.org/abs/2104.10350.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, abs/1803.05457, 2018. URL http://arxiv.org/abs/1803.05457.

William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. First quora dataset release: Question pairs, 2017. URL https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany, August 2016b. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*, abs/2007.08124, 2020. URL https://arxiv.org/abs/2007.08124.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1245. URL https://aclanthology.org/N19-1245.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Stéphane Aroca-Ouellette, Cory Paik, Alessandro Roncone, and Katharina Kann. PROST: Physical reasoning about objects through space and time. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4597–4608, Online, August 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.findings-acl.404.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016. URL https://arxiv.org/abs/1606.05250.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*, 2017. URL https://arxiv.org/abs/1704.04683.

Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, 2017.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*, 2019.

Andrew Gordon, Zornitsa Kozareva, and Melissa Roemmele. SemEval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 394–398, Montréal, Canada, 7-8 June 2012. Association for Computational Linguistics. URL https://aclanthology.org/S12-1052.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface:a challenge set for reading comprehension over multiple sentences. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer, 2005.

Mohammad Taher Pilehvar and os'e Camacho-Collados. Wic: 10, 000 example pairs for evaluating context-sensitive representations. *arXiv preprint arXiv:1808.09121*, abs/1808.09121, 2018. URL http://arxiv.org/abs/1808.09121.

Hector Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2012.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544, 2013.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019. URL https://arxiv.org/abs/1907.10641.

A Contributions

Thomas Wang wrote code, ran experiments, performed evaluation, generated plots, and helped with paper writing. Adam Roberts led the creation of the codebase used in this project, proposed some experiments, ran all of the final experiments, generated plots, and helped with paper writing. Daniel Hesslow made it possible to evaluate models with the EleutherAI harness, created diagrams, and helped with paper writing. Teven Le Scao ran evaluations and plotted results. Hyung Won Chung implemented infrastructure code for different architectural and objective variants. Iz Beltagy co-chaired the BigScience Architecture & Scaling Group and helped with paper editing. Julien Launay co-chaired the BigScience Architecture & Scaling Group and had the largest role in paper writing. Colin Raffel proposed the project, the experiments, and the adaptation methods and wrote portions of the paper.

B Broader impacts

B.1 Societal impact

The risks and societal challenges raised by large language models have been discussed extensively in the literature [Solaiman et al., 2019, Bommasani et al., 2021]. Our research is strictly oriented on benchmarking modeling aspects, and thus does not introduce any novel challenge beside those already identified. Notably, many similarly capable models have already been released publicly in the past [Raffel et al., 2020, Wang and Komatsuzaki, 2021, Sanh et al., 2021, Black et al., 2022].

In the spirit of reproducibility and openness, we release all artefacts produced during this study: the configs necessary to reproduce our results from scratch, checkpoints of all of the models trained, and detailed evaluation results. These artefacts are intended for research only: we did not evaluate the potential biases of the models trained, and cannot guarantee they won't produce harmful content. Accordingly, **these models should not be used in production or exposed to the public.**

We also highlight that algorithmic choices can introduce biases on their own [Hooker, 2021]: one limitation of our study is that we did not explore whether specific architectures and objectives had an impact on the toxicity and biases of a given model. However, the public availability of all the models trained for this study enables researchers to conduct such a follow-up study at minimal compute cost.

B.2 Environmental impact

Across all experiments undertaken over the course of this study (including unreported preliminary and failed experiments), we performed training for 1,854 hours on 64 chips (TPUv4-128) and 1,395 hours on 512 TPUv4 chips (TPUv4-1024) for a total of 832,896 chip-hours. Recently, Chowdhery et al. [2022] presented the results of training a 540 billion parameter language model on TPUv4 chips in the same datacenter where we ran our experiments. Their model was trained for 1,200 hours on 6,144 TPUv4 chips and 336 hours on 3,072 TPUv4 chips, for a total of 8,404,992 chip-hours. Chowdhery et al. [2022] estimates the carbon emissions of their model training to be 240.5 tCO2e based on the net tCO2e per MWh of the datacenter during training and the energy usage of TPUv4 chips. We therefore estimate our carbon emissions to be approximately 23.8 tCO2e, which is approximately half of what Patterson et al. [2021] report for the original T5 model training (46.7 tCO2).

C Implementation: prefix language modeling for encoder-decoder on TPU

Due to constant size constraints, for encoder-decoder with prefix language modeling, we have to concatenate two examples of 626 tokens into one. We randomly sample an index i between 0 and 626, and use i and $626 - i$ as prefix indices in the two examples. We use masking to keep them independent throughout training. The encoder-decoder thus has a 1,252 sequence length, and we train it with a batch size of 1,024 sequences instead of 2,048 to keep the number of tokens constant.

D Evaluation: benchmarks composition and baselines

We detail the split across EAI-Eval and T0-Eval in Table 4, and provide random baselines in Table 5.

Table 4: **Tasks used for zero-shot evaluation within EAI-Eval and T0-Eval, as well as tasks included in the T0 training set for multitask finetuning.** Note that T0-Eval/Training tasks include multiple prompts, while EAI-Eval tasks have a single prompt. Although tasks are shared between the two, there are no shared prompts between EAI-Eval and T0-Eval.

TASK		TYPE	DATASET		
			T0 Training	EAI-Eval	T0-Eval
ANLI		Natural Language Inference		✓	✓
ARC [Clark et al., 2018]	Challenge	Closed-Book Question Answering		✓	
	Easy	Closed-Book Question Answering		✓	
GLUE	MRPC [Dolan and Brockett, 2005]	Paraphrase Identification	✓	✓	
	QQP [Iyer et al., 2017]	Paraphrase Identification	✓	✓	
HEAD-QA		Multiple-Choice Question Answering		✓	
HellaSwag [Zellers et al., 2019]		Sentence Completion		✓	✓
LAMBADA [Paperno et al., 2016b]		Sentence Completion		✓	
LogiQA [Liu et al., 2020]		Multiple-Choice Question Answering		✓	
MathQA [Amini et al., 2019]		Multiple-Choice Question Answering		✓	
OpenBookQA [Mihaylov et al., 2018]		Multiple-Choice Question Answering		✓	
PIQA [Bisk et al., 2020]		Multiple-Choice Question Answering		✓	
PROST [Aroca-Ouellette et al., 2021]		Multiple-Choice Question Answering		✓	
PudMedQA [Jin et al., 2019]		Multiple-Choice Question Answering		✓	
QNLI [Rajpurkar et al., 2016, Wang et al., 2019]		Sentence Completion		✓	
Race Lai et al. [2017]		Multiple-Choice Question Answering		✓	
SciQ [Welbl et al., 2017]		Multiple-Choice Question Answering	✓	✓	
SST [Socher et al., 2013] [Socher et al., 2013]		Sentiment		✓	
StoryCloze		Sentence Completion			✓
SuperGLUE	Boolq [Clark et al., 2019]	Multiple-Choice Question Answering		✓	
	CB	Natural Language Inference		✓	✓
	COPA [Gordon et al., 2012]	Sentence Completion		✓	✓
	MultiRC [Khashabi et al., 2018]	Multiple-Choice Question Answering		✓	
	RTE [Dagan et al., 2005]	Natural Language Inference		✓	✓
	WIC [Pilehvar and os'e Camacho-Collados, 2018]	Word Sense Disambiguation		✓	✓
	WSC [Levesque et al., 2012]	Word Sense Disambiguation		✓	✓
TriviaQA [Joshi et al., 2017]		Closed-Book Question Answering		✓	
WebQuestions [Berant et al., 2013]		Closed-Book Question Answering		✓	
Winogrande [Sakaguchi et al., 2019]		Coreference resolution		✓	✓
WNLI [Sakaguchi et al., 2019]		Natural language inference		✓	

Table 5: **Random baselines for all tasks considered across EAI harness and T0 Eval.** These baselines were obtained from the papers introducing these tasks.

TASK		RANDOM BASELINE
ANLI		33.3
ARC	Challenge	25.0
	Easy	25.0
GLUE	MRPC	50.0
	QQP	50.0
HEAD-QA		25.0
HellaSwag		25.0
LAMBADA		0.0
LogiQA		25.0
MathQA		20.1
OpenBookQA		25.0
PIQA		50.0
PROST		25.0
PudMedQA		33.3
QNLI		50.0
Race		25.0
SciQ		25.0
SST		50.0
StoryCloze		50.0
SuperGLUE	Boolq	50.0
	CB	50.0
	COPA	50.0
	MultiRC	5.8
	RTE	50.0
	WIC	50.0
	WSC	50.0
TriviaQA		0.0
WebQuestions		0.0
Winogrande		50.0
WNLI		50.0
EAI-EVAL		33.3
T0-EVAL		41.7

E Additional results

E.1 Preliminary results and evolution throughout pre-training

Leveraging early pretraining results at 42B and 84B tokens, we motivate in this section two special design decisions in our study:

- **Not considering span corruption for evaluation after pretraining only.** In Table 3, we only report zero-shot generalization results immediately after pretraining for the full and prefix language modeling objectives. We choose not to report results when using a masked language modeling objective, as Table 6 demonstrates that after 84B tokens of pretraining, models pretrained with this objective still achieve close to random performance, and severely underperform models pretrained with prefix or full language modeling.
- **Not systematically performing LM adaptation before multitask finetuning.** Sanh et al. [2021] originally perform LM adaptation before multitask finetuning. As outlined in Table 7, using early models pretrained for 42B tokens, we found this practice did not consistently improve zero-shot generalization, and in fact worsened it in most cases. Accordingly, results in Figure 4 do not use LM adaptation before multitask finetuning. This is in line with the findings of Lester et al. [2021] that larger models (of the same scale that we are considering in our study) do not benefit from performing LM adaptation before prompt tuning.

Table 6: **Models pretrained with masked language modeling achieve performance close to the random 33.3% baseline on EAI-Eval, significantly underperforming full and prefix language modeling.** Average accuracy on EAI-Eval after pretraining for 84B tokens. This observations leads us to not consider masked language modeling for evaluations after pretraining only.

	EAI-EVAL
FLM/PLM	
Causal decoder	**42.4**
Non-causal decoder	42.2
Encoder-decoder	39.6
MLM	
Causal decoder	37.8
Non-causal decoder	37.7
Encoder-decoder	34.6

Table 7: **Performing LM adaptation before multitask finetuning does not improve results, and in fact hinders performance in most cases.** Average accuracy on EAI-Eval and T0-Eval for different adaptation strategies after 42B tokens of masked language modeling pretraining. LM adaptation alone is insufficient, and most performance gains come from MT finetuning. Accordingly, we diverge from the setup of Sanh et al. [2021], and forego systematic LM adaptation before multitask finetuning.

	EAI-EVAL	T0-EVAL
LM ADAPTATION		
Causal decoder	38.6	43.9
Non-causal decoder	39.5	40.8
Encoder-decoder	38.6	39.1
MT FINETUNING		
Causal decoder	43.3	45.8
Non-causal decoder	**45.9**	48.9
Encoder-decoder	45.4	**53.7**
LM-A+MT FINETUNING		
Causal decoder	43.9	46.7
Non-causal decoder	45.0	48.0
Encoder-decoder	45.7	52.6

E.2 Complete Results

We report results for all intermediary checkpoints produced in Table 8, and specifically for all multitask finetuned checkpoints on T0-Eval in Figure 8.

Table 8: **Average accuracy on EAI-EVal and T0 Eval for all experiments.** Experiments are represented as a combination of *architecture*:*objective* (*tokens*) training stages, where *architecture* is one of causal decoder-only (CD), non-causal decoder-only (ND), or encoder-decoder (ED), and *objective* is one of full language modeling (FLM), prefix language modeling (PLM), masked language modeling (MLM), or multitask finetuning (MTF).

Pretraining	Training Stage Adaptation	Finetuning	Total Tokens	EAI-Eval	T0-Eval
CD:MLM (38B)	CD:FLM (4B)		42B	38.6	43.9
ND:MLM (38B)	ND:PLM (4B)		42B	39.5	40.8
ED:MLM (38B)	ED:MLM (4B)		42B	38.6	39.1
CD:MLM (42B)		CD:MTF (13B)	55B	43.3	45.8
ND:MLM (42B)		ND:MTF (13B)	55B	45.9	48.9
ED:MLM (42B)		ED:MTF (13B)	55B	45.4	53.7
CD:MLM (38B)	CD:FLM (4B)	CD:MTF (13B)	55B	43.9	46.7
ND:MLM (38B)	ND:PLM (4B)	ND:MTF (13B)	55B	45.0	48.0
ED:MLM (38B)	ED:PLM (4B)	ED:MTF (13B)	55B	45.7	52.6
CD:FLM (84B)			84B	42.4	-
ND:PLM (84B)			84B	42.2	-
ED:PLM (84B)			84B	39.6	-
CD:MLM (84B)			84B	37.8	-
ND:MLM (84B)			84B	37.7	-
ED:MLM (84B)			84B	34.6	-
CD:FLM (84B)		CD:MTF (13B)	97B	49.0	49.9
ND:PLM (84B)		ND:MTF (13B)	97B	46.3	50.0
ED:PLM (84B)		ED:MTF (13B)	97B	43.2	46.5
CD:MLM (84B)		CD:MTF (13B)	97B	45.8	48.2
ND:MLM (84B)		ND:MTF (13B)	97B	49.0	52.6
ED:MLM (84B)		ED:MTF (13B)	97B	49.0	56.5
CD:FLM (168B)			168B	44.2	42.4
ND:PLM (168B)			168B	43.5	41.8
ED:PLM (168B)			168B	39.9	41.7
CD:FLM (168B)		CD:MTF (13B)	181B	50.4	51.4
ND:PLM (168B)		ND:MTF (13B)	181B	48.9	54.0
ED:PLM (168B)		ED:MTF (13B)	181B	44.2	45.8
CD:MLM (168B)		CD:MTF (13B)	181B	47.1	50.3
ND:MLM (168B)		ND:MTF (13B)	181B	51.0	55.2
ED:MLM (168B)		ED:MTF (13B)	181B	51.3	60.6
CD:FLM (168B)	CD:FLM (51B)	CD:MTF (13B)	232B	51.3	52.1
CD:FLM (168B)	CD:FLM (51B)	ND:MTF (13B)	232B	52.3	52.0
CD:FLM (168B)	ND:MLM (51B)	ND:MTF (13B)	232B	52.3	54.9
T5-LM Baseline [Lester et al., 2021]					
ED:MLM (1.28T)	ED:PLM (131B)		1.41T	39.0	43.2
T0 Baseline [Sanh et al., 2021]					
ED:MLM (1.28T)	ED:PLM (131B)	ED:MTF (13B)	1.43T	52.2	62.5
Random Baseline			-	32.9	41.7

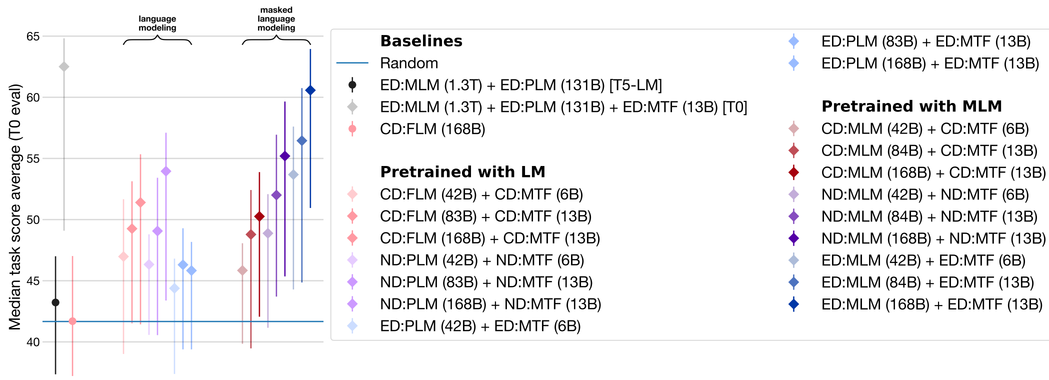

Figure 8: **Performance on T0-Eval after multitask finetuning for increasing amounts of pretraining (measured in tokens).** Our best model, an encoder-decoder trained with masked language modeling, is already above the final performance of the other configurations with only a quarter of the pretraining tokens. Note that the ordering does not change significantly throughout pretraining.

E.3 Impact of dropout on multitask finetuning

We also performed multitask finetuning without using dropout, with results in Figure 9. We find that using dropout as originally suggested by Sanh et al. [2021] significantly boosts zero-shot generalization. Results are consistent across architectures and pretraining objectives.

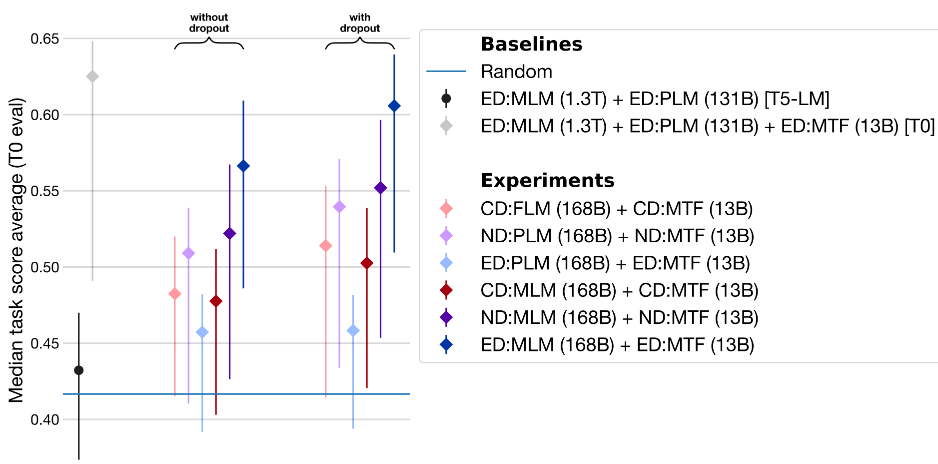

Figure 9: **Using dropout during multitask finetuning improves zero-shot generalization.** Performance on T0-Eval with and without dropout. The impact of dropout is proportionally similar across architecture and objectives, not benefitting any specific configuration more.

E.4 Adaptation from an encoder-decoder

When studying adaptation and the conversion from one architecture to another, we also considered converting to and from encoder-decoder models. Conversion across causal and non-causal decoder-only models is straightforward, simply by switching the attention mask; for encoder-decoder, parameters have to be either pruned or added for both the entire encoder, and for the cross-attention in the decoder. Results from one of our attempt to convert an encoder-decoder into a causal decoder are reported in Figure 10. While converting across causal/non-causal decoder provides an improvement over training from scratch, this is not the case here.

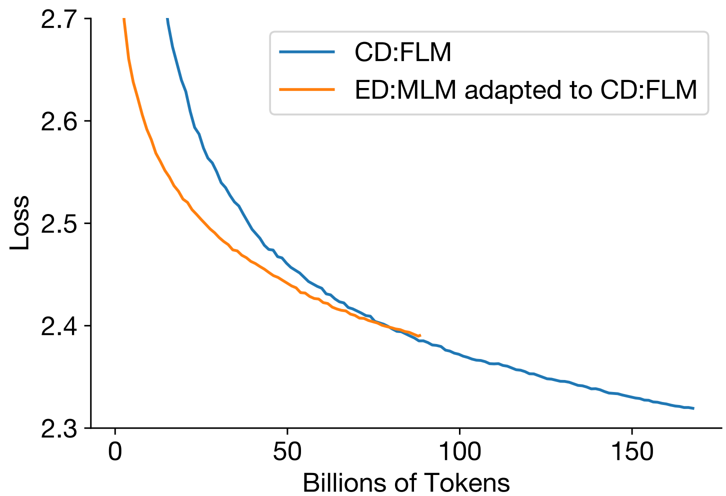

Figure 10: **Converting an encoder-decoder pretrained with MLM to a causal decoder-only using FLM leads to worse performance compared to training from scratch.** Validation loss when adapting an encoder-decoder pretrained with MLM to a causal decoder-only using FLM. We adapted a pretrained (for 168B tokens) encoder-decoder model to decoder-only by feeding an empty input into the encoder and causally training with a FLM objective on the decoder. We stopped this adaptation once it was clear the performance would not match that of a causal FLM trained from scratch, in contrast with the other adaptations we studied.