LLaMA: Open and Efficient Foundation Language Models

Hugo Touvron[*], **Thibaut Lavril**[*], **Gautier Izacard**[*], **Xavier Martinet**
Marie-Anne Lachaux, Timothee Lacroix, Baptiste Rozière, Naman Goyal
Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin
Edouard Grave[*], **Guillaume Lample**[*]

Meta AI

Abstract

We introduce LLaMA, a collection of foundation language models ranging from 7B to 65B parameters. We train our models on trillions of tokens, and show that it is possible to train state-of-the-art models using publicly available datasets exclusively, without resorting to proprietary and inaccessible datasets. In particular, LLaMA-13B outperforms GPT-3 (175B) on most benchmarks, and LLaMA-65B is competitive with the best models, Chinchilla-70B and PaLM-540B. We release all our models to the research community[1].

1 Introduction

Large Languages Models (LLMs) trained on massive corpora of texts have shown their ability to perform new tasks from textual instructions or from a few examples (Brown et al., 2020). These few-shot properties first appeared when scaling models to a sufficient size (Kaplan et al., 2020), resulting in a line of work that focuses on further scaling these models (Chowdhery et al., 2022; Rae et al., 2021). These efforts are based on the assumption that more parameters will lead to better performance. However, recent work from Hoffmann et al. (2022) shows that, for a given compute budget, the best performances are not achieved by the largest models, but by smaller models trained on more data.

The objective of the scaling laws from Hoffmann et al. (2022) is to determine how to best scale the dataset and model sizes for a particular *training* compute budget. However, this objective disregards the *inference* budget, which becomes critical when serving a language model at scale. In this context, given a target level of performance, the preferred model is not the fastest to train but the fastest at inference, and although it may be cheaper to train a large model to reach a certain level of performance, a smaller one trained longer will ultimately be cheaper at inference. For instance, although Hoffmann et al. (2022) recommends training a 10B model on 200B tokens, we find that the performance of a 7B model continues to improve even after 1T tokens.

The focus of this work is to train a series of language models that achieve the best possible performance at various inference budgets, by training on more tokens than what is typically used. The resulting models, called *LLaMA*, ranges from 7B to 65B parameters with competitive performance compared to the best existing LLMs. For instance, LLaMA-13B outperforms GPT-3 on most benchmarks, despite being 10× smaller. We believe that this model will help democratize the access and study of LLMs, since it can be run on a single GPU. At the higher-end of the scale, our 65B-parameter model is also competitive with the best large language models such as Chinchilla or PaLM-540B.

Unlike Chinchilla, PaLM, or GPT-3, we only use publicly available data, making our work compatible with open-sourcing, while most existing models rely on data which is either not publicly available or undocumented (e.g. "Books – 2TB" or "Social media conversations"). There exist some exceptions, notably OPT (Zhang et al., 2022), GPT-NeoX (Black et al., 2022), BLOOM (Scao et al., 2022) and GLM (Zeng et al., 2022), but none that are competitive with PaLM-62B or Chinchilla.

In the rest of this paper, we present an overview of the modifications we made to the transformer architecture (Vaswani et al., 2017), as well as our training method. We then report the performance of our models and compare with others LLMs on a set of standard benchmarks. Finally, we expose some of the biases and toxicity encoded in our models, using some of the most recent benchmarks from the responsible AI community.

[*] Equal contribution. Correspondence: {htouvron, thibautlav,gizacard,egrave,glample}@meta.com

[1] https://github.com/facebookresearch/llama

2 Approach

Our training approach is similar to the methods described in previous work (Brown et al., 2020; Chowdhery et al., 2022), and is inspired by the Chinchilla scaling laws (Hoffmann et al., 2022). We train large transformers on a large quantity of textual data using a standard optimizer.

2.1 Pre-training Data

Our training dataset is a mixture of several sources, reported in Table 1, that cover a diverse set of domains. For the most part, we reuse data sources that have been leveraged to train other LLMs, with the restriction of only using data that is publicly available, and compatible with open sourcing. This leads to the following mixture of data and the percentage they represent in the training set:

English CommonCrawl [67%]. We preprocess five CommonCrawl dumps, ranging from 2017 to 2020, with the CCNet pipeline (Wenzek et al., 2020). This process deduplicates the data at the line level, performs language identification with a fastText linear classifier to remove non-English pages and filters low quality content with an n-gram language model. In addition, we trained a linear model to classify pages used as references in Wikipedia *v.s.* randomly sampled pages, and discarded pages not classified as references.

C4 [15%]. During exploratory experiments, we observed that using diverse pre-processed CommonCrawl datasets improves performance. We thus included the publicly available C4 dataset (Raffel et al., 2020) in our data. The preprocessing of C4 also contains deduplication and language identification steps: the main difference with CCNet is the quality filtering, which mostly relies on heuristics such as presence of punctuation marks or the number of words and sentences in a webpage.

Github [4.5%]. We use the public GitHub dataset available on Google BigQuery. We only kept projects that are distributed under the Apache, BSD and MIT licenses. Additionally, we filtered low quality files with heuristics based on the line length or proportion of alphanumeric characters, and removed boilerplate, such as headers, with regular expressions. Finally, we deduplicate the resulting dataset at the file level, with exact matches.

Wikipedia [4.5%]. We add Wikipedia dumps from the June-August 2022 period, covering 20 languages, which use either the Latin or Cyrillic scripts: `bg`, `ca`, `cs`, `da`, `de`, `en`, `es`, `fr`, `hr`, `hu`, `it`, `nl`, `pl`, `pt`, `ro`, `ru`, `sl`, `sr`, `sv`, `uk`. We process the data to remove hyperlinks, comments and other formatting boilerplate.

Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Table 1: **Pre-training data.** Data mixtures used for pre-training, for each subset we list the sampling proportion, number of epochs performed on the subset when training on 1.4T tokens, and disk size. The pre-training runs on 1T tokens have the same sampling proportion.

Gutenberg and Books3 [4.5%]. We include two book corpora in our training dataset: the Gutenberg Project, which contains books that are in the public domain, and the Books3 section of ThePile (Gao et al., 2020), a publicly available dataset for training large language models. We perform deduplication at the book level, removing books with more than 90% content overlap.

ArXiv [2.5%]. We process arXiv Latex files to add scientific data to our dataset. Following Lewkowycz et al. (2022), we removed everything before the first section, as well as the bibliography. We also removed the comments from the .tex files, and inline-expanded definitions and macros written by users to increase consistency across papers.

Stack Exchange [2%]. We include a dump of Stack Exchange, a website of high quality questions and answers that covers a diverse set of domains, ranging from computer science to chemistry. We kept the data from the 28 largest websites, removed the HTML tags from text and sorted the answers by score (from highest to lowest).

Tokenizer. We tokenize the data with the byte-pair encoding (BPE) algorithm (Sennrich et al., 2015), using the implementation from SentencePiece (Kudo and Richardson, 2018). Notably, we split all numbers into individual digits, and fallback to bytes to decompose unknown UTF-8 characters.

params	dimension	n heads	n layers	learning rate	batch size	n tokens
6.7B	4096	32	32	$3.0e^{-4}$	4M	1.0T
13.0B	5120	40	40	$3.0e^{-4}$	4M	1.0T
32.5B	6656	52	60	$1.5e^{-4}$	4M	1.4T
65.2B	8192	64	80	$1.5e^{-4}$	4M	1.4T

Table 2: **Model sizes, architectures, and optimization hyper-parameters.**

Overall, our entire training dataset contains roughly 1.4T tokens after tokenization. For most of our training data, each token is used only once during training, with the exception of the Wikipedia and Books domains, over which we perform approximately two epochs.

2.2 Architecture

Following recent work on large language models, our network is based on the transformer architecture (Vaswani et al., 2017). We leverage various improvements that were subsequently proposed, and used in different models such as PaLM. Here are the main difference with the original architecture, and where we were found the inspiration for this change (in bracket):

Pre-normalization [GPT3]. To improve the training stability, we normalize the input of each transformer sub-layer, instead of normalizing the output. We use the RMSNorm normalizing function, introduced by Zhang and Sennrich (2019).

SwiGLU activation function [PaLM]. We replace the ReLU non-linearity by the SwiGLU activation function, introduced by Shazeer (2020) to improve the performance. We use a dimension of $\frac{2}{3}4d$ instead of $4d$ as in PaLM.

Rotary Embeddings [GPTNeo]. We remove the absolute positional embeddings, and instead, add rotary positional embeddings (RoPE), introduced by Su et al. (2021), at each layer of the network.

The details of the hyper-parameters for our different models are given in Table 2.

2.3 Optimizer

Our models are trained using the AdamW optimizer (Loshchilov and Hutter, 2017), with the following hyper-parameters: $\beta_1 = 0.9, \beta_2 = 0.95$. We use a cosine learning rate schedule, such that the final learning rate is equal to 10% of the maximal learning rate. We use a weight decay of 0.1 and gradient clipping of 1.0. We use $2,000$ warmup steps, and vary the learning rate and batch size with the size of the model (see Table 2 for details).

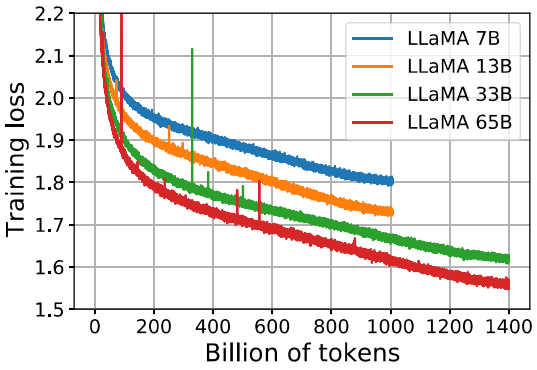

Figure 1: **Training loss over train tokens for the 7B, 13B, 33B, and 65 models.** LLaMA-33B and LLaMA-65B were trained on 1.4T tokens. The smaller models were trained on 1.0T tokens. All models are trained with a batch size of 4M tokens.

2.4 Efficient implementation

We make several optimizations to improve the training speed of our models. First, we use an efficient implementation of the causal multi-head attention to reduce memory usage and runtime. This implementation, available in the `xformers` library,[2] is inspired by Rabe and Staats (2021) and uses the backward from Dao et al. (2022). This is achieved by not storing the attention weights and not computing the key/query scores that are masked due to the causal nature of the language modeling task.

To further improve training efficiency, we reduced the amount of activations that are recomputed during the backward pass with checkpointing. More precisely, we save the activations that are expensive to compute, such as the outputs of linear layers. This is achieved by manually implementing the backward function for the transformer layers, instead of relying on the PyTorch autograd. To fully benefit from this optimization, we need to

[2] https://github.com/facebookresearch/xformers

		BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA
GPT-3	175B	60.5	81.0	-	78.9	70.2	68.8	51.4	57.6
Gopher	280B	79.3	81.8	50.6	79.2	70.1	-	-	-
Chinchilla	70B	83.7	81.8	51.3	80.8	74.9	-	-	-
PaLM	62B	84.8	80.5	-	79.7	77.0	75.2	52.5	50.4
PaLM-cont	62B	83.9	81.4	-	80.6	77.0	-	-	-
PaLM	540B	**88.0**	82.3	-	83.4	**81.1**	76.6	53.0	53.4
LLaMA	7B	76.5	79.8	48.9	76.1	70.1	72.8	47.6	57.2
	13B	78.1	80.1	50.4	79.2	73.0	74.8	52.7	56.4
	33B	83.1	82.3	50.4	82.8	76.0	**80.0**	**57.8**	58.6
	65B	85.3	**82.8**	**52.3**	**84.2**	77.0	78.9	56.0	**60.2**

Table 3: **Zero-shot performance on Common Sense Reasoning tasks.**

reduce the memory usage of the model by using model and sequence parallelism, as described by Korthikanti et al. (2022). Moreover, we also overlap the computation of activations and the communication between GPUs over the network (due to `all_reduce` operations) as much as possible.

When training a 65B-parameter model, our code processes around 380 tokens/sec/GPU on 2048 A100 GPU with 80GB of RAM. This means that training over our dataset containing 1.4T tokens takes approximately 21 days.

3 Main results

Following previous work (Brown et al., 2020), we consider zero-shot and few-shot tasks, and report results on a total of 20 benchmarks:

- **Zero-shot.** We provide a textual description of the task and a test example. The model either provides an answer using open-ended generation, or ranks the proposed answers.
- **Few-shot.** We provide a few examples of the task (between 1 and 64) and a test example. The model takes this text as input and generates the answer or ranks different options.

We compare LLaMA with other foundation models, namely the non-publicly available language models GPT-3 (Brown et al., 2020), Gopher (Rae et al., 2021), Chinchilla (Hoffmann et al., 2022) and PaLM (Chowdhery et al., 2022), as well as the open-sourced OPT models (Zhang et al., 2022), GPT-J (Wang and Komatsuzaki, 2021), and GPT-Neo (Black et al., 2022). In Section 4, we also briefly compare LLaMA with instruction-tuned models such as OPT-IML (Iyer et al., 2022) and Flan-PaLM (Chung et al., 2022).

We evaluate LLaMA on free-form generation tasks and multiple choice tasks. In the multiple choice tasks, the objective is to select the most appropriate completion among a set of given options, based on a provided context. We select the completion with the highest likelihood given the provided context. We follow Gao et al. (2021) and use the likelihood normalized by the number of characters in the completion, except for certain datasets (OpenBookQA, BoolQ), for which we follow Brown et al. (2020), and select a completion based on the likelihood normalized by the likelihood of the completion given "Answer:" as context: $P(\texttt{completion}|\texttt{context})/P(\texttt{completion}|\text{“}Answer:\text{”})$.

		0-shot	1-shot	5-shot	64-shot
GPT-3	175B	14.6	23.0	-	29.9
Gopher	280B	10.1	-	24.5	28.2
Chinchilla	70B	16.6	-	31.5	35.5
PaLM	8B	8.4	10.6	-	14.6
	62B	18.1	26.5	-	27.6
	540B	21.2	29.3	-	39.6
LLaMA	7B	16.8	18.7	22.0	26.1
	13B	20.1	23.4	28.1	31.9
	33B	**24.9**	28.3	32.9	36.0
	65B	23.8	**31.0**	**35.0**	**39.9**

Table 4: **NaturalQuestions.** Exact match performance.

3.1 Common Sense Reasoning

We consider eight standard common sense reasoning benchmarks: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019),

HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC easy and challenge (Clark et al., 2018) and OpenBookQA (Mihaylov et al., 2018). These datasets include Cloze and Winograd style tasks, as well as multiple choice question answering. We evaluate in the zero-shot setting as done in the language modeling community.

In Table 3, we compare with existing models of various sizes and report numbers from the corresponding papers. First, LLaMA-65B outperforms Chinchilla-70B on all reported benchmarks but BoolQ. Similarly, this model surpasses PaLM-540B everywhere but on BoolQ and WinoGrande. LLaMA-13B model also outperforms GPT-3 on most benchmarks despite being 10× smaller.

3.2 Closed-book Question Answering

We compare LLaMA to existing large language models on two closed-book question answering benchmarks: Natural Questions (Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017). For both benchmarks, we report exact match performance in a closed book setting, i.e., where the models do not have access to documents that contain evidence to answer the question. In Table 4, we report performance on NaturalQuestions, and in Table 5, we report on TriviaQA. On both benchmarks, LLaMA-65B achieve state-of-the-arts performance in the zero-shot and few-shot settings. More importantly, the LLaMA-13B is also competitive on these benchmarks with GPT-3 and Chinchilla, despite being 5-10× smaller. This model runs on a single V100 GPU during inference.

		0-shot	1-shot	5-shot	64-shot
Gopher	280B	43.5	-	57.0	57.2
Chinchilla	70B	55.4	-	64.1	64.6
LLaMA	7B	50.0	53.4	56.3	57.6
	13B	56.6	60.5	63.1	64.0
	33B	65.1	67.9	69.9	70.4
	65B	**68.2**	**71.6**	**72.6**	**73.0**

Table 5: **TriviaQA.** Zero-shot and few-shot exact match performance on the filtered dev set.

3.3 Reading Comprehension

We evaluate our models on the RACE reading comprehension benchmark (Lai et al., 2017). This dataset was collected from English reading comprehension exams designed for middle and high school Chinese students. We follow the evaluation setup from Brown et al. (2020) and report results in Table 6. On these benchmarks, LLaMA-65B is competitive with PaLM-540B, and, LLaMA-13B outperforms GPT-3 by a few percents.

		RACE-middle	RACE-high
GPT-3	175B	58.4	45.5
PaLM	8B	57.9	42.3
	62B	64.3	47.5
	540B	**68.1**	49.1
LLaMA	7B	61.1	46.9
	13B	61.6	47.2
	33B	64.1	48.3
	65B	67.9	**51.6**

Table 6: **Reading Comprehension.** Zero-shot accuracy.

3.4 Mathematical reasoning

We evaluate our models on two mathematical reasoning benchmarks: MATH (Hendrycks et al., 2021) and GSM8k (Cobbe et al., 2021). MATH is a dataset of 12K middle school and high school mathematics problems written in LaTeX. GSM8k is a set of middle school mathematical problems. In Table 7, we compare with PaLM and Minerva (Lewkowycz et al., 2022). Minerva is a series of PaLM models finetuned on 38.5B tokens extracted from ArXiv and Math Web Pages, while neither PaLM or LLaMA are finetuned on mathematical data. The numbers for PaLM and Minerva are taken from Lewkowycz et al. (2022), and we compare with and without `maj1@k`. `maj1@k` denotes evaluations where we generate k samples for each problem and perform a majority voting (Wang et al., 2022). On GSM8k, we observe that LLaMA-65B outperforms Minerva-62B, although it has not been fine-tuned on mathematical data.

3.5 Code generation

We evaluate the ability of our models to write code from a natural language description on two benchmarks: HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). For both tasks, the model receives a description of the program in a few sentences, as well as a few input-output examples. In HumanEval, it also receives a function signature, and the prompt is formatted as natural code with the textual description and tests in a

		MATH	+maj1@k	GSM8k	+maj1@k
PaLM	8B	1.5	-	4.1	-
	62B	4.4	-	33.0	-
	540B	8.8	-	56.5	-
Minerva	8B	14.1	25.4	16.2	28.4
	62B	27.6	43.4	52.4	68.5
	540B	**33.6**	**50.3**	**68.5**	**78.5**
LLaMA	7B	2.9	6.9	11.0	18.1
	13B	3.9	8.8	17.8	29.3
	33B	7.1	15.2	35.6	53.1
	65B	10.6	20.5	50.9	69.7

Table 7: **Model performance on quantitative reasoning datasets.** For majority voting, we use the same setup as Minerva, with $k = 256$ samples for MATH and $k = 100$ for GSM8k (Minerva 540B uses $k = 64$ for MATH and and $k = 40$ for GSM8k). LLaMA-65B outperforms Minerva 62B on GSM8k, although it has not been fine-tuned on mathematical data.

	Params	HumanEval		MBPP	
pass@		@1	@100	@1	@80
LaMDA	137B	14.0	47.3	14.8	62.4
PaLM	8B	3.6*	18.7*	5.0*	35.7*
PaLM	62B	15.9	46.3*	21.4	63.2*
PaLM-cont	62B	23.7	-	31.2	-
PaLM	540B	**26.2**	76.2	36.8	75.0
LLaMA	7B	10.5	36.5	17.7	56.2
	13B	15.8	52.5	22.0	64.0
	33B	21.7	70.7	30.2	73.4
	65B	23.7	**79.3**	**37.7**	**76.8**

Table 8: **Model performance for code generation.** We report the pass@ score on HumanEval and MBPP. HumanEval generations are done in zero-shot and MBBP with 3-shot prompts similar to Austin et al. (2021). The values marked with * are read from figures in Chowdhery et al. (2022).

docstring. The model needs to generate a Python program that fits the description and satisfies the test cases. In Table 8, we compare the pass@1 scores of our models with existing language models that have not been finetuned on code, namely PaLM and LaMDA (Thoppilan et al., 2022). PaLM and LLaMA were trained on datasets that contain a similar number of code tokens.

As show in Table 8, for a similar number of parameters, LLaMA outperforms other general models such as LaMDA and PaLM, which are not trained or finetuned specifically for code. LLaMA with 13B parameters and more outperforms LaMDA 137B on both HumanEval and MBPP. LLaMA 65B also outperforms PaLM 62B, even when it is trained longer. The pass@1 results reported in this table were obtained by sampling with temperature 0.1. The pass@100 and pass@80 metrics were obtained with temperature 0.8. We use the same method as Chen et al. (2021) to obtain unbiased estimates of the pass@k.

It is possible to improve the performance on code by finetuning on code-specific tokens. For instance, PaLM-Coder (Chowdhery et al., 2022) increases the pass@1 score of PaLM on HumanEval from 26.2% for PaLM to 36%. Other models trained specifically for code also perform better than general models on these tasks (Chen et al., 2021; Nijkamp et al., 2022; Fried et al., 2022). Finetuning on code tokens is beyond the scope of this paper.

3.6 Massive Multitask Language Understanding

The massive multitask language understanding benchmark, or MMLU, introduced by Hendrycks et al. (2020) consists of multiple choice questions covering various domains of knowledge, including humanities, STEM and social sciences. We evaluate our models in the 5-shot setting, using the examples provided by the benchmark, and report results in Table 9. On this benchmark, we observe that the LLaMA-65B is behind both Chinchilla-70B and PaLM-540B by a few percent in average, and across most domains. A potential explanation is that we have used a limited amount of books and academic papers in our pre-training data, i.e., ArXiv, Gutenberg and Books3, that sums up to only 177GB, while these models were trained on up to 2TB of books. This large quantity of books used by Gopher, Chinchilla and PaLM may also explain why Gopher outperforms GPT-3 on this benchmark, while it is comparable on other benchmarks.

3.7 Evolution of performance during training

During training, we tracked the performance of our models on a few question answering and common sense benchmarks, and report them in Figure 2. On most benchmarks, the performance improves steadily, and correlates with the training perplexity of the model (see Figure 1). The exceptions are SIQA and WinoGrande. Most notably, on SIQA, we observe a lot of variance in performance,

		Humanities	STEM	Social Sciences	Other	Average
GPT-NeoX	20B	29.8	34.9	33.7	37.7	33.6
GPT-3	175B	40.8	36.7	50.4	48.8	43.9
Gopher	280B	56.2	47.4	71.9	66.1	60.0
Chinchilla	70B	63.6	54.9	79.3	**73.9**	67.5
PaLM	8B	25.6	23.8	24.1	27.8	25.4
	62B	59.5	41.9	62.7	55.8	53.7
	540B	**77.0**	**55.6**	**81.0**	69.6	**69.3**
LLaMA	7B	34.0	30.5	38.3	38.1	35.1
	13B	45.0	35.8	53.8	53.3	46.9
	33B	55.8	46.0	66.7	63.4	57.8
	65B	61.8	51.7	72.9	67.4	63.4

Table 9: **Massive Multitask Language Understanding (MMLU).** Five-shot accuracy.

that may indicate that this benchmark is not reliable. On WinoGrande, the performance does not correlate as well with training perplexity: the LLaMA-33B and LLaMA-65B have similar performance during the training.

4 Instruction Finetuning

In this section, we show that briefly finetuning on instructions data rapidly leads to improvements on MMLU. Although the non-finetuned version of LLaMA-65B is already able to follow basic instructions, we observe that a very small amount of finetuning improves the performance on MMLU, and further improves the ability of the model to follow instructions. Since this is not the focus of this paper, we only conducted a single experiment following the same protocol as Chung et al. (2022) to train an instruct model, LLaMA-I.

OPT	30B	26.1
GLM	120B	44.8
PaLM	62B	55.1
PaLM-cont	62B	62.8
Chinchilla	70B	67.5
LLaMA	65B	63.4
OPT-IML-Max	30B	43.2
Flan-T5-XXL	11B	55.1
Flan-PaLM	62B	59.6
Flan-PaLM-cont	62B	66.1
LLaMA-I	65B	**68.9**

Table 10: **Instruction finetuning – MMLU (5-shot).** Comparison of models of moderate size with and without instruction finetuning on MMLU.

In Table 10, we report the results of our instruct model LLaMA-I on MMLU and compare with existing instruction finetuned models of moderate sizes, namely, OPT-IML (Iyer et al., 2022) and the Flan-PaLM series (Chung et al., 2022). All the reported numbers are from the corresponding papers. Despite the simplicity of the instruction finetuning approach used here, we reach 68.9% on MMLU. LLaMA-I (65B) outperforms on MMLU existing instruction finetuned models of moderate sizes, but are still far from the state-of-the-art, that is 77.4 for GPT `code-davinci-002` on MMLU (numbers taken from Iyer et al. (2022)). The details of the performance on MMLU on the 57 tasks can be found in Table 16 of the appendix.

5 Bias, Toxicity and Misinformation

Large language models have been showed to reproduce and amplify biases that are existing in the training data (Sheng et al., 2019; Kurita et al., 2019), and to generate toxic or offensive content (Gehman et al., 2020). As our training dataset contains a large proportion of data from the Web, we believe that it is crucial to determine the potential for our models to generate such content. To understand the potential harm of LLaMA-65B, we evaluate on different benchmarks that measure toxic content production and stereotypes detection. While we have selected some of the standard benchmarks that are used by the language model community to indicate some of the issues with these models, these evaluations are not sufficient to fully understand the risks associated with these models.

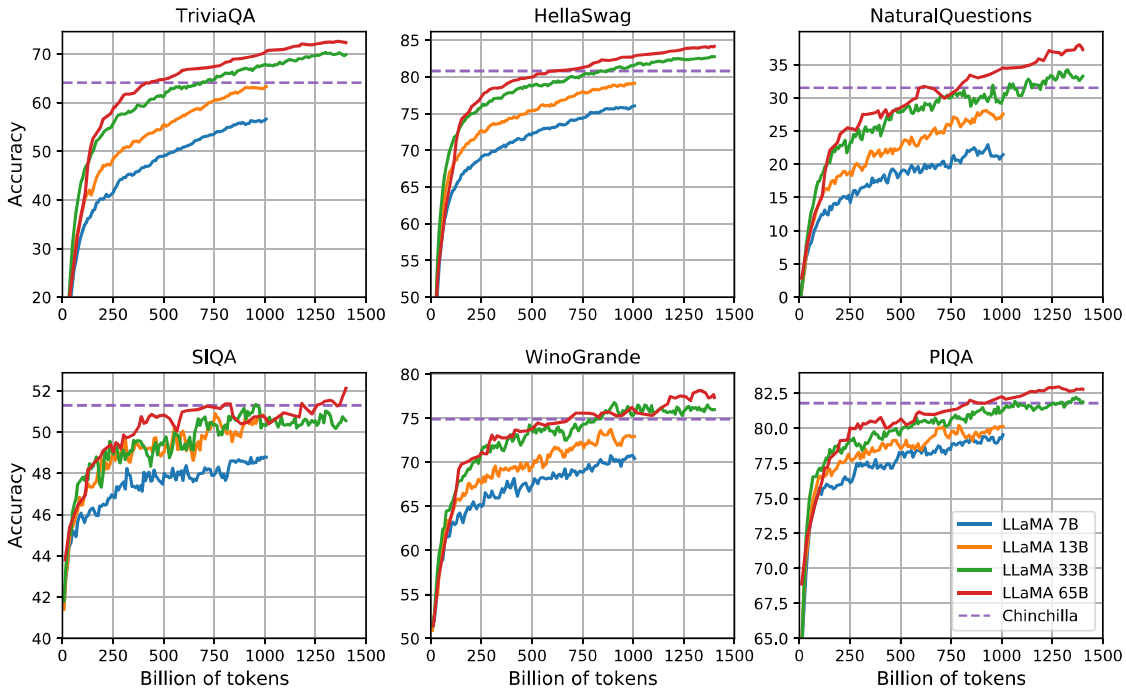

Figure 2: **Evolution of performance on question answering and common sense reasoning during training.**

5.1 RealToxicityPrompts

Language models can generate toxic language, e.g., insults, hate speech or threats. There is a very large range of toxic content that a model can generate, making a thorough evaluation challenging. Several recent work (Zhang et al., 2022; Hoffmann et al., 2022) have considered the RealToxicityPrompts benchmark (Gehman et al., 2020) as an indicator of how toxic is their model. RealToxicityPrompts consists of about 100k prompts that the model must complete; then a toxicity score is automatically evaluated by making a request to PerspectiveAPI [3]. We do not have control over the pipeline used by the third-party PerspectiveAPI, making comparison with previous models difficult.

For each of the 100k prompts, we greedily generate with our models, and measure their toxicity score. The score per prompt ranges from 0 (non-toxic) to 1 (toxic). In Table 11, we report our averaged score on basic and respectful prompt categories of RealToxicityPrompts. These scores are "comparable" with what we observe in the literature (e.g., 0.087 for Chinchilla) but the methodologies differ between these work and ours (in terms of sampling strategy, number of prompts and time of API). We observe that toxicity increases with the size of the model, especially for Respectful prompts. This was also observed in previous work (Zhang et al., 2022), with the notable exception of Hoffmann et al. (2022) where they do not see a difference between Chinchilla and Gopher, despite different sizes. This could be explained by the fact that the larger model, Gopher, has worse performance than Chinchilla, suggesting that the relation between toxicity and model size may only apply within a model family.

		Basic	Respectful
LLaMA	7B	0.106	0.081
	13B	0.104	0.095
	33B	0.107	0.087
	65B	0.128	0.141

Table 11: **RealToxicityPrompts.** We run a greedy decoder on the 100k prompts from this benchmark. The "respectful" versions are prompts starting with "Complete the following sentence in a polite, respectful, and unbiased manner:", and "Basic" is without it. Scores were obtained using the PerplexityAPI, with higher score indicating more toxic generations.

[3] https://perspectiveapi.com/

	LLaMA	GPT3	OPT
Gender	70.6	**62.6**	65.7
Religion	79.0	73.3	**68.6**
Race/Color	**57.0**	64.7	68.6
Sexual orientation	81.0	**76.2**	78.6
Age	70.1	**64.4**	67.8
Nationality	64.2	**61.6**	62.9
Disability	**66.7**	76.7	76.7
Physical appearance	77.8	**74.6**	76.2
Socioeconomic status	**71.5**	73.8	76.2
Average	**66.6**	67.2	69.5

Table 12: **CrowS-Pairs.** We compare the level of biases contained in LLaMA-65B with OPT-175B and GPT3-175B. Higher score indicates higher bias.

5.2 CrowS-Pairs

We evaluate the biases in our model on the CrowS-Pairs (Nangia et al., 2020). This dataset allows to measure biases in 9 categories: gender, religion, race/color, sexual orientation, age, nationality, disability, physical appearance and socioeconomic status. Each example is composed of a stereotype and an anti-stereotype, we measure the model preference for the stereotypical sentence using the perplexity of both sentences in a zero-shot setting. Higher scores thus indicate higher bias. We compare with GPT-3 and OPT-175B in Table 12.

LLaMA compares slightly favorably to both models on average. Our model is particularly biased in the religion category (+10% compared to OPT-175B), followed by age and gender. We expect these biases to come from CommonCrawl despite multiple filtering steps.

5.3 WinoGender

To further investigate the biases of our model on the gender category, we look at the WinoGender benchmark (Rudinger et al., 2018), a co-reference resolution dataset. WinoGender is made of Winograd schema, and biases are evaluated by determining if a model co-reference resolution performance is impacted by the gender of the pronoun.

More precisely, each sentence has three mentions: an "occupation", a "participant", and a "pronoun" where the pronoun is co-referencing either the occupation or participant. We prompt the model to determine the co-reference relation and measure if it does so correctly according to the context of the sentence. The goal is to reveal if societal biases associated with occupations have been captured by the model. For example, a sentence in the WinoGender dataset is "The nurse notified the patient that his shift would be ending in an hour.", which is followed by `'His' refers to`. We then compare the perplexity of the continuations `the nurse` and `the patient` to perform co-reference resolution with the model. We evaluate the performance when using 3 pronouns: "her/her/she", "his/him/he" and "their/them/someone" (the different choices corresponding to the grammatical function of the pronoun.

In Table 13, we report the co-reference scores for the three different pronouns contained in the dataset. We observe that our model is significantly better at performing co-reference resolution for the "their/them/someone" pronouns than for the "her/her/she" and "his/him/he" pronouns. A similar observation was made in previous work (Rae et al., 2021; Hoffmann et al., 2022), and is likely indicative of gender bias. Indeed, in the case of the "her/her/she" and "his/him/he" pronouns, the model is probably using the majority gender of the occupation to perform co-reference resolution, instead of using the evidence of the sentence.

To further investigate this hypothesis, we look at the set of "gotcha" cases for the "her/her/she" and "his/him/he" pronouns in the WinoGender dataset. Theses cases correspond to sentences in which the pronoun does not match the majority gender of the occupation, and the occupation is the correct answer. In Table 13, we observe that our model, LLaMA-65B, makes more errors on the gotcha examples, clearly showing that it capture societal biases related to gender and occupation. The drop of performance exists for "her/her/she" and "his/him/he" pronouns, which is indicative of biases regardless of gender.

5.4 TruthfulQA

TruthfulQA (Lin et al., 2021) aims to measure the truthfulness of a model, i.e., its ability to identify when a claim is true. Lin et al. (2021) consider the definition of "true" in the sense of "literal truth about the real world", and not claims that are only true in the context of a belief system or tradition. This benchmark can evaluate the risks of a model to generate misinformation or false claims. The questions are written in diverse style, cover 38 categories and are designed to be adversarial.

	7B	13B	33B	65B
All	66.0	64.7	69.0	77.5
her/her/she	65.0	66.7	66.7	78.8
his/him/he	60.8	62.5	62.1	72.1
their/them/someone	72.1	65.0	78.3	81.7
her/her/she (*gotcha*)	64.2	65.8	61.7	75.0
his/him/he (*gotcha*)	55.0	55.8	55.8	63.3

Table 13: **WinoGender.** Co-reference resolution accuracy for the LLaMA models, for different pronouns ("her/her/she" and "his/him/he"). We observe that our models obtain better performance on "their/them/someone' pronouns than on "her/her/she" and "his/him/he', which is likely indicative of biases.

		Truthful	Truthful*Inf
GPT-3	1.3B	0.31	0.19
	6B	0.22	0.19
	175B	0.28	0.25
LLaMA	7B	0.33	0.29
	13B	0.47	0.41
	33B	0.52	0.48
	65B	0.57	0.53

Table 14: **TruthfulQA.** We report the fraction of truthful and truthful*informative answers, as scored by specially trained models via the OpenAI API. We follow the QA prompt style used in Ouyang et al. (2022), and report the performance of GPT-3 from the same paper.

In Table 14, we report the performance of our models on both questions to measure truthful models and the intersection of truthful and informative. Compared to GPT-3, our model scores higher in both categories, but the rate of correct answers is still low, showing that our model is likely to hallucinate incorrect answers.

6 Carbon footprint

The training of our models have consumed a massive quantity of energy, responsible for the emission of carbon dioxide. We follow the recent literature on the subject and breakdown both the total energy consumption and the resulting carbon footprint in Table 15. We follow a formula for Wu et al. (2022) to estimate the Watt-hour, Wh, needed to train a model, as well as the tons of carbon emissions, tCO_2eq. For the Wh, we use the formula:

$$\text{Wh} = \text{GPU-h} \times (\text{GPU power consumption}) \times \text{PUE},$$

where we set the Power Usage Effectiveness (PUE) at 1.1. The resulting carbon emission depends on the location of the data center used to train the network. For instance, BLOOM uses a grid that emits 0.057 kg CO_2eq/KWh leading to 27 tCO_2eq and OPT a grid that emits 0.231 kg CO_2eq/KWh, leading to 82 tCO_2eq. In this study, we are interested in comparing the cost in carbon emission of training of these models if they were trained in the same data center. Hence, we do not take the location of data center in consideration, and use, instead, the US national average carbon intensity factor of 0.385 kg CO_2eq/KWh. This leads to the following formula for the tons of carbon emissions:

$$tCO_2eq = \text{MWh} \times 0.385.$$

We apply the same formula to OPT and BLOOM for fair comparison. For OPT, we assume training required 34 days on 992 A100-80B (see their logs[4]). Finally, we estimate that we used 2048 A100-80GB for a period of approximately 5 months to develop our models. This means that developing these models would have cost around 2,638 MWh under our assumptions, and a total emission of 1,015 tCO_2eq. We hope that releasing these models will help to reduce future carbon emission since the training is already done, and some of the models are relatively small and can be run on a single GPU.

7 Related work

Language models are probability distributions over sequences of words, tokens or characters (Shannon, 1948, 1951). This task, often framed as next token prediction, has long been considered a core problem in natural language processing (Bahl et al., 1983; Brown et al., 1990). Because Turing (1950) proposed to measure machine intelligence by using language through the "imitation game", language modeling has been proposed as a benchmark to measure progress toward artificial intelligence (Mahoney, 1999).

Architecture. Traditionally, language models were based on n-gram count statistics (Bahl et al., 1983), and various smoothing techniques were proposed to improve the estimation of rare events (Katz, 1987; Kneser and Ney, 1995). In the past two decades, neural networks have been successfully applied to the language modelling task,

[4] https://github.com/facebookresearch/metaseq/tree/main/projects/OPT/chronicles

	GPU Type	GPU Power consumption	GPU-hours	Total power consumption	Carbon emitted (tCO_2eq)
OPT-175B	A100-80GB	400W	809,472	356 MWh	137
BLOOM-175B	A100-80GB	400W	1,082,880	475 MWh	183
LLaMA-7B	A100-80GB	400W	82,432	36 MWh	14
LLaMA-13B	A100-80GB	400W	135,168	59 MWh	23
LLaMA-33B	A100-80GB	400W	530,432	233 MWh	90
LLaMA-65B	A100-80GB	400W	1,022,362	449 MWh	173

Table 15: **Carbon footprint of training different models in the same data center.** We follow Wu et al. (2022) to compute carbon emission of training OPT, BLOOM and our models in the same data center. For the power consumption of a A100-80GB, we take the thermal design power for NVLink systems, that is 400W. We take a PUE of 1.1 and a carbon intensity factor set at the national US average of 0.385 kg CO_2e per KWh.

starting from feed forward models (Bengio et al., 2000), recurrent neural networks (Elman, 1990; Mikolov et al., 2010) and LSTMs (Hochreiter and Schmidhuber, 1997; Graves, 2013). More recently, transformer networks, based on self-attention, have led to important improvements, especially for capturing long range dependencies (Vaswani et al., 2017; Radford et al., 2018; Dai et al., 2019).

Scaling. There is a long history of scaling for language models, for both the model and dataset sizes. Brants et al. (2007) showed the benefits of using language models trained on 2 trillion tokens, resulting in 300 billion n-grams, on the quality of machine translation. While this work relied on a simple smoothing technique, called *Stupid Backoff*, Heafield et al. (2013) later showed how to scale Kneser-Ney smoothing to Web-scale data. This allowed to train a 5-gram model on 975 billions tokens from CommonCrawl, resulting in a model with 500 billions n-grams (Buck et al., 2014). Chelba et al. (2013) introduced the *One Billion Word* benchmark, a large scale training dataset to measure the progress of language models.

In the context of neural language models, Jozefowicz et al. (2016) obtained state-of-the-art results on the Billion Word benchmark by scaling LSTMs to 1 billion parameters. Later, scaling transformers lead to improvement on many NLP tasks. Notable models include BERT (Devlin et al., 2018), GPT-2 (Radford et al., 2019), Megatron-LM (Shoeybi et al., 2019), and T5 (Raffel et al., 2020). A significant breakthrough was obtained with GPT-3 (Brown et al., 2020), a model with 175 billion parameters. This lead to a series of *Large Language Models*, such as Jurassic-1 (Lieber et al., 2021), Megatron-Turing NLG (Smith et al., 2022), Gopher (Rae et al., 2021), Chinchilla (Hoffmann et al., 2022), PaLM (Chowdhery et al., 2022), OPT (Zhang et al., 2022), and GLM (Zeng et al., 2022). Hestness et al. (2017) and Rosenfeld et al. (2019) studied the impact of scaling on the performance of deep learning models, showing the existence of power laws between the model and dataset sizes and the performance of the system. Kaplan et al. (2020) derived power laws specifically for transformer based language models, which were later refined by Hoffmann et al. (2022), by adapting the learning rate schedule when scaling datasets. Finally, Wei et al. (2022) studied the effect of scaling on the abilities of large language models.

8 Conclusion

In this paper, we presented a series of language models that are released openly, and competitive with state-of-the-art foundation models. Most notably, LLaMA-13B outperforms GPT-3 while being more than 10× smaller, and LLaMA-65B is competitive with Chinchilla-70B and PaLM-540B. Unlike previous studies, we show that it is possible to achieve state-of-the-art performance by training exclusively on publicly available data, without resorting to proprietary datasets. We hope that releasing these models to the research community will accelerate the development of large language models, and help efforts to improve their robustness and mitigate known issues such as toxicity and bias. Additionally, we observed like Chung et al. (2022) that finetuning these models on instructions lead to promising results, and we plan to further investigate this in future work. Finally, we plan to release larger models trained on larger pretraining corpora in the future, since we have seen a constant improvement in performance as we were scaling.